STUDENT ENROLLMENT SYSTEM MODEL
- A PROPOSAL

A project report submitted to the Graduate School in partial
fulfilment of the requirements for the degree
Master of Science (Information Technology)
Universiti Utara Malaysia

by
Gee Kheng Chuan

© Gee Kheng Chuan, 2003. All rights reserved.

Fakulti Teknologi Maklumat
(*Faculty of Information Technology*)
Universiti Utara Malaysia

PERAKUAN KERJA KERTAS PROJEK
(Certificate of Projek Paper)

Saya, yang bertandatangan, memperakukan bahawa
(I, the undersigned, certify that)

GEE KHENG CHUAN

calon untuk Ijazah
(candidate for the degree of) **MSc. INFORMATION TECHNOLOGY**

telah mengemukakan kertas projek yang bertajuk
(has presented his/ her project paper of the following title)

STUDENT ENROLLMENT SYSTEM MODEL – A PROPOSAL

seperti yang tercatat di muka surat tajuk dan kulit kertas projek
(as it appears on the title page and front cover of project paper)

bahawa kertas projek tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan.
(that the project paper acceptable in form and content, and that a satisfactory knowledge of the filed is covered by the project paper).

Nama Penyelia Utama
(Name of Main Supervisor): **AZIDA ZAINOL**

Tandatangan
(Signature) : _____

Tarikh
(Date) : _____

PERMISSION TO USE

In presenting this project report in partial fulfillment of the requirement for a postgraduate degree from University Utara Malaysia, I agree that the University Library may make it freely available for inspection. I further agree that permission for copying of this project report in any manner, in whole or in part, for scholarly purposes may be granted by my supervisor(s), in their absence, by the Dean of the Graduate School. It is understood that any copying or publication or use of this project report or parts thereof for financial gain shall not be allowed without my written permission. It is understood that due recognition shall be given to me and to University Utara Malaysia for any scholarly use which may be made of any material from my project report.

Request for permission to copy or to make use of material in this project report, in whole or in part should be addressed to:

Dean of Graduate School
University Utara Malaysia
06010 UUM Sintok
Kedah Darul Aman.

ABSTRAK

Pengurusan data pelajar terrnasuk butir-butir diri pelajar merupakan satu aspek penting dalam pentadbiran sekolah. Biasanya butir-butir pelajar dikumpul semasa pendaftaran kemasukan ke sekolah. Selain daripada sistem manual, tidak terdapat sebarang model yang khusus sebagai panduan untuk proses pendaftaran. Pendaftaran secara manual amat membebankan pentadbir sekolah terutamanya terhadap guru tingkatan. Pengurusan rekod pelajar yang sedia ada juga member-i tekanan kepada pihak pentadbir atasan. Project ini mengenalpasti dan mencadangkan satu model untuk mendaftar kemasukan pelajar ke sekolah-sekolah menengah awam di Malaysia. Model ini membenarkan perkongsian tanggungjawab antara pentadbir atasan, pentadbir biasa dan pelajar dalam hal urusan rekod kemasukan pelajar. Pengagihan semula tanggungjawab dan beban kerja dalam hal ini bergantung pada teknologi berasaskan web.

ABSTRACT

Management of student data including their personal particulars is an important area in any school administration. Usually, the personal particulars are collected during enrollment process. Apart from the manual system, there exists no specific model to guide the enrollment. The traditional manual student registration system place a heavy burden towards school administrators especially the class teachers. The management of student records already input stresses top level school managers. This project identifies and proposes a model in enrolling students to Malaysian government secondary school that allows the sharing of responsibilities among top school management, administrators, and students. The key to this redistribution of burdens and responsibilities is the web-based technologies.

ACKNOWLEDGEMENTS

In carrying out this project in partial fulfillment of the requirement for a postgraduate degree from University Utara Malaysia, many parties had helped. The writer wishes to thank the following parties:

- Cik Azida Bt. Zainol, the project advisor that has guided the writer along from beginning till the end.
- The panel of project evaluators who has given their valuable suggestions and comments during the project seminar.
- The principal, senior assistants and senior teachers of SMK Seri Keledang, Menglembu that have allowed me to look through the relevant documents and participated in interviews at requirement capture phase and tried out the prototype.
- The selected student of SMK Seri Keledang, Menglembu that have participated in interviews at requirement capture phase tried out the prototype.

Thank you.

TABLE OF CONTENTS

TITLE PAGE I

CERTIFICATION OF PROJECT PAPER II

PERMISSION TO USE III

ABSTRAK IV

ABSTRACT V

ACKNOWLEDGEMENT VI

TABLE OF CONTENTS VII

LIST OF TABLES IX

LIST OF FIGURES X

1.0 INTRODUCTION 1

1.1 Literature Review 1

1.2 Problem Statements 5

1.3 Project Objective and Scope 6

2.0 METHODOLOGY 8

2.1 Methodology Overview 8

2.2 Requirement Capture 8

2.2.1 Fact Finding 9

2.3 Requirement Analysis 10

2.4 Preliminary Design 10

2.4.1 Identifying Sub-System and Its Major Component 11

2.4.2 Identifying Any Inherent Concurrency 11

2.4.3 Allocating Sub-System to Processors, ... 12

2.4.4 Select a Data Management Strategy 13

2.4.5 Selecting a Strategy and Standard for Human-Computer Interaction 13

2.4.6 Specify Code Development Standards 14

2.4.7 Plan the Control Aspects of the Application 14

2.4.8 Produce Test Plan 14

2.4.9 Set Priority For Design Trade-Off. 15

2.5 Detail Design 15

3.0 FINDINGS AND DISCUSSION 16
3.1 Findings 16
3.1.1 Finding Overview 16
3.1.2 Model Showing System Capability and Division of Roles...18
3.1.3 Model Showing Internal Classes 40
3.1.4 Class Diagrams 42
3.1.5 Data Specifications 46
3.1.6 Testing 47
3.2 Discussion 50

4.0 SIGNIFICANCE OF STUDY 51

5.0 CONCLUSION AND RECOMMENDATIONS.....52

REFERENCES 53

APPENDICES
1. Student Enrollment System_Software Requirement Specification
2. Student Enrollment System_Software Design Document
3. Student Enrollment System_Software Installation Guide
4. Student Enrollment System_Source Codes

ATTACHMENTS
1. Student Enrollment System Installer and Project Report. (Delivered in 650 Mb compact disk)

LIST OF TABLES

Table 3.1: User Requirements of Student Enrollment System 16

Table 3.2: Description Use Case System Administration 20

Table 3.3: Description Use Case Student Enrollment. 22

Table 3.4: Description Use Case Search Student Record 23

Table 3.5: Description Use Case Generate Attendance Roster. 24

Table 3.6 Data Specifications 46

Table 3.7 Test Plan and Results 49

LIST OF FIGURES

Figure 2.1 Deployment Diagram for SES System.....12
Figure 3.1: Overall functions of Student Enrollment System 17
Figure 3.2 Use Case Package Diagram for SES 19
Figure 3.3 Use Case Diagrams for System Administration 20
Figure 3.4 Use Case Diagram for Student Enrollment Use Case Package. 21
Figure 3.5 Use Case Diagram for Search Student Record Use Case Package....22
Figure 3.6 Use Case Diagram for Generate Attendance Roster Use Case Package. 23
Figure 3.7 Sequence Diagram System Administrator Login 25
Figure 3.8 Sequence Diagram System Administrator Login /E1_if password invalid. 25
Figure 3.9 Sequence Diagram School Administrator Login 25
Figure 3.10 Sequence Diagram School Administrator Login /E1_if password invalid 26
Figure 3.11 Sequence Diagram Student Login 26
Figure 3.12 Sequence Diagram Student Login/E1_if password invalid 26
Figure 3.13 Sequence Diagram Student Login /A1_if student particulars not submitted 27
Figure 3.14 Sequence Diagram System Administrator Logout 27
Figure 3.15 Sequence Diagram School Administrator Logout 27
Figure 3.16 Sequence Diagram Student Logout 28
Figure 3.17 Sequence Diagram System Administrator Change Password 28
Figure 3.18 Sequence Diagram School Administrator Change Password 28
Figure 3.19 Sequence Diagram Student Change Password 29
Figure 3.20 Sequence Diagram Student Change Password /A1_if student particulars not Submitted 29
Figure 3.21 Sequence Diagram Activate Staff 29
Figure 3.22 Sequence Diagram Activate Staff/E1_if user already activate 30
Figure 3.23 Sequence Diagram Activate Student 30
Figure 3.24 Sequence Diagram Activate Student /E1_if student already activated 30

Figure 3.25 Sequence Diagram Deactivate Staff 31
Figure 3.26 Sequence Diagram Deactivate Student 31
Figure 3.27 Sequence Diagram Add Student Particulars 32
Figure 3.28 Sequence Diagram Delete Student Record 32
Figure 3.29 Sequence Diagram Delete Student Record /E1_if confirmation denied 32
Figure 3.30 Sequence Diagram Update Student Record 33
Figure 3.31 Sequence Diagram Generate Class 33
Figure 3.32 Sequence Diagram Add Class Member 33
Figure 3.33 Sequence Diagram Add Class Member /E1_if student record not found 34
Figure 3.34 Sequence Diagram Search Student Record 34
Figure 3.35 Sequence Diagram Search Student Record /E1_if student record not found. 34
Figure 3.36 Sequence Diagram Print Individual Student Record 35
Figure 3.37 Sequence Diagram Print Individual Student Record /E1_if user cancels transacrtion 35
Figure 3.38 Sequence Diagram Print Student Record by Class 36
Figure 3.39 Sequence Diagram Print Student Record by Class /E1_if class not found 36
Figure 3.40 Sequence Diagram Print Student Record by Class /A1_if user cancels transaction 37
Figure 3.41 Sequence Diagram Print Student Record by Intake 37
Figure 3.42 Sequence Diagram Print Student Record by Intake /E1_if record not found 38
Figure 3.43 Sequence Diagram Print Student Record by Intake /A1_if user cancels transaction 38
Figure 3.44 Sequence Diagram Print Class List 39
Figure 3.45 Sequence Diagram Print Class List/E1_if class not found 39
Figure 3.46 Sequence Diagram Print Class List /A1_if user cancels transaction 40
Figure 3.47 Visibility of Presentation Layer with Other Package 41
Figure 3.48: Visibility of Application Logic Layer with Other Packages 41
Figure 3.49: Visibility of Storage Layer with Other Packages 42

Figure 3.50 Presentation Layer Package 43
Figure 3.51 Application Logic Layer Package 44
Figure 3.52 Storage Layer package 45

1 INTRODUCTION

This project identifies and suggests a model in solving the problem faced in registering new student in government secondary school in Malaysia. A prototype of student enrollment system was developed to test the model. This project report is part of the project deliverables. This report includes (1) a project description that includes literature review, problem statement, and project's objective and scope, (2) Methodology, (3) Findings, (4) Significance of Study, and (5) Conclusion.

1.1 Literature Review

Bennett, McRobb and Farmer (2002) say that a model represent something but they differ from the the things that they are present. They further stress that a models have the following characteristics:

- They are quicker and easy to built
- They can be used in simulations to learn more about the thing it represent
- They can evolve as we learn more about the thing it represents
- We can choose which details to represent in a model and which to ignore. It is an abstraction.
- A model can represent real or imaginary things from any domain.

Streatfield (1992) opines that the range of tasks need for computer support in school administration is extensive. Among the task listed are:

- Enrolling and keeping track of pupils and their general progress.
- Organizing pupils and their teachers so that they consistently come together at the same place and time.
- Finding the right courses and exams for each pupil.
- Making sure that the right individuals were entered for each exam.
- Receiving and recording the exam results and
- Finding a uniform way to describe each student to the outside world so that they were not merely represented by their exam results.

Isdale (1996) says that a School Information Management System (SMS) is an "integrated computerised administration package designed to automate many of the tasks common to schools... [SMS] groups the administrative functions of the school in an organised and logical fashion..."

Isdale further opines the SMS should be organised into a series of modules which have the following functions:

- The school module records information relating to the school and its facilities and includes details of school's address, bus routes, school calendar, policies, time-tables for students and staff, asset management, books and equipment lending and records of work experience.
- The student's module records student transfer information, emergency contacts, personal details (including photograph), academic results, course information, absentee records and class locations.

- The staff module contains personal and employment information on all staff including ancillary and part-time workers. Other functions include emergency contacts, accident reports, teaching subjects, substitute lessons and absence and relief details.
- The subject module interfaces with the timetable and contains information about all subjects, classes and timetabling issues.
- The finance module manages all accounting functions within the school through a Chart of Accounts which is standard to all schools throughout Queensland. The components of this module are purchasing records, incoming and outgoing payment records, cash register functions, general ledger and bank reconciliation.

Telem & Avidov (1994) state that a School Information System (SIS) is a specialized management information system (MIS) that "matches the structure, management tasks, instructional processes, and special needs of the school".

Hicks (1990) as cited in Telem & Buvitski (1995) states that like the traditional MIS, a SIS integrates data from multiple sources to provide the information people need to make important management decisions.

Telem (1996) opines that a comprehensive SIS manages a school or district's key functional data including, but not limited to, enrollment, student and staff demographics, course enrollments, class schedules, attendance, disciplinary actions, special programs, grades, standardized assessments, and health information. Less comprehensive systems address only some of these areas. Any number of

customizable reports may be generated from the information a system stores. Telem & Buvitski (1995) further explains that most data can be exported for use in spreadsheets, word processors, databases, and statistical packages.

Ramer & Snowden (1994) as cited in Bober (2001) says that such a system integrates several components given below:

- applications: programs that provide an interface allowing users to access and manipulate computer resources
- networks: computers connected via sophisticated protocols that allow resource sharing (working in groups) and computer-mediated communications (communication with a work group); and
- operating systems: the set of instructions the computer uses to operate itself and its various input and output devices

Hedberg et al (1992) opines that if schools are going to be able to integrate data into a total information systems strategy for all data in the organisation and come up with an integrated information system for the whole organisation, the planning for this process needs to be based on a complete structure, and the educational administrators will need to have the expertise to implement such systems.

Telem and Buvitski (1995) while agreeing to the fact that the school in their study were in the initial phases far from maturation phases that will allow full data integration, they opine that even partial system implementation led to broad changes to schools technical system. It is hope that this model will show the possibility of improving school administration.

Sirotnik and Burnstein (1987) as cited in Hedberg et al (1992) suggest that at initiation phase; one should focus on the time consuming clerical activities. This model thus expects to show some positive impact on school information management.

1.2 Problem Statements

Review of the current system in student enrollment show the use of a big log book to register the new intake of students. The book is entitled "Buku Kemasukan Sekolah" literally translated as School Entry Book. The items involve are Serial Number, Admission Number, Date Entered, Name, Gender, Race, Religion, Identity Card Number, Date of Birth, Birth Certificate Number, Place of Birth, Form When Entered, Authorized Number, Guardian's Name, Relationship, Occupation, Previous School, Date Left. It is not an easy task to make entries for hundreds of new intake each year. Any teacher or administrator who had experience trying to locate a particular entry will understand the severe problem of data retrieval. This is worse if the former student came years after he has left and he couldn't furnish the year he entered the school. There were instances of repeat entries. This is mainly due to a student reenter the school after he has left for certain reasons. There were also cases of unexplained repeated entries. To sum it up, the problem statement is:

The Manual Student Enrollment System is very difficult to store and retrieve as well as causes redundancies.

The attendance roster is an important document subjected to audit by inspectorate and financial auditors. Here, the student's particulars are almost identical to those in the School Entry Book. Form teachers are required to verify the data before writing them into the roster. The ultimate data source would be the student enrollment log discussed above. Nevertheless, with the size of the book and only a single copy is available, teachers would copy them from attendance roster of previous year. Mistakes were made in copying from one book to another; from year to year. In the case of class were to be formed from different classes where not all the entries needed are from the same roster, the problem is compounded. This scenario leads to the second problem statements as follow:

The shortcomings of Manual Student Enrollment System cause problem of data integrity in attendance roster and other related documents.

1.3 Project Objective and Scope

This project is entitled "Student Enrollment System Model- A Proposal" The project objective is to identify and suggest a model to solve the problems discussed above. A prototype was developed to store students' enrollment data in order to minimize the data redundancies; to search and retrieve student records based on the National Registration Identity (NRID) or birth certificate number; and to generate attendance roster with intention to improve data integrity of students' attendance roster.

The subject area concerns school information management. To be more precise it concerns management of student data listed in student registration log book that is found in every government school. As almost the same data is needed in maintaining attendant roster. The subject areas also concerns student data management in class.

This study is confined to Malaysian government secondary school where the requirement in student enrollment is similar. The study is also restricted to data management in student enrollment and attendance roster. Other aspects of student data management such as disciplinary records management, co-curriculum registration, internal examination records, and external examination, health records etc. are not included.

2.0 METHODOLOGY

Object Oriented Analysis and Design Methodology (OOAD) was employed for this project. For modelling purposes Unified Modelling Language (UML) notations were employed.

2.1 Methodology Overview

The methodology employed in this project can be divided into six phases. Namely: (1) Requirement Capture, (2) Requirement Analysis, (3) Preliminary Design, (4) Detail Design, (5) Construction, and (6) Testing.

2.2 Requirement Capture

Requirement capture is to gain a clear understanding of existing system as well as to examine the functional requirements, non functional requirements and usability requirements of the new system. The tasks undertaken in this project involve:

(1) Fact finding and (2) documenting requirements.

2.2.1 Fact Finding

Fact finding techniques employed this project includes background reading, interviewing and document sampling. The litreture review presented earlier was based on the reading done.

In interviewing, the author has targeted three user groups; (1) the top administrator, (2) middle level administrator or called school administrator, and (3) the student users. The top administrators interviewed were the school principal, the administrative senior assistant, the senior assistant in charge of student affairs, and the afternoon supervisor. The middle level administrators interviewed were the supervisors of each form and selected form teachers. In document sampling, the Student Registration Log Book was acquired. A Class Attendance Roster was also acquired from school. Besides these, samples of school leaving certificate, testimonies, school leaving certificate forms, school enrollment form, and discipline record book were sampled.

2.2.2 Documenting Requirements.

Records of interview and copies of existing documents were maintained to extract the requirements. At this stage requirements were documented using plain English. Context use case diagrams were also used to document user requirements.

2.3 Requirement Analysis

Requirement analysis is to analyse the logical structure of the problem situation and also the way logical elements interacts with each other. Here, the tasks involve the realization of use case diagram in requirement capture. This involves the identification of possible classes, together with an understanding how these classes interact to deliver functionality of a use case. It involves the modeling of collaboration diagram and class diagram. At the end of this phase a Software Requirement Specifications (SRS) was produced. This document entitled Software Requirement Specifications for The Web-Based Student Enrollment System for Malaysian Government Secondary School (SES_SRS) is presented as attachment to this report. Though listed under appendix, they are actually an integral part of this report. It is necessary to read it to understand the proposed Student Enrollment System.

2.4 Preliminary Design

Preliminary design involves consideration of overall system architecture, the relationship among its major components and their interactions. The activities involve: (1) identifying sub-system and its major components, (2) identifying any inherent concurrency, (3) allocating sub-system to processors, (4) select a data management strategy, (5) selecting a strategy and standard for human-computer interaction, (6) specify code development standards, (7) plan the control aspects of the application, (8) produce test plan, and (9) set priority for design trade-off.

2.4.1 Identifying Sub-System and Its Major Component

In identifying sub-system and its major components, consideration was made about the number of tiers to be employed in the whole architecture. As the model was intended to be web-based, it was the decided that the applications can be model using three layers. The layers are presentation, application logic, and storage layers. The presentation layer is the set of html pages with which the user request services from the application. This layer maps the presentation entities within the html pages onto application logic layer services. The application layer contains the logic, " which translates user requests into data manipulation and presentation tasks. These tasks include the reformatting of data from the storage layer into an appropriate form for the presentation layer and the mapping of service requests from the presentation layer into a set of tasks for the storage layer. The storage layer is used to connect to data entities with a data-store or database. This layer translates data manipulation tasks into database tasks.

2.4.2 Identifying Any Inherent Concurrency

As the suggested model is a web-based application, it is inevitable that concurrency problem would occur. A concurrency control has to be decided at this preliminary design phase. It was decided that the model would employed Optimistic with Row Versioning type of concurrency control. This optimistic concurrency control option is based on row versioning. With row versioning, the underlying table must have a version identifier of some type that the server can use to determine whether the row has been changed after it was read into the cursor. In SQL Server that capability is provided by the timestamp data type, which is a binary number that indicates the relative sequence of modifications in a database. Each database has a global current

timestamp value (DBTS). Each time a row with a timestamp column is modified in any way; SQL Server stores the current DBTS value in the timestamp column and then increments DBTS. If a table has a timestamp column, then the timestamps are taken down to the row level. The server can then compare the current timestamp value of a row with the timestamp value that was stored when the row was last fetched to determine whether the row has been updated.

2.4.3 Allocating Sub-System to Processors,

In allocating sub-system to processors, the overall system architecture is considered. This project suggests the use of logical three-tiered architecture. Physically, however, two-tiered architecture is employed. The presentation layer is allocated to the client computer; the application layer and storage layer assigned to server. The model thus suggests a client/server system using TCP/IP to connect client with server. This architecture is shown in the deployment diagram below.

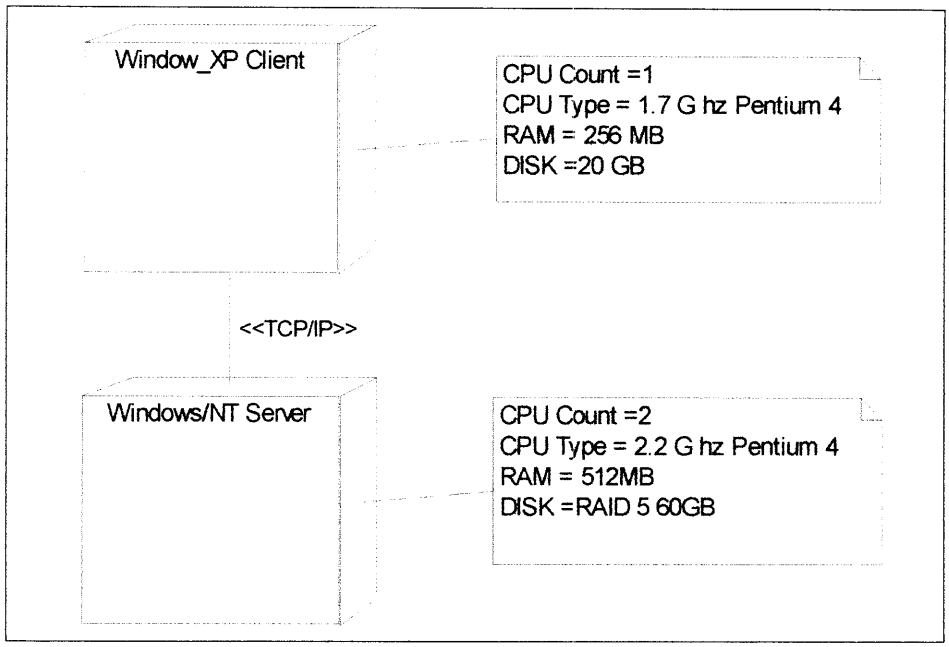


Figure 2.1 Deployment Diagram for SES System.

2.4.4 Select a Data Management Strategy

At this phase decision has to be made about strategy to store and retrieve persistant data. Is the data to be stored in files or DBMS? If DBMS were to be used, which DBMS would be employed? In this project a relational database management system is used. To be more precise the Microsoft SQL Server 2000 is choosen. It is a Structured Query Language (SQL)–based, scalable, relational database with integrated Extensible Markup Language (XML) support for Internet applications.

2.4.5 Selecting a Strategy and Standard for Human-Computer Interaction

In deciding the strategy of the Human-Computer Interation (HCI), a few considerations were made, namely:

1. The nature of the task that the user is carrying out.
2. The type of user.
3. The amount of training that the user will have undertaken.
4. The frequency of use
5. The hardware and software architecture of the system.

As most users here are familiar with Microsofts Window Interface, this model proposes the use of the direct manipulation metaphor which is event driven and similar to that of Microsoft's GUI. In fixing the standard, four criteria were considered. They are (1) consistency, (2) appropriate user support, (3) adequate feedback from the system and, (4) minimal user input.

2.4.6 Specify Code Development Standards

A typical object-oriented standard is adopted. The naming of classes, attributes, operations and other elements of the system were determined. In naming the classes, classes are named with an initial capital letter. Words are concatenated together when the class name is longer than one word. However, in this model entity classes were named in all capital as not to confuse CLASS (as in classroom) with class in OOAD. Attributes were named with an initial lower case letter. The same concatenation approach is taken for attributes with more than one word. For operations, the naming standard is as that of attributes.

2.4.7 Plan the Control Aspects of the Application

Although the specifying of control is done in detail design, the planning of it should have been incorporated into sequence diagrams and state diagrams at the Requirement Analysis phase. At this juncture the control aspects of the applications were reviewed.

2.4.8 Produce Test Plan

To test the model a prototype was developed. This prototyped was named "A Web-based Student Enrollment System for Malaysian Government Secondary School". Test plan was done to test the functionality of this prototype. However, only black box testing was planned to ensure the completeness and correctness of the prototype. This is due to the fact that the prototype was meant to test the working of the model. In another word the plan was to see if the prototype does what it is meant to do. This test plan and its resulus would be produced in the findings sections.

2.4.9 Set Priority For Design Trade-Off.

No specific guidelines for design trade-off were laid down for this project. However there was one major design trade-off implemented. The initial user requirement was to print the student records and class list on line. But due to difficulty in formating, it was decided that it would be done locally from a standalone sited at a specific location.

2.5 Detail Design

The detail design also known as object design in OOAD concerns the detail design of objects and their interactions. The tasks involve the specification of object attribute types, specification of parameters pass and return, and specification of associations between classes. Construction or coding of the prototype commences after detail design and black box testing would be employed to test if the prototype does what the users require. The final deliverable of this phase is the Software Design Document (SDD). The SDD for this project is entitled Software Design Document for the Web-based Student Enrollment System for Malaysian Secondary School (SES_SDD). It is listed under appendixes but in effect is an integral part of project report in order to understand the proposed model.

3.0 FINDINGS AND DISCUSSIONS

3.1 Findings

The findings of this project is given below.

3.1.1 Finding Overview

At the Requirement Capture phase, the below requirements were recorded.

User Group	Requirements
The System administrators	1. To activate and deactivate users. 2. To change password. 3. To view, add, edit and delete student records. 4. To create class and assign students to it. 5. To print student record by intake year, class, and individually. 6. To print class name list.
The School administrators	1. To change password. 2. To view, add, edit and delete student records. 3. To create class and assign students to it. 4. To print student record by class and individually. 5. To print class name list.
The Students	1. To view own personal record. 2. To change password. 3. To update record.

Table 3.1: User requirements of Student Enrollment System.

Further consultation with top management and consideration of project objectives give rise to required system function shown in context use case diagram in Figure 1 below:

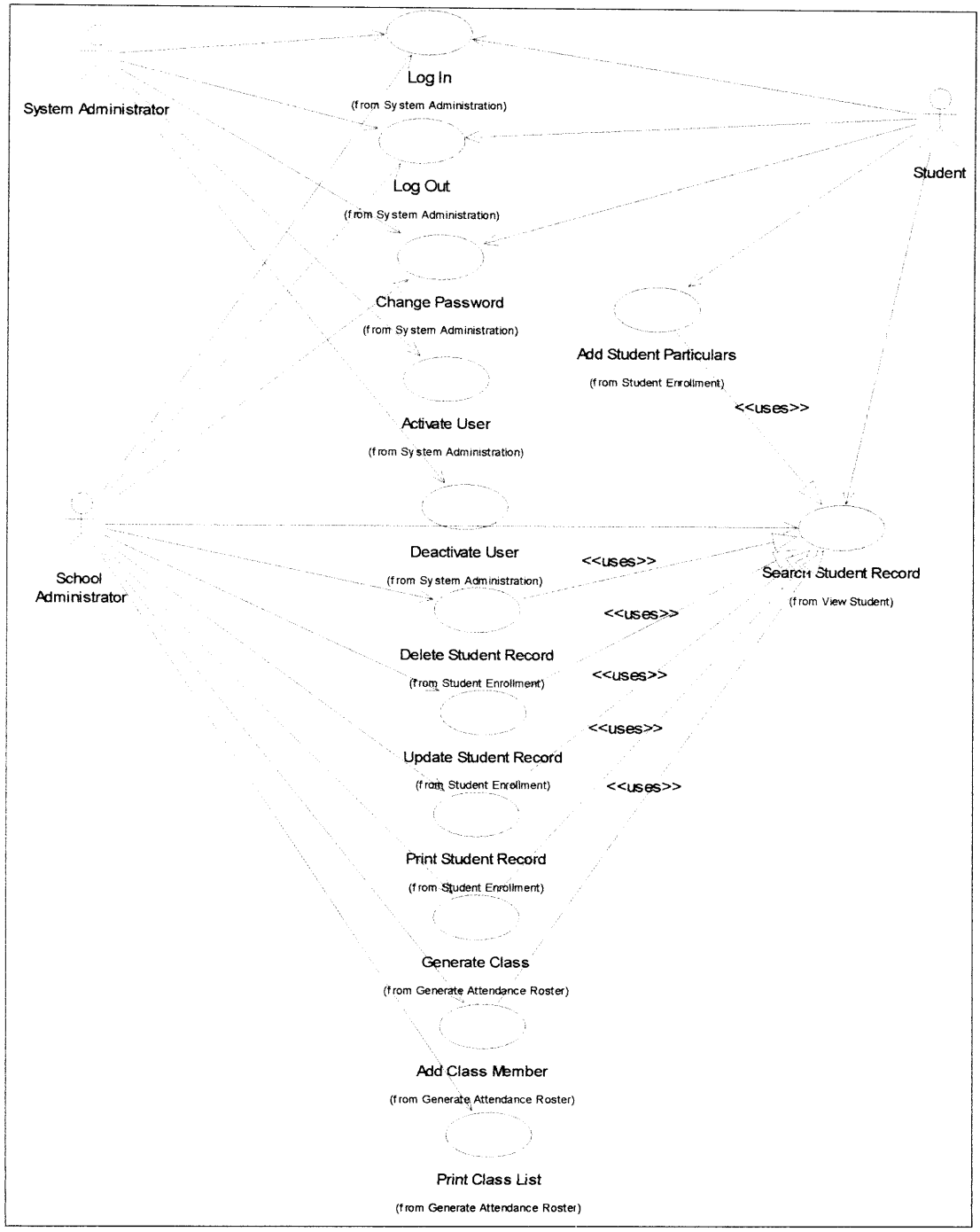


Figure 3.1: Overall functions of Student Enrollment System

From here, the requirement analysis commences and gives rise to first cut model in detail. This model is presented in Software Requirement Specification entitled SES_SRS. This model consists of (1) Context Use Case Diagram, (2) Use Case

Package Diagram for SES, (3) Use Case Diagrams for Sub-Systems, (4) Use Case Specifications (5) Class Diagram Package for Use Case Sub-Systems, (6) Sequence Diagrams, (7) State Diagrams, (8) Class Diagrams.

The Preliminary and Detail Design phases further developed this model into a model that consists of: (1) System Architecture Diagram, (2) Data Specifications, (3) Class Diagrams, (4) Component Diagrams, (5) Deployment Diagram, (6) Sequence diagrams, (7) Collaboration Diagrams, and (8) State Diagrams.

This model is thus presented in detail as SES_SRS and SES_SDD attached. Due to page constraint only selected diagrams and tables would be presented here to give a general view of the model. To have complete view of the model please refer to the abovementioned documents attached.

3.1.2 Model Showing System Capability and Division of Roles.

Below is the use case package diagram for Student Enrollment System. Like the context use case diagram, it shows the system overview but has been grouped to show sub-systems. Take note that System Administrator does not involve in other packages other than System Administration.

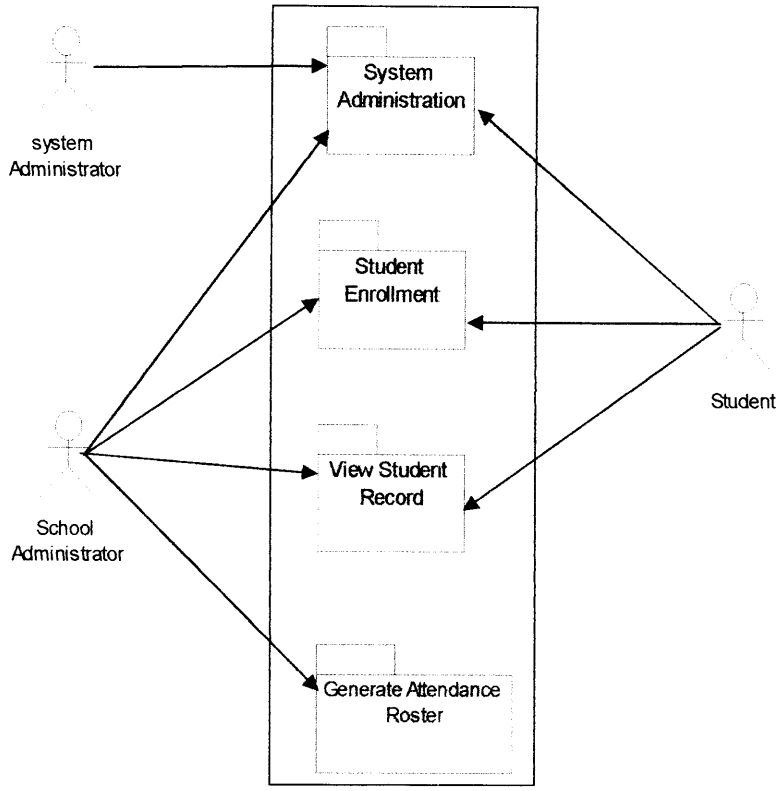


Figure 3.2 Use Case Package Diagram for SES

Below are the use case diagram and its descriptions for System Administration Use Case Package. The use cases give an overall view of the package. Take note that System Administrator Activate and Deactivate other system users. Also take note that all system users can change password.

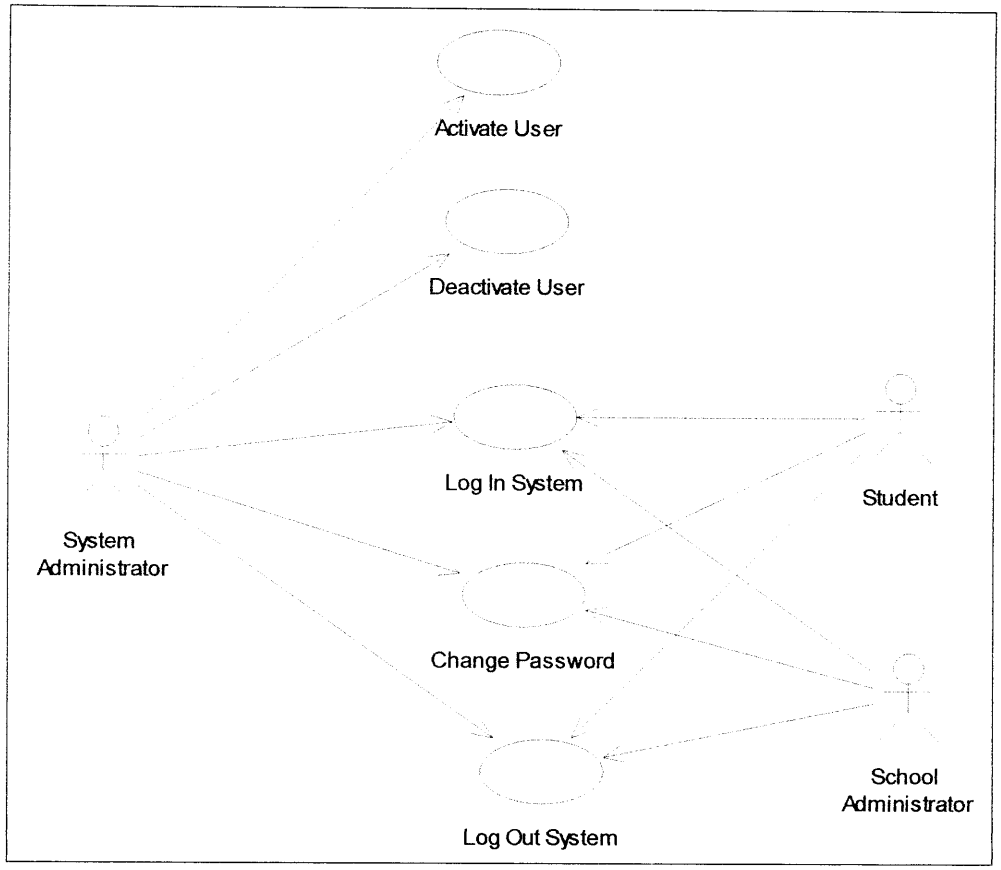


Figure 3.3 Use Case Diagrams for System Administration

Use case	**Description**
Activate User [SRS_SES_100]	The System Administrator log in system to activate the User by input user name and user id.
Deactivate User [SRS_SES_200]	The System Administrator log in to deactivate user that are longer having any linkage with the school.
Login System [SRS_SES_300]	The System Administrator, the School Administrator, and the Student log into their web site when they input their password and user name.
Change Password [SRS_SES_400]	For security reason, each user can change their password which in default was the NRID or Birth Certificate Number.
Logout System [SRS_SES_500]	The System Administrator, the School Administrator, and the Student log out of the web site after they have finish using the web site. This will exit the user from the system.

Table 3.2: Description Use Case System Administration

Below are the use case diagram and its descriptions for Student Enrollment Use Case Package. The use cases give an overall view of the package. Take note that Student is responsible to add own particulars but school administrator can update and delete student record.

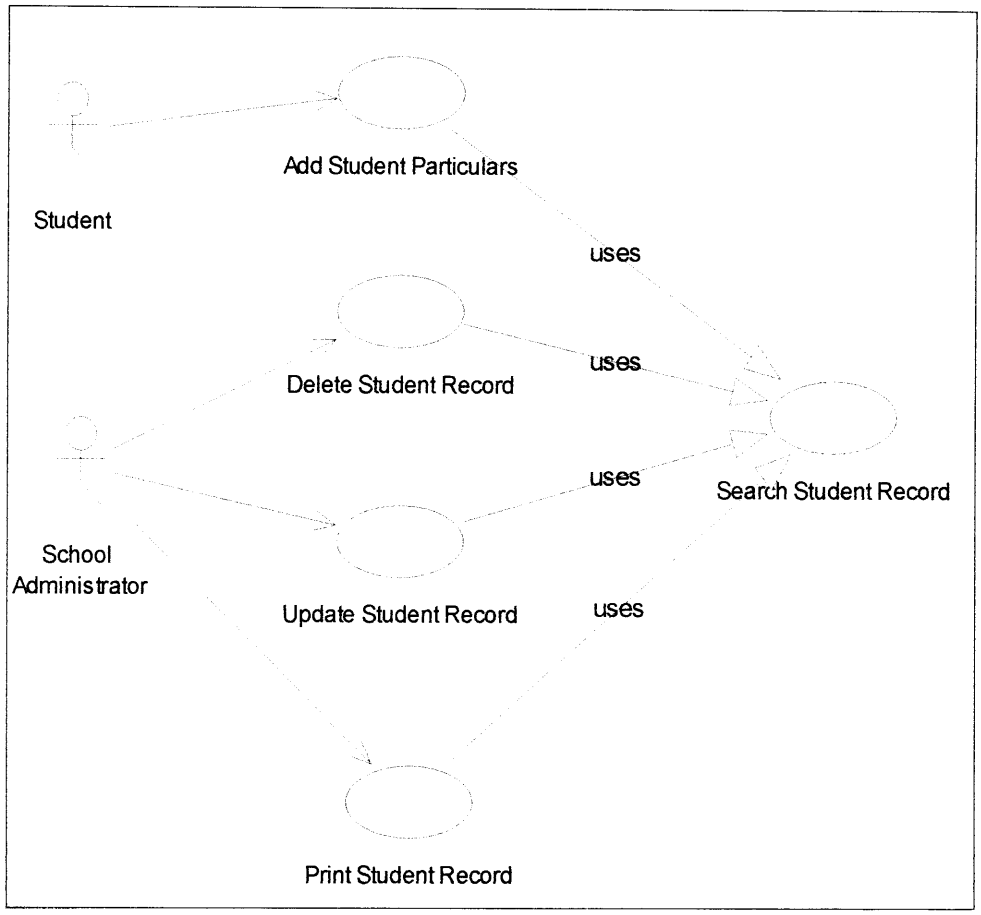


Figure 3.4 Use Case Diagram for Student Enrollment Use Case Package.

Use Case	Description
Add Student Particulars [SRS_SES_600]	New Student enters their personal particulars and saved as a new record.
Delete Student Record [SRS_SES_700]	When a new student fail to report physically after entering his/her particulars, the School Administrator may delete his/her record completely in the database.
Update Student Record [SRS_SES_800]	There is a need to edit and update a Student's record if there are changes in their particulars such as changes of address, phone number etc.
Print Student Record [SRS_SES_900]	The School Administrator may of various reason wish to print student record. The School Administrator can choose to print individual record, by intake or by class.

Table 3.3: Description Use Case Student Enrollment

Below are the use case diagram and its descriptions for View Student Record Use Case Package. The use cases give an overall view of the package.

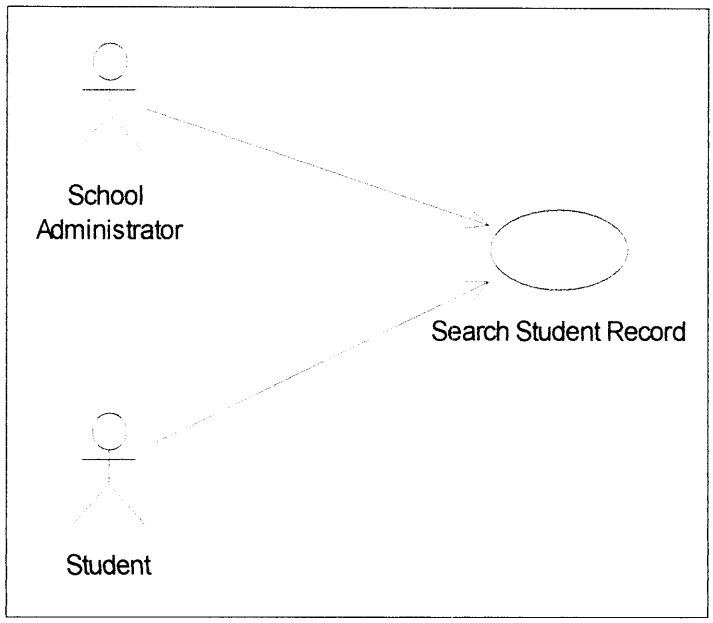


Figure 3.5 Use Case Diagram for Search Student Record Use Case Package.

Use Case	Description
Search Student Record [SRS_SES_1000]	When the School Administrator and students needs to view the Student Record, they may search the record by enter Student NRID or Birth Certificate Number. If the record exists, system will display the record on screen or otherwise shows error message.

Table 3.4: Description Use Case Search Student Record

Below are the use case diagram and descriptions for Generate Attendance Roster Use Case Package. The use cases give an overall view of the package.

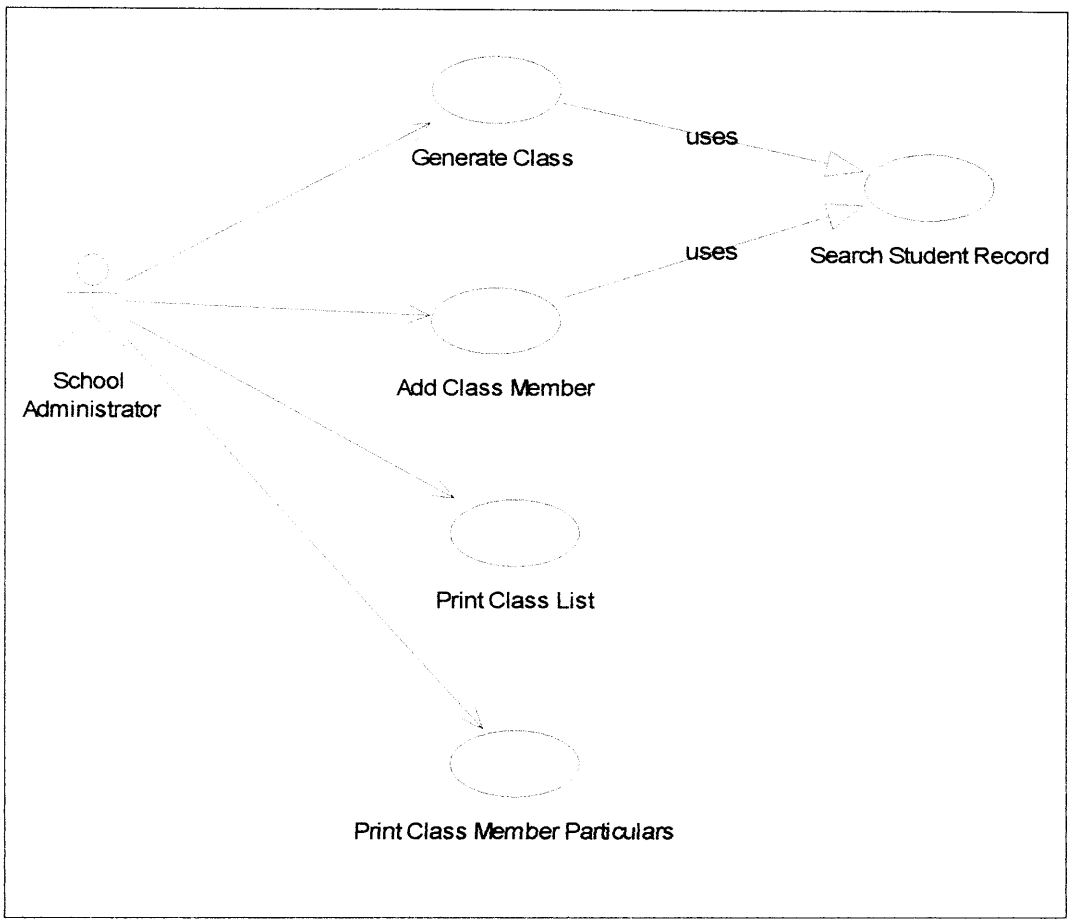


Figure 3.6 Use Case Diagram for Generate Attendance Roster Use Case Package.

Use Case	Description
Generate Class Member [SRS_SES_1100]	School Administrator includes class members to respective class.
Add Class Member [SRS_SES_1200]	School Administrator add new class member to class after the class member has been initially generated.
Print Name List [SRS_SES_1300]	School Administrator print class member list.
Print Member Particulars [SRS_SES_1400]	School Administrator prints the particulars of student in an academic class.

Table 3.5: Description Use Case Generate Attendance Roster.

This model has shown the possibilty of sharing workload and responsibilities among the top administrator, middle level administrators and students. This is achieved by allocating activation and deactivation role to system administrator; allocating school and class administration roles to school administrators and role of inputing own personal particulars to the students. Besides this the model also demonstrates the allocation to safeguard privacy by allowing users of all levels to change their passwords. To further investigate the seperations of roles and responsibilities while allowing storing of students' enrollment data in order to minimize the data redundancies; to search and retrieve student records based on the National Registration Identity (NRID) or birth certificate number; and to generate attendance roster with intention to improve data integrity of students' attendance roster, please study the Design Phase sequence diagrams below.

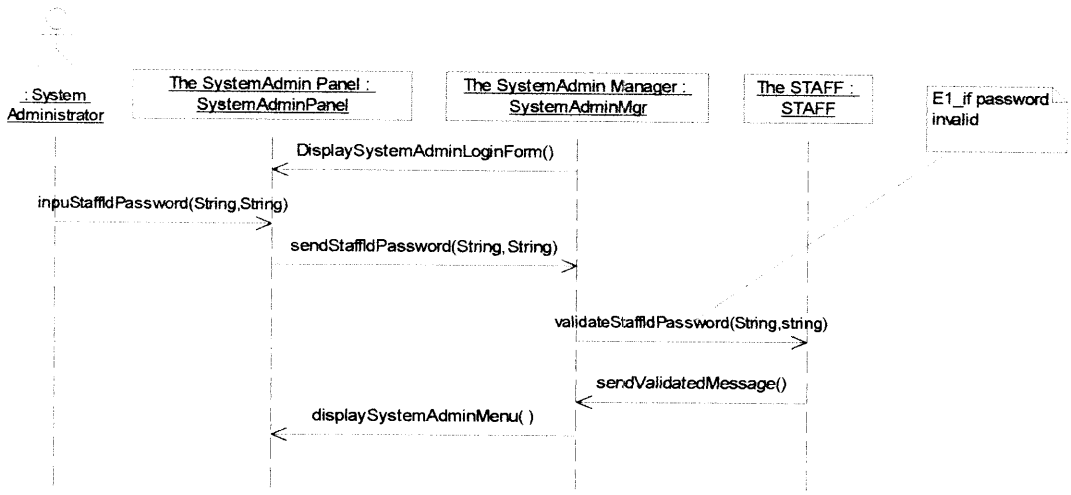


Figure 3.7 Sequence Diagram System Administrator Login

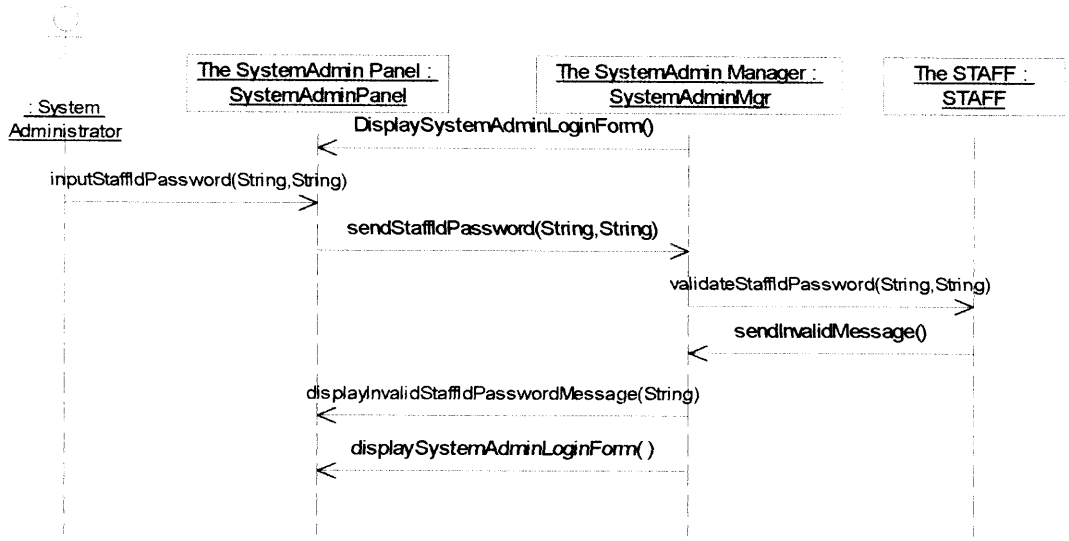


Figure 3.8 Sequence Diagram System Administrator Login/E1_if password invalid.

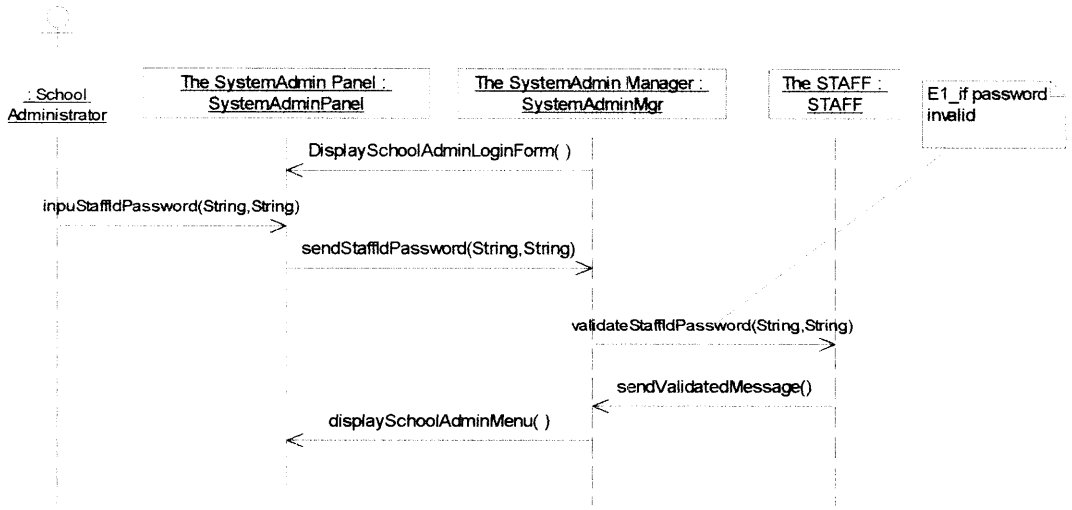


Figure 3.9 Sequence Diagram School Administrator Login

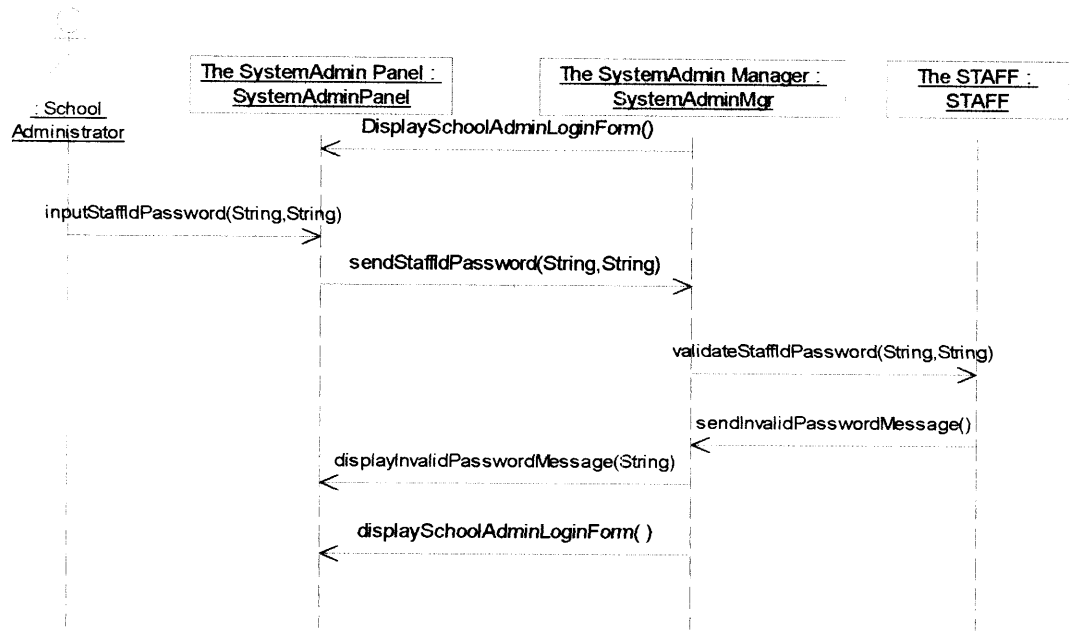


Figure 3.10 Sequence Diagram School Administrator Login/E1_if password invalid

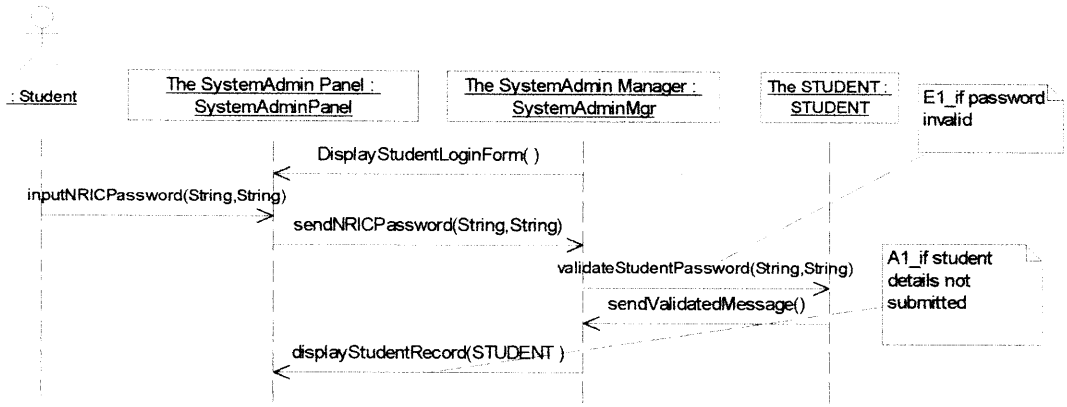


Figure 3.11 Sequence Diagram Student Login

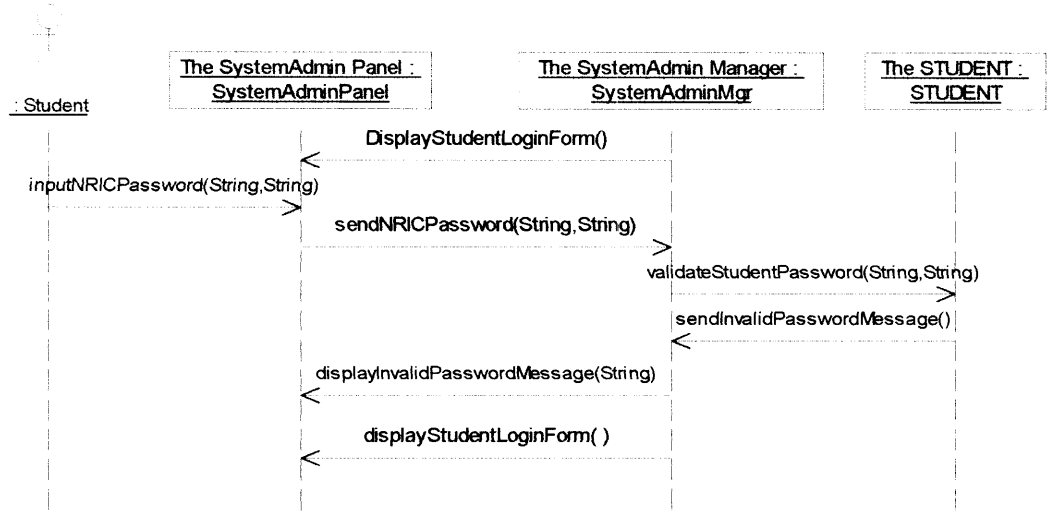


Figure 3.12 Sequence Diagram Student Login/E1_if password invalid

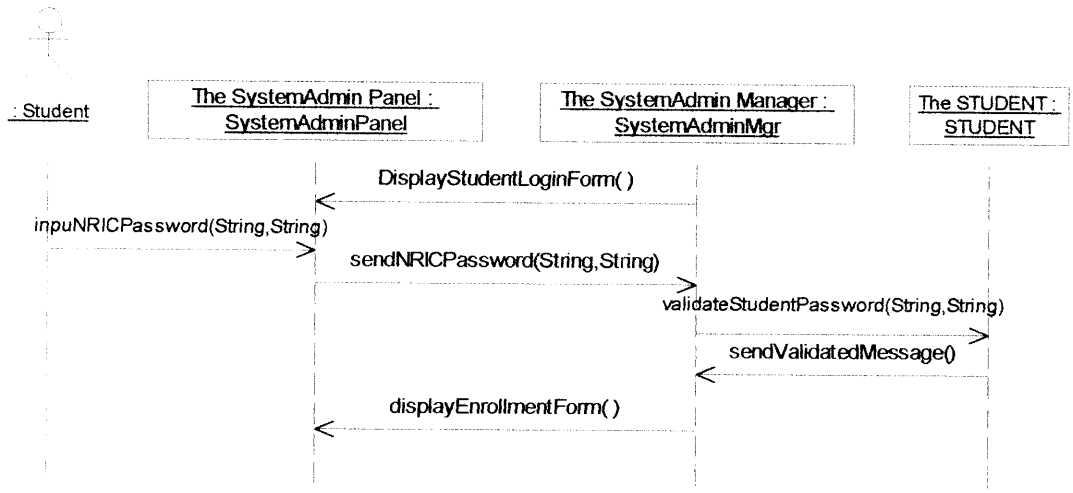


Figure 3.13 Sequence Diagram Student Login/A1_if student particulars not submitted

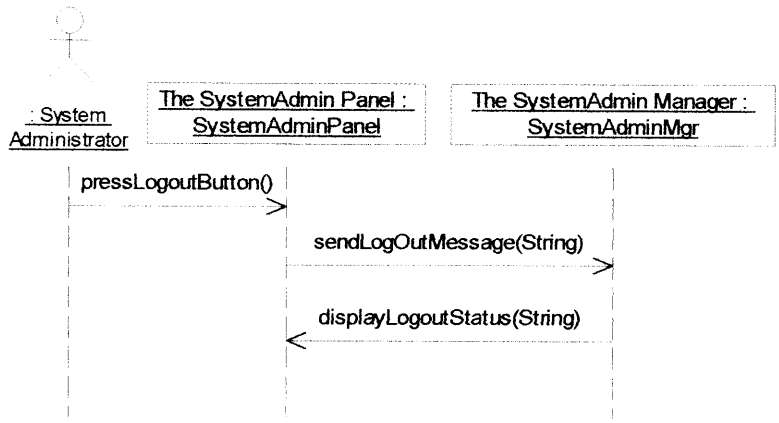


Figure 3.14 Sequence Diagram System Administrator Logout

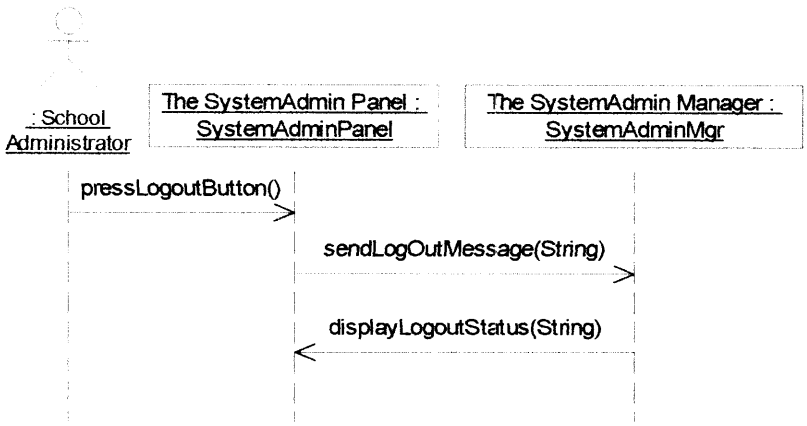


Figure 3.15 Sequence Diagram School Administrator Logout

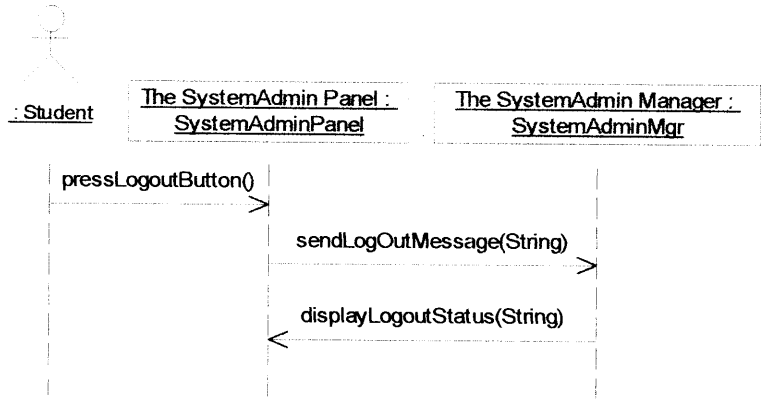


Figure 3.16 Sequence Diagram Student Logout

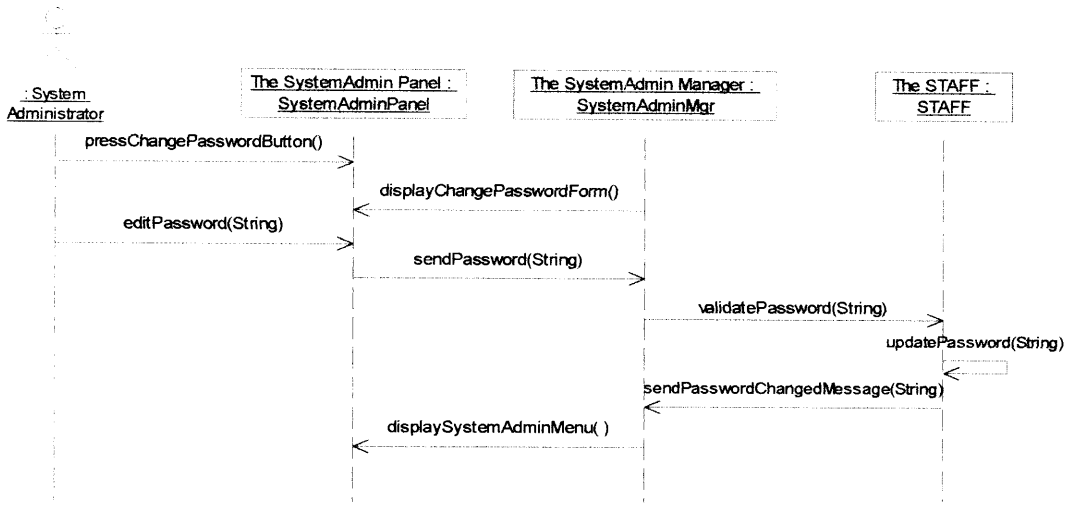


Figure 3.17 Sequence Diagram System Administrator Change Password

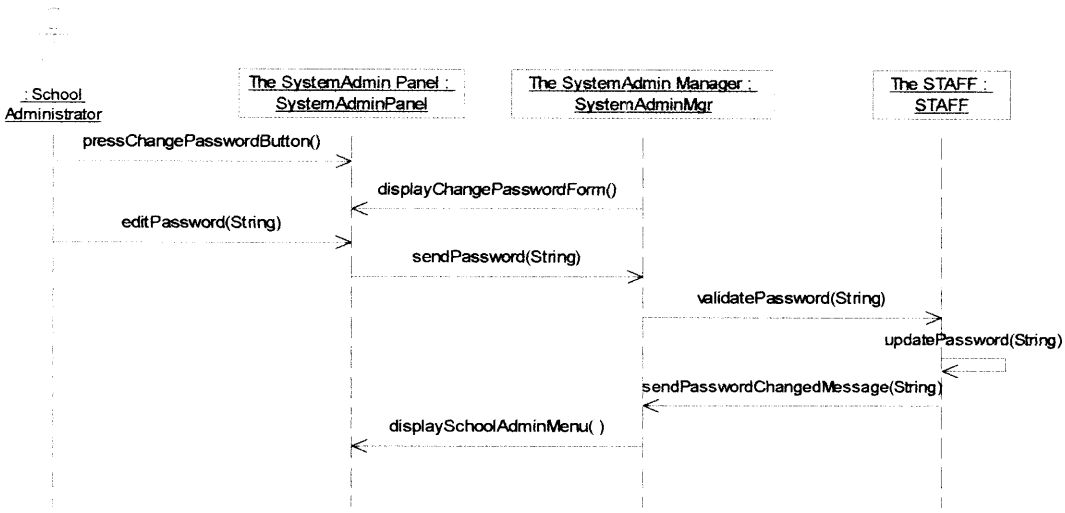


Figure 3.18 Sequence Diagram School Administrator Change Password

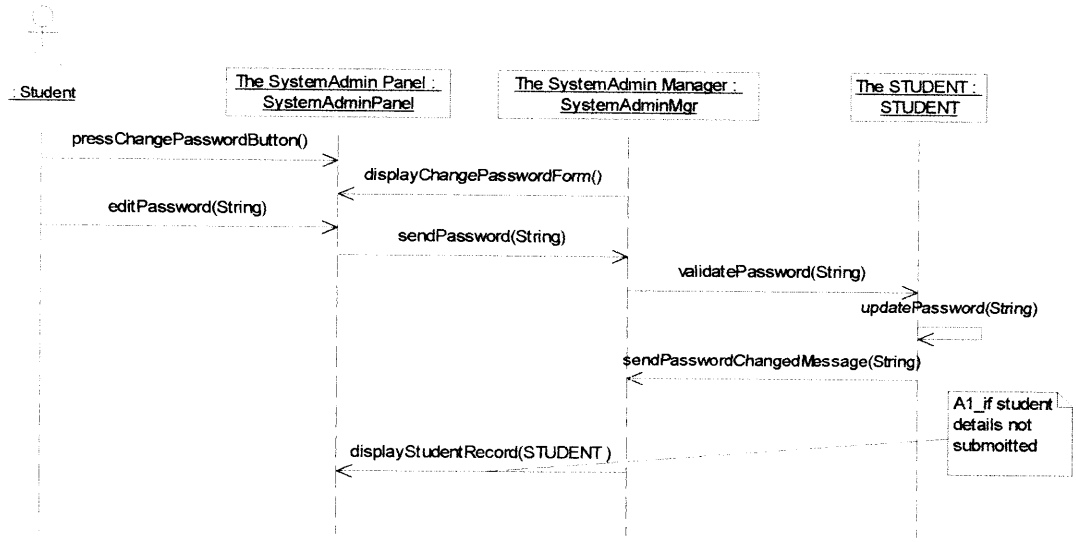


Figure 3.19 Sequence Diagram Student Change Password

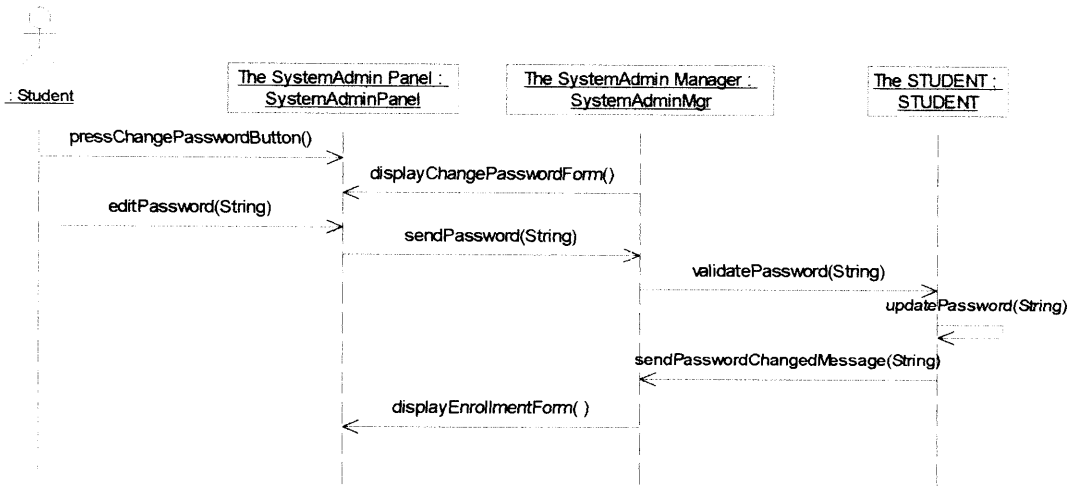


Figure 3.20 Sequence Diagram Student Change Password/A1_if student particulars not Submitted

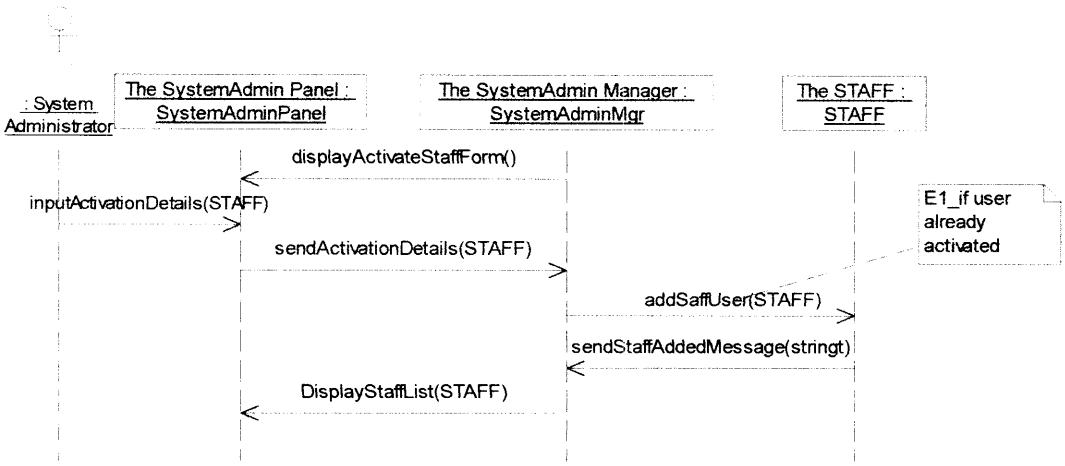


Figure 3.21 Sequence Diagram Activate Staff

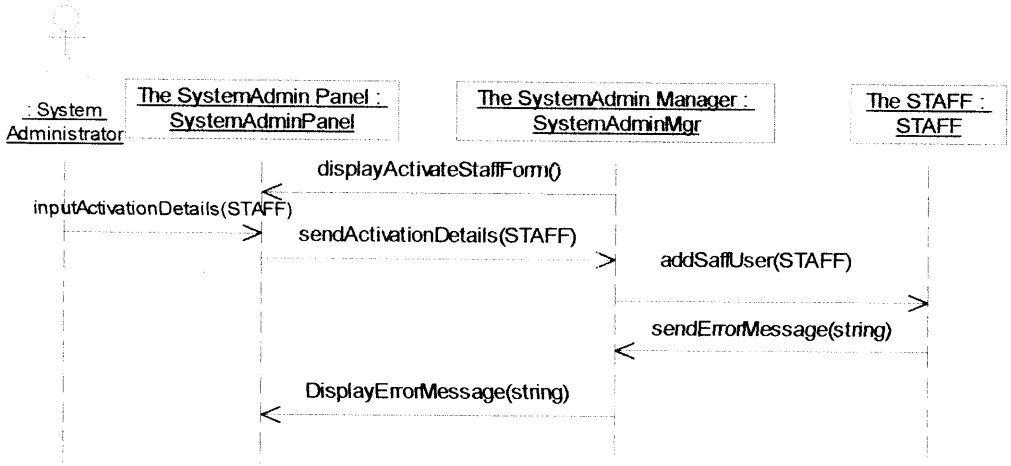


Figure 3.22 Sequence Diagram Activate Staff/E1_if user already activate

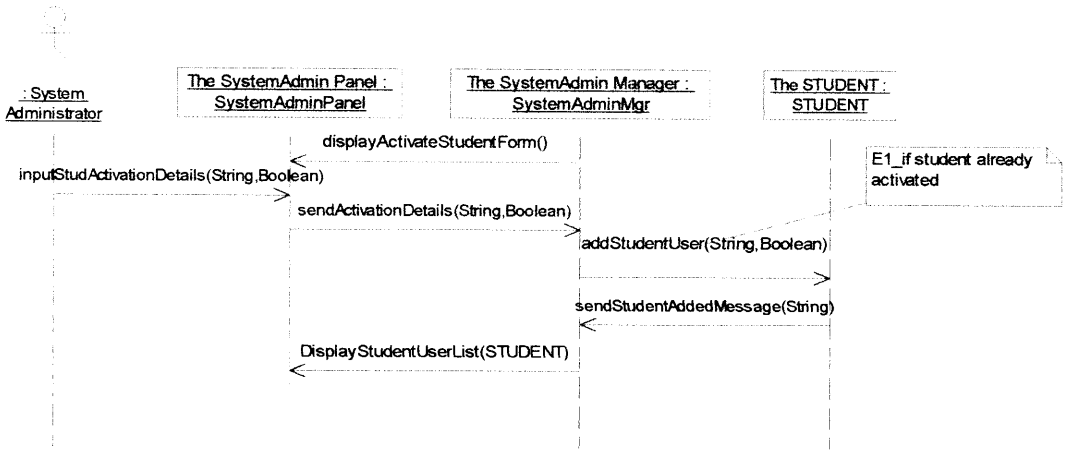


Figure 3.23 Sequence Diagram Activate Student

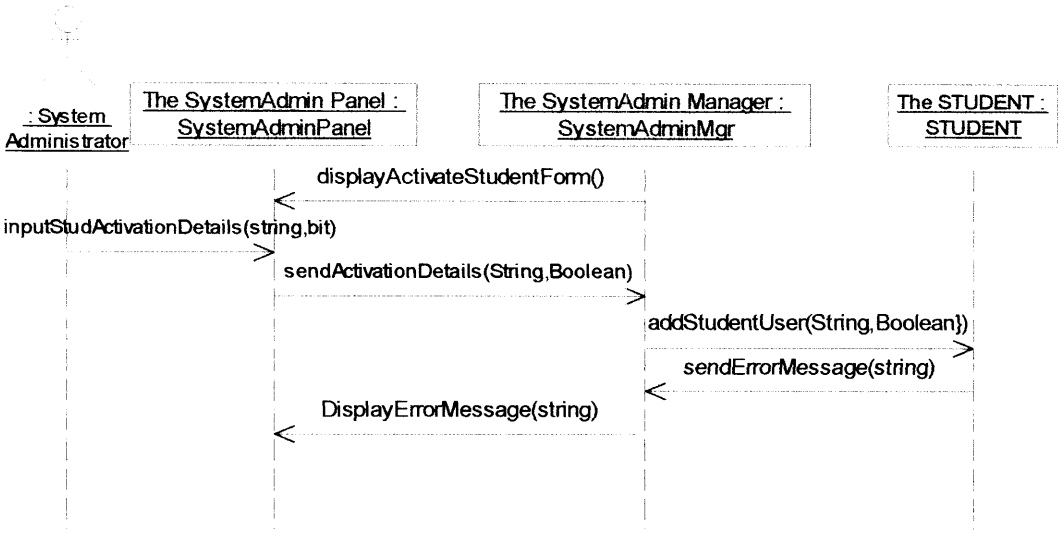


Figure 3.24 Sequence Diagram Activate Student/E1_if student already activated

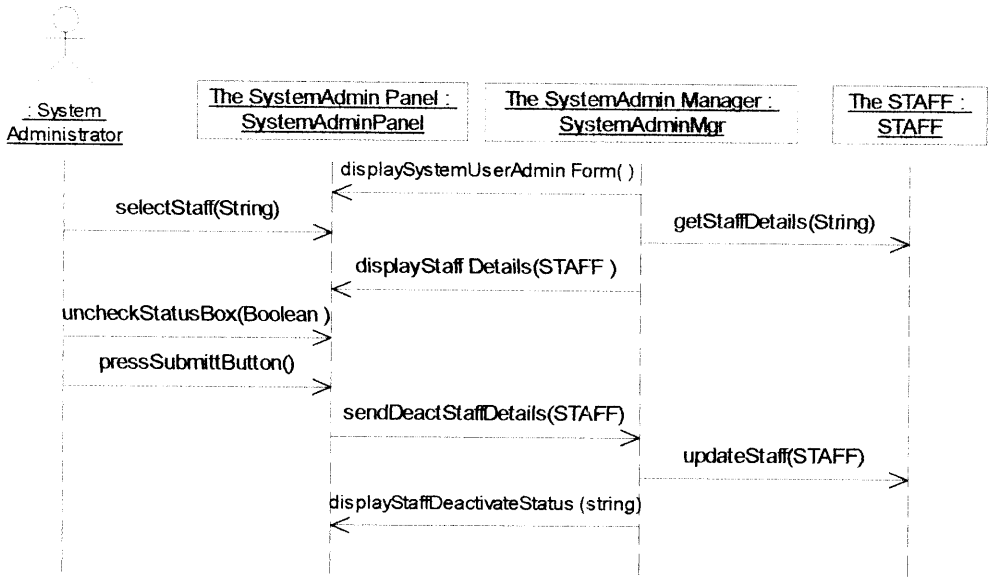


Figure 3.25 Sequence Diagram Deactivate Staff

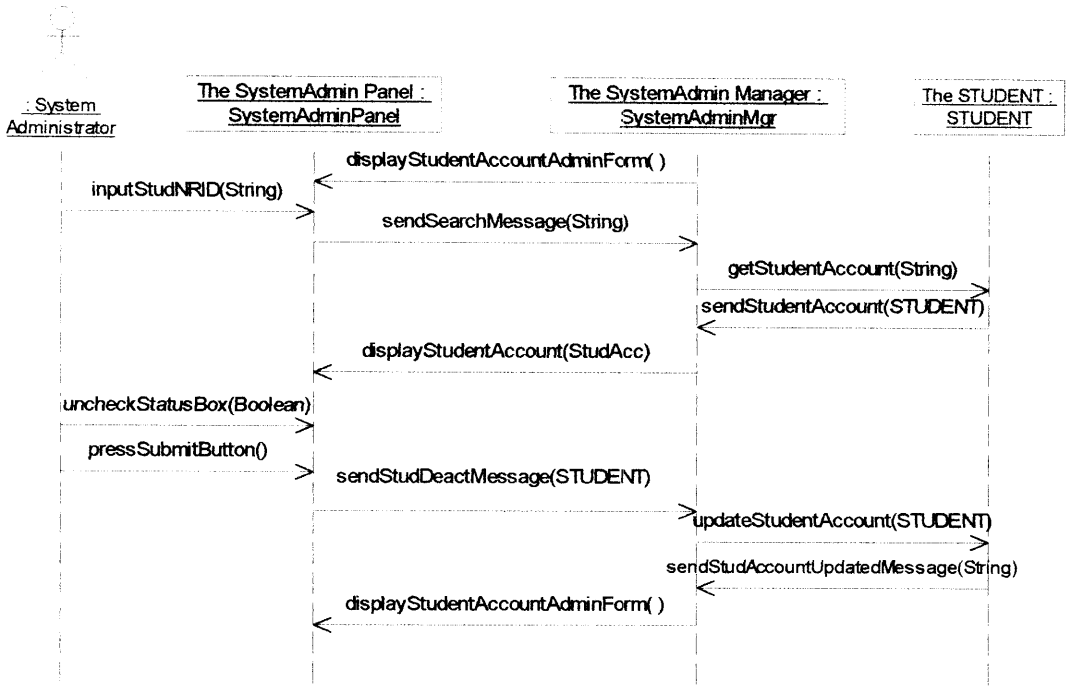


Figure 3.26 Sequence Diagram Deactivate Student

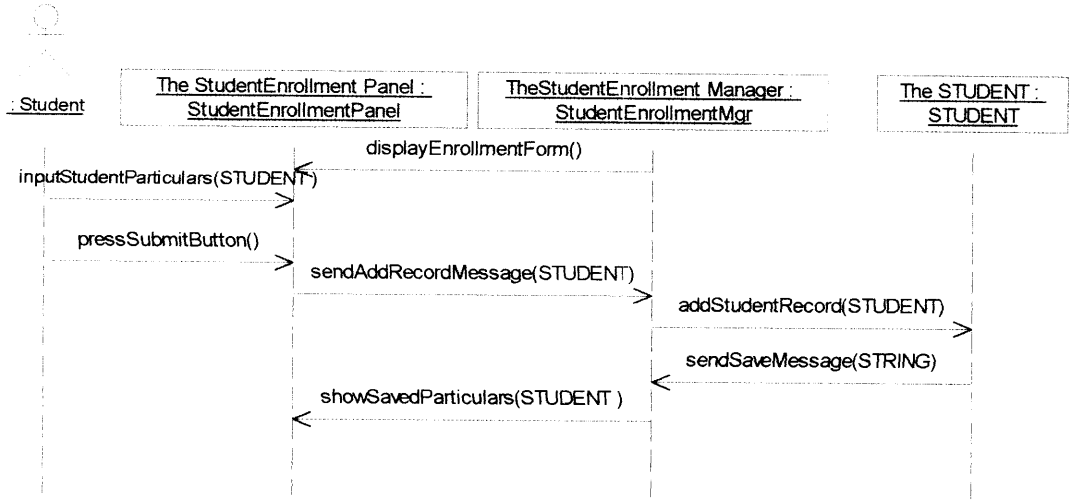


Figure 3.27 Sequence Diagram Add Student Particulars

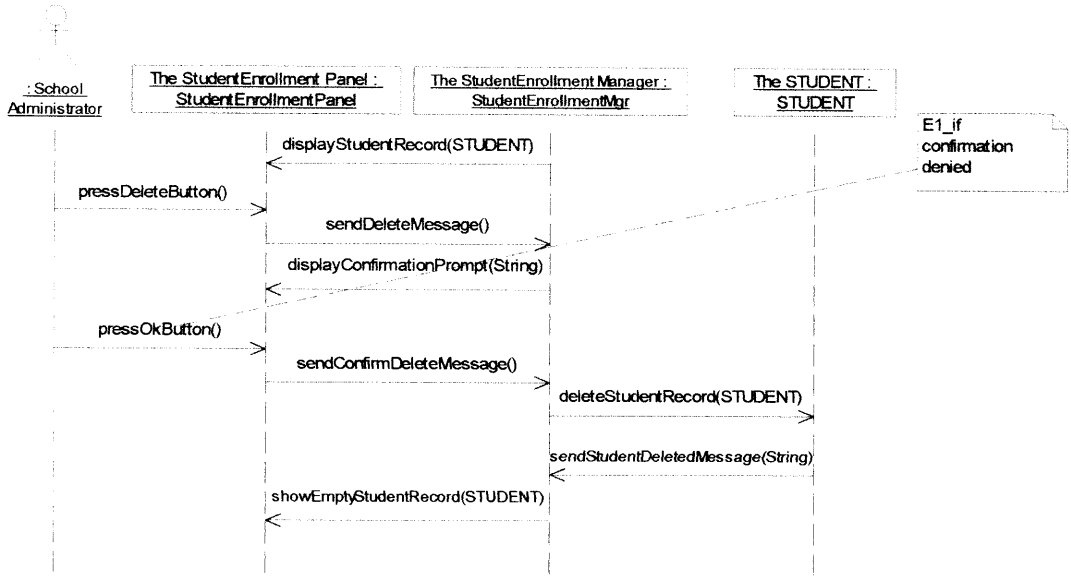


Figure 3.28 Sequence Diagram Delete Student Record

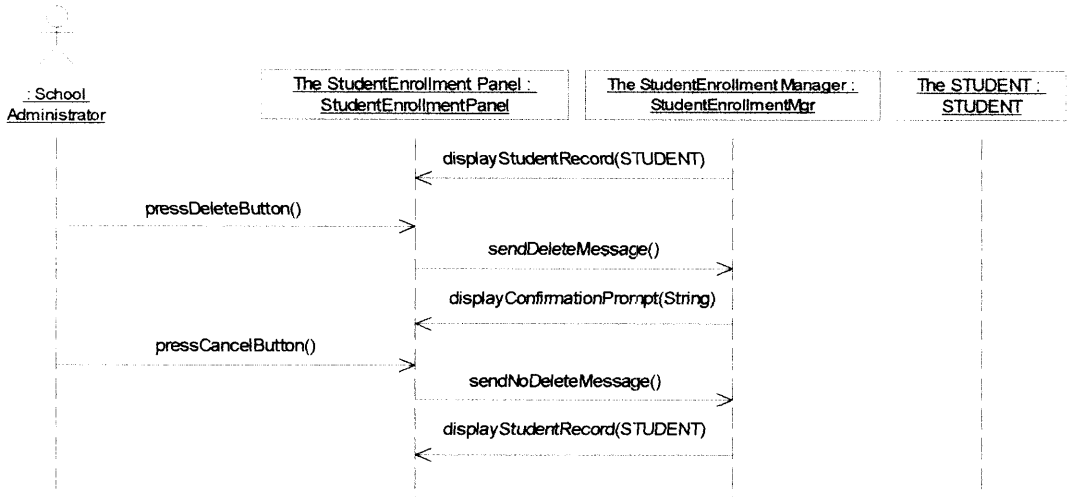


Figure 3.29 Sequence Diagram Delete Student Record/E1_if confirmation denied

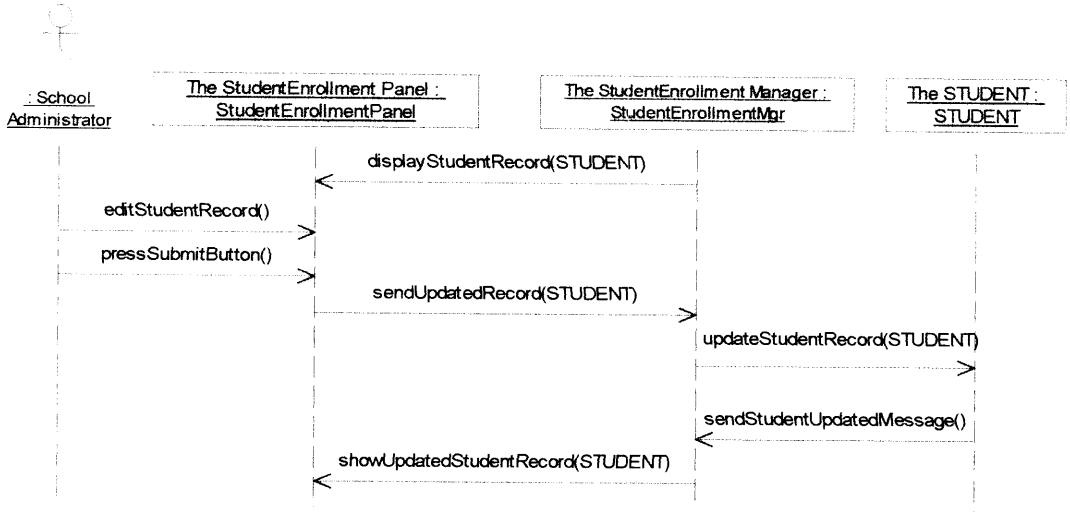


Figure 3.30 Sequence Diagram Update Student Record

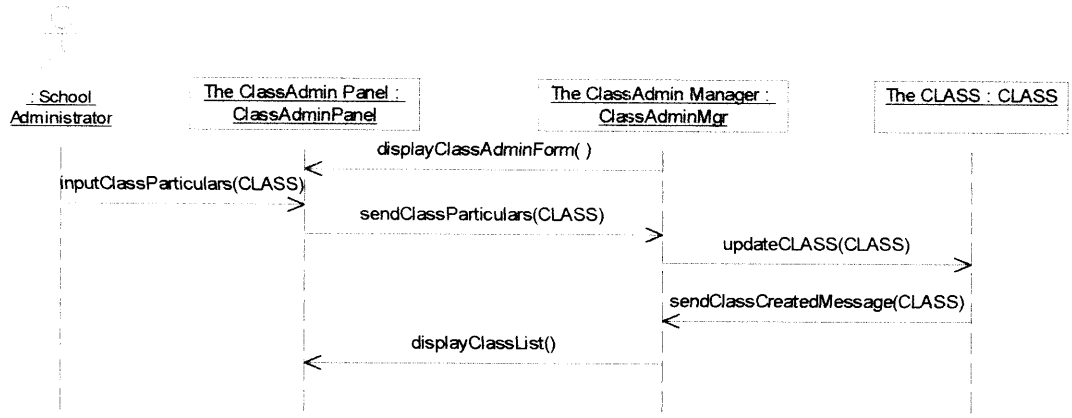


Figure 3.31 Sequence Diagram Generate Class

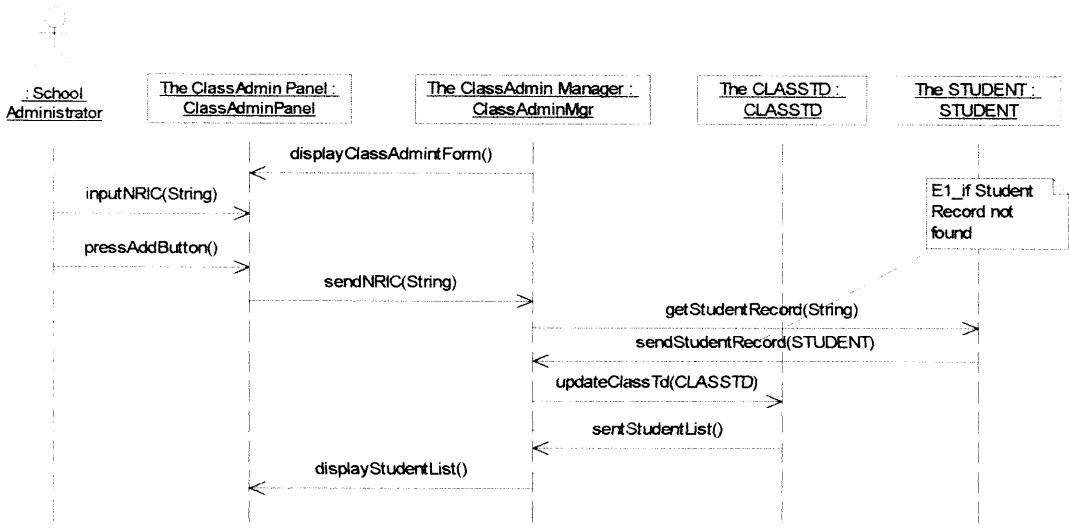


Figure 3.32 Sequence Diagram Add Class Member

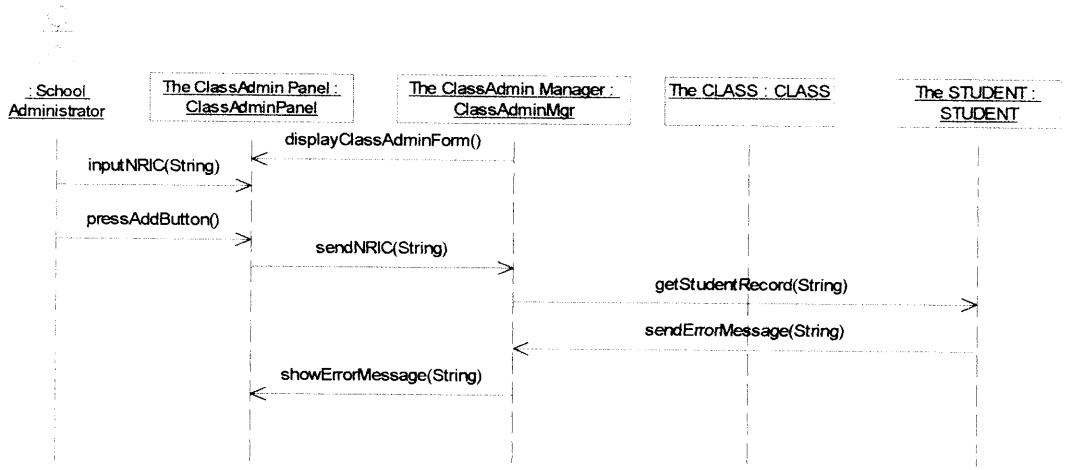


Figure 3.33 Sequence Diagram Add Class Member/E1_if student record not found

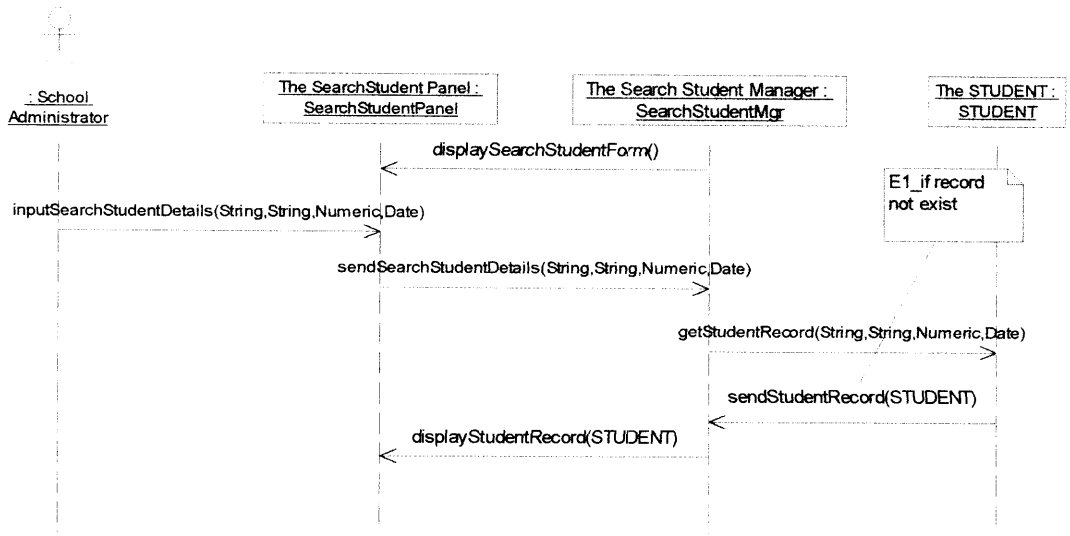


Figure 3.34 Sequence Diagram Search Student Record

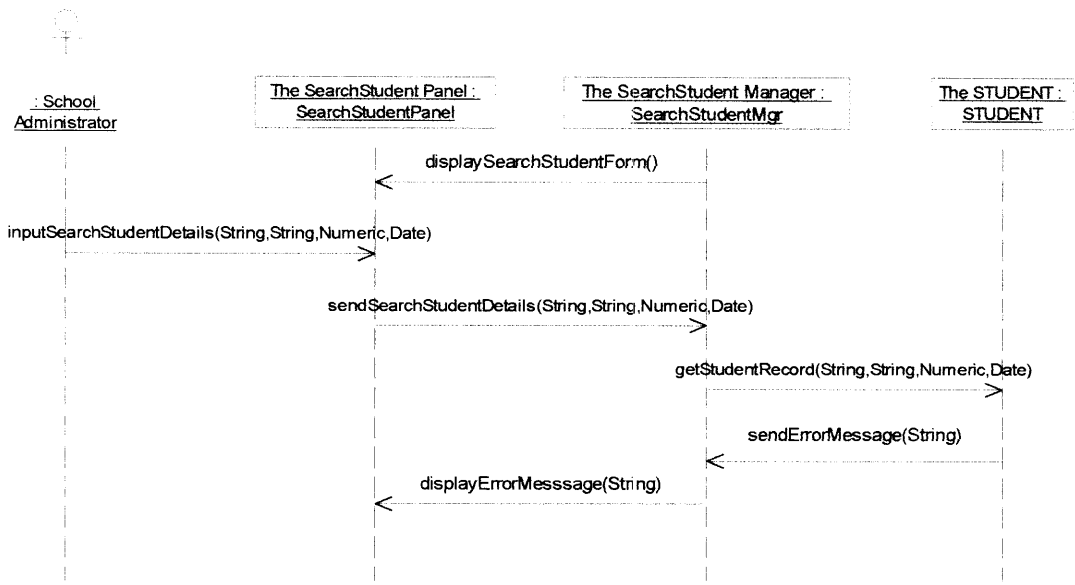


Figure 3.35 Sequence Diagram Search Student Record/E1_if student record not found.

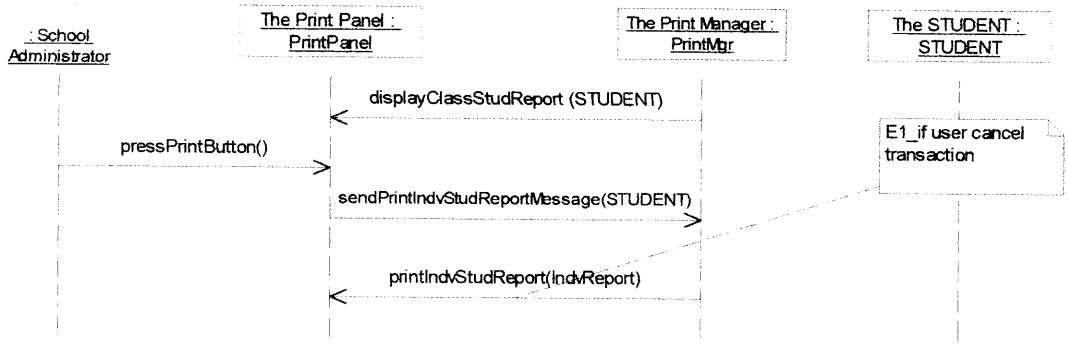


Figure 3.36 Sequence Diagram Print Individual Student Record

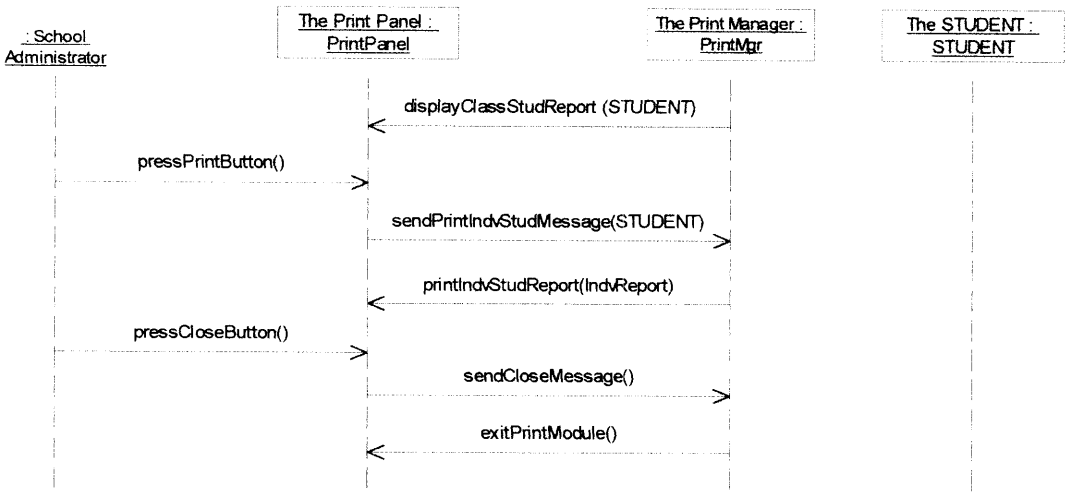


Figure 3.37 Sequence Diagram Print Individual Student Record/E1_if user cancels transacrtion

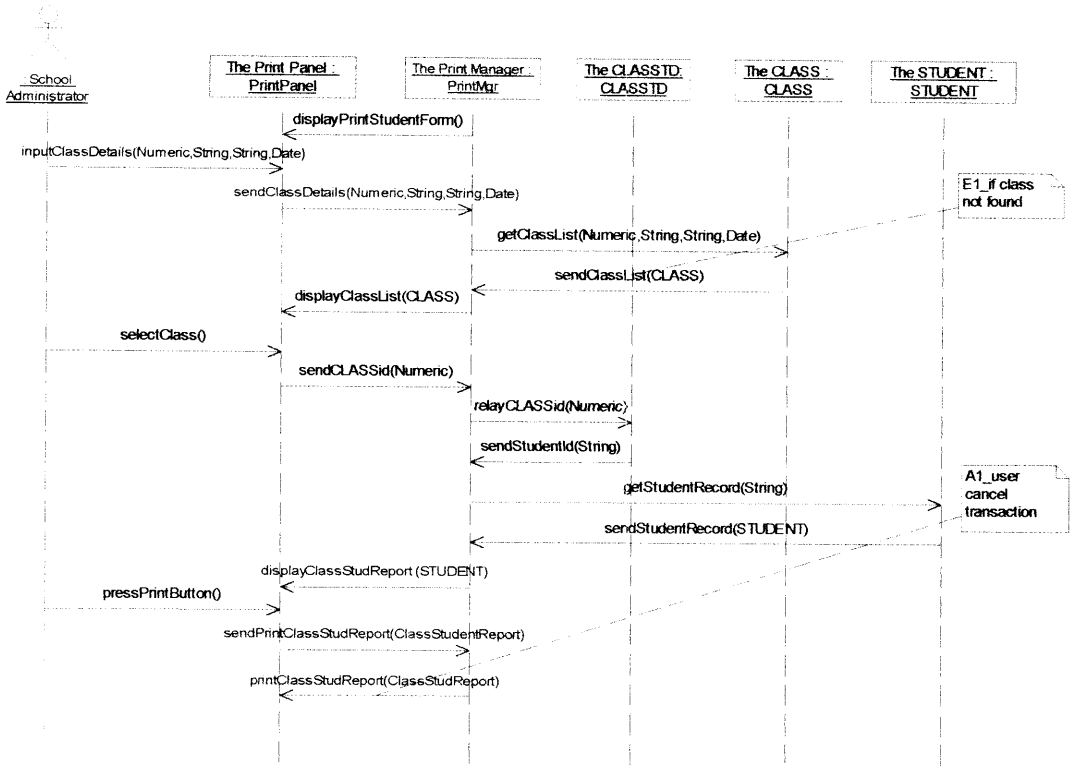


Figure 3.38 Sequence Diagram Print Student Record by Class

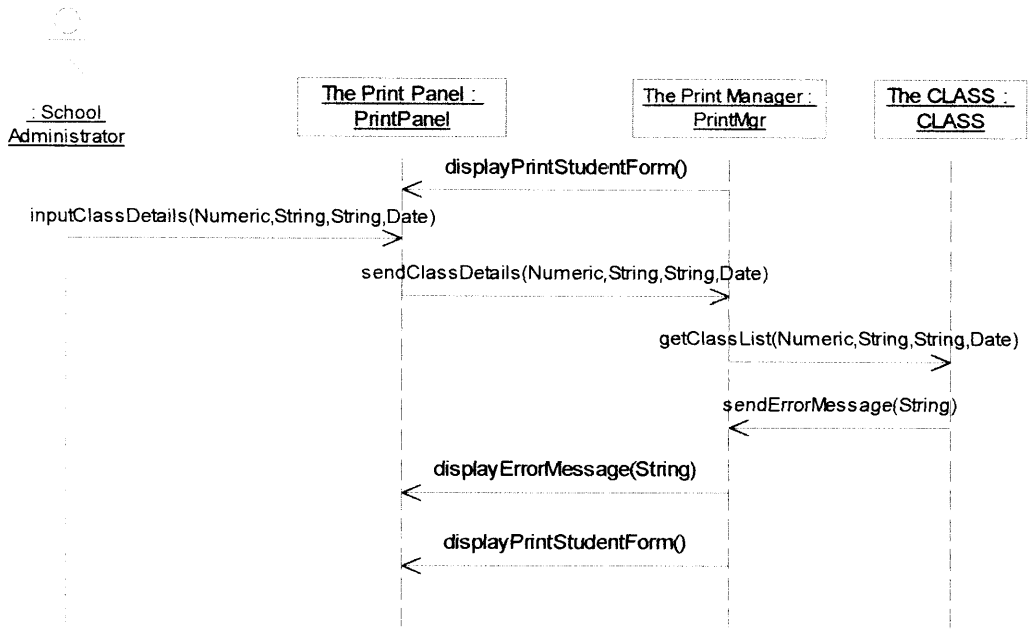


Figure 3.39 Sequence Diagram Print Student Record by Class/E1_if class not found

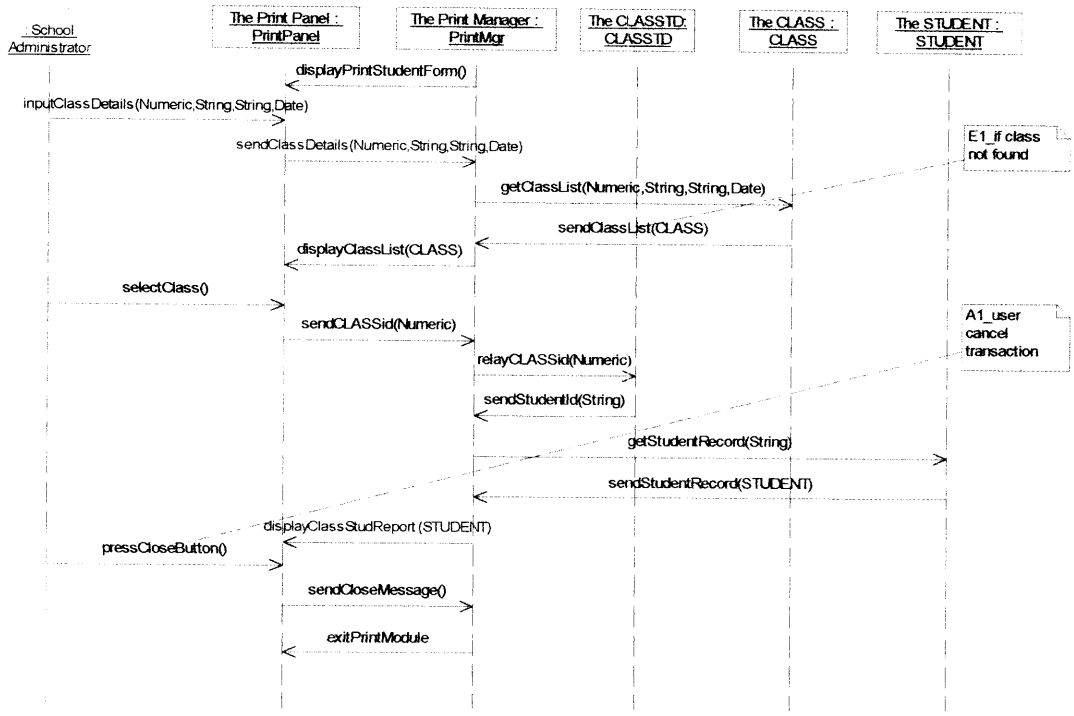


Figure 3.40 Sequence Diagram Print Student Record by Class/A1_if user cancels transaction

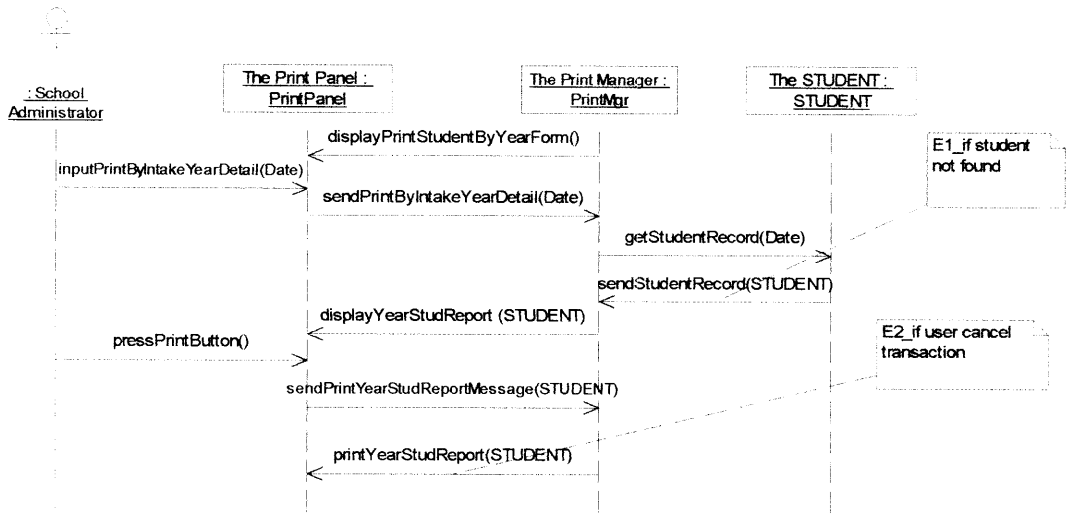


Figure 3.41 Sequence Diagram Print Student Record by Intake

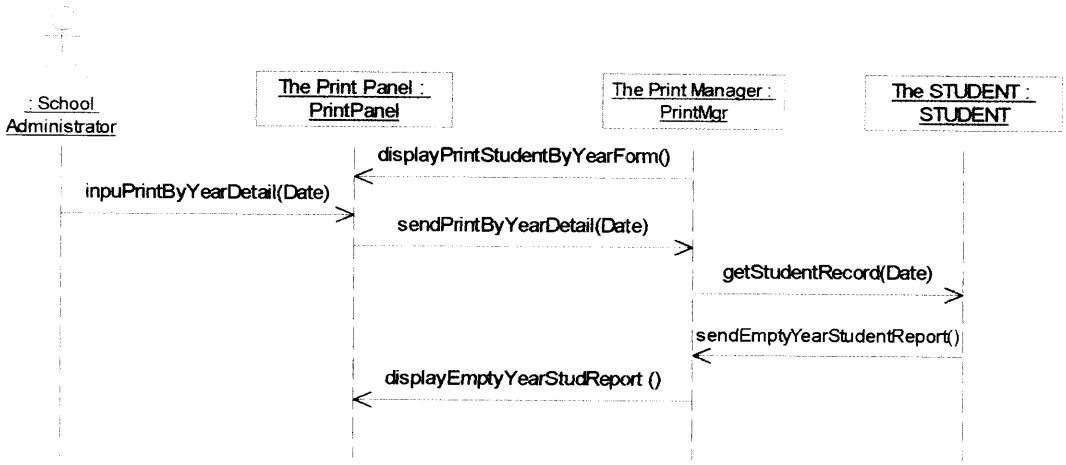


Figure 3.42 Sequence Diagram Print Student Record by Intake/E1_if record not found

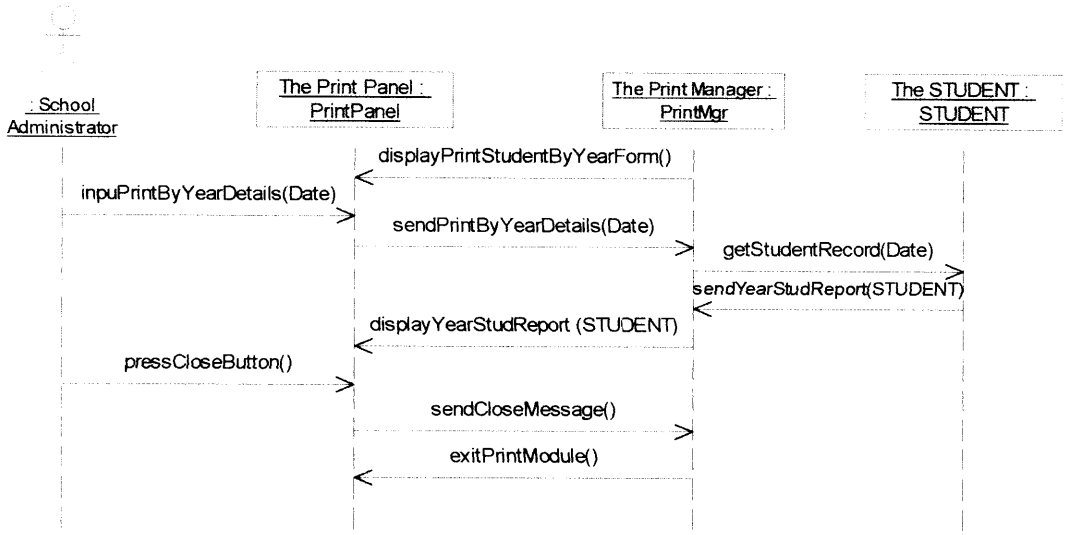


Figure 3.43 Sequence Diagram Print Student Record by Intake/A1_if user cancels transaction

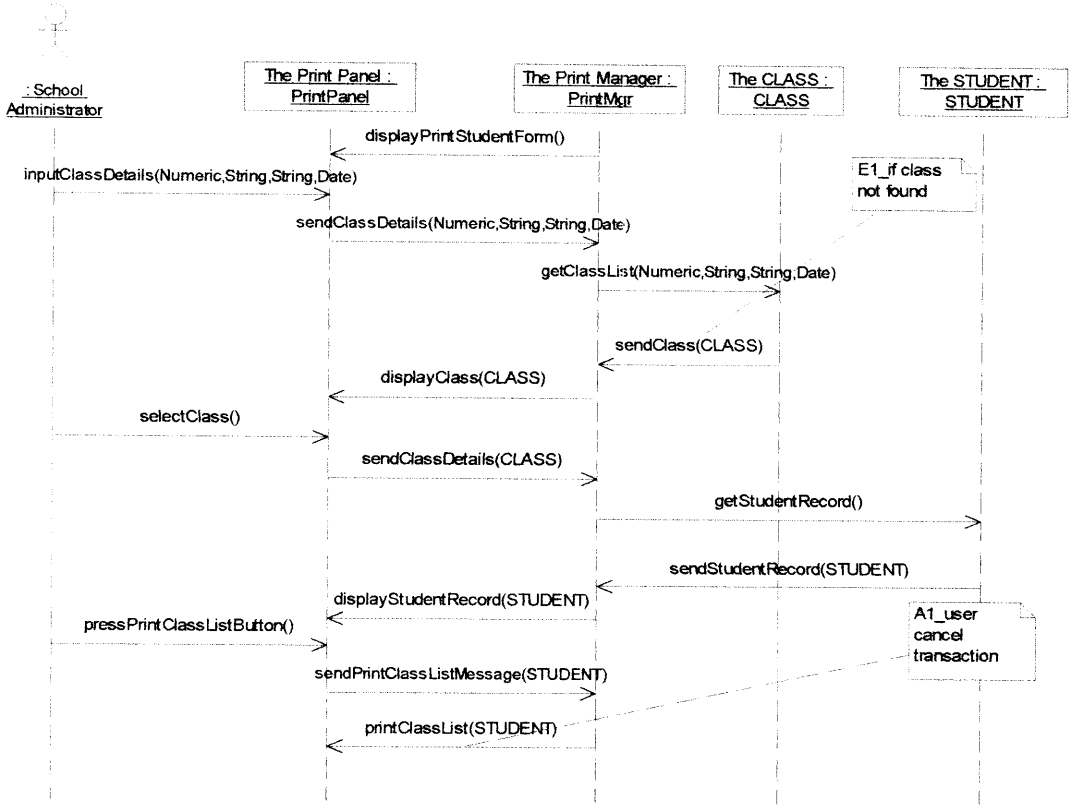


Figure 3.44 Sequence Diagram Print Class List

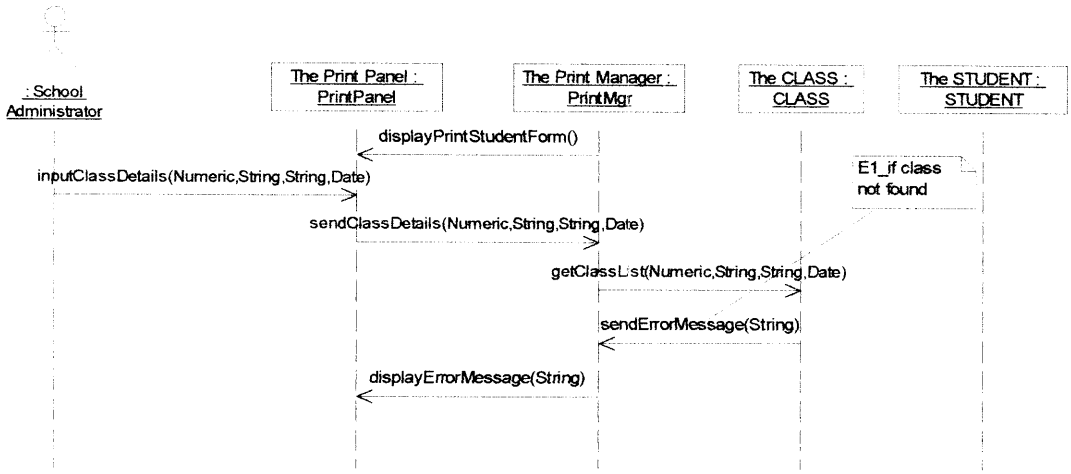


Figure 3.45 Sequence Diagram Print Class List/E1_if class not found

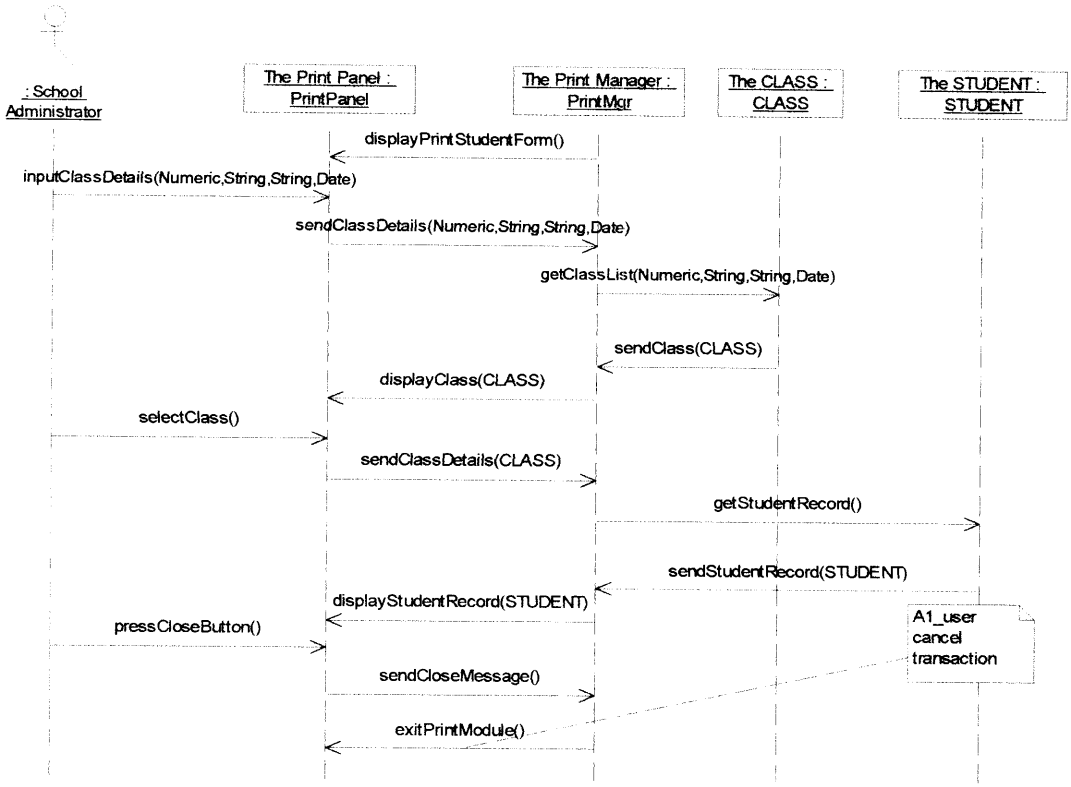


Figure 3.46 Sequence Diagram Print Class List/A1_if user cancels transaction

3.1.3 Model Showing Internal Classes

The diagrams and descriptions below show the visibility of Presentation Layer, Application Logic Layer and Storage Layer thus giving and overall picture of the model.

The presentation layer package groups the elements that will administer system and manage staff activities. The classes that can be found in this package are SystemPanel, SystemAdminLogin Panel, StaffLogin Panel, StudentLogin Panel, SystemAdmin Panel, StudentEnrollment Panel, and ClassAdmin Panel. The classes for SystemAdminLogin Panel, StaffLogin Panel, StudentLogin Panel, SystemAdmin Panel, StudentEnrollment Panel, ClassAdmin Panel are inheriting System Panel. The presentation layer package communicates with the application layer package, as shown below.

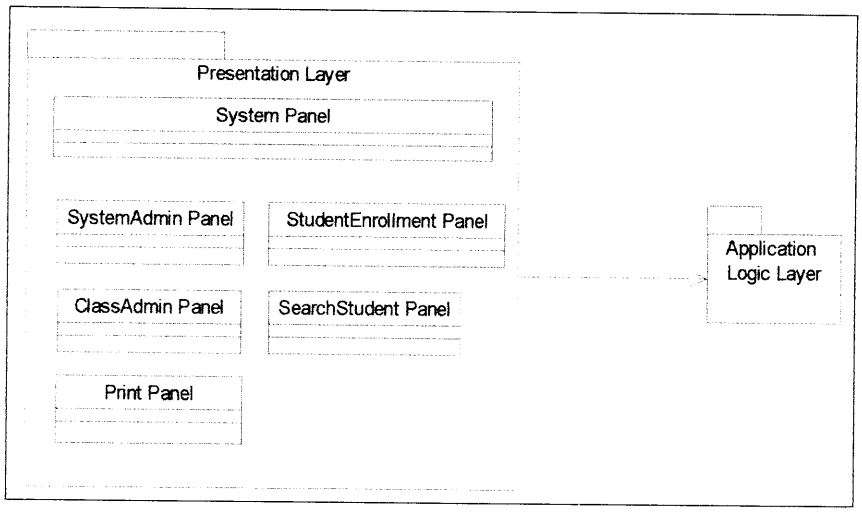


Figure 3.47 Visibility of Presentation Layer with Other Package

The application logic layer package groups the elements that will be managing and controlling the system. The classes that can be found in this package are SystemAdmin Manager, StudentEnrollment Manager, ClassAdmin Manager, SearchStudent Manager and Print Manager. Communication with others package like boundary and entity is shown below.

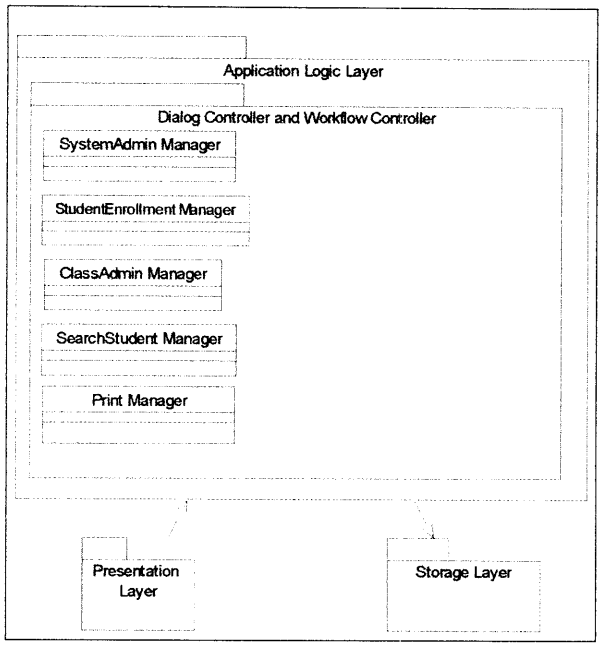


Figure 3.48: Visibility of Application Logic Layer with Other Packages

The storage layer package groups the elements that manage record of the system. The classes that can be found in this package are STUDENT, STAFF, CLASS, and CLASSTD. The SES entity communicates with the application logic layer package as shown below.

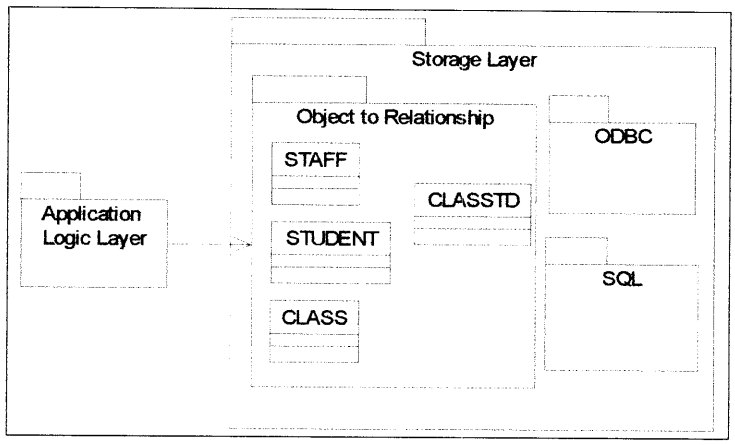


Figure 3.49: Visibility of Storage Layer with Other Packages

3.1.4 Class Diagrams

Class Diagram in details design is all classes with the types of attributes specified and the operation parameters defined, which gives overall working of the model.

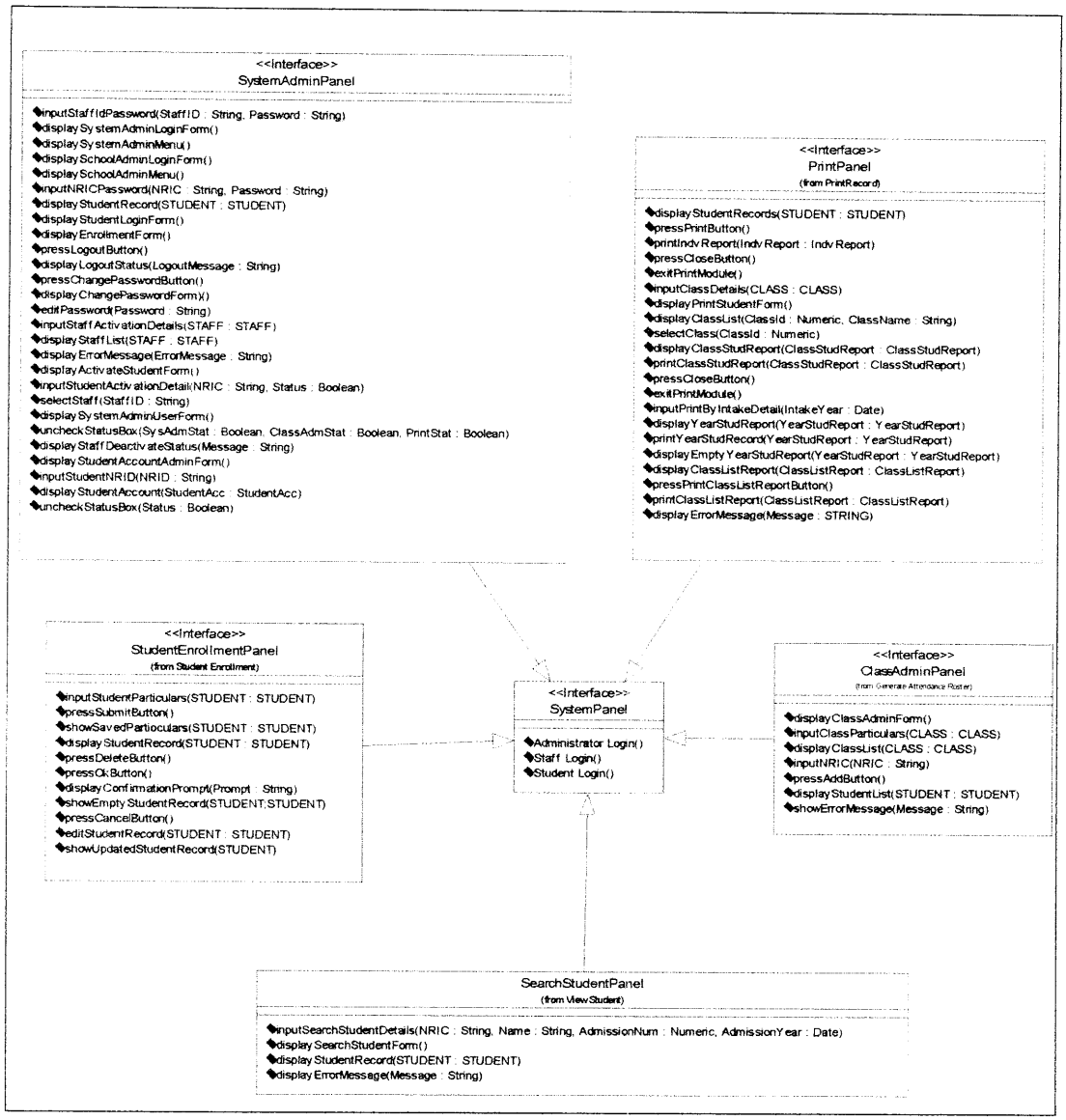


Figure 3.50 Presentation Layer Package

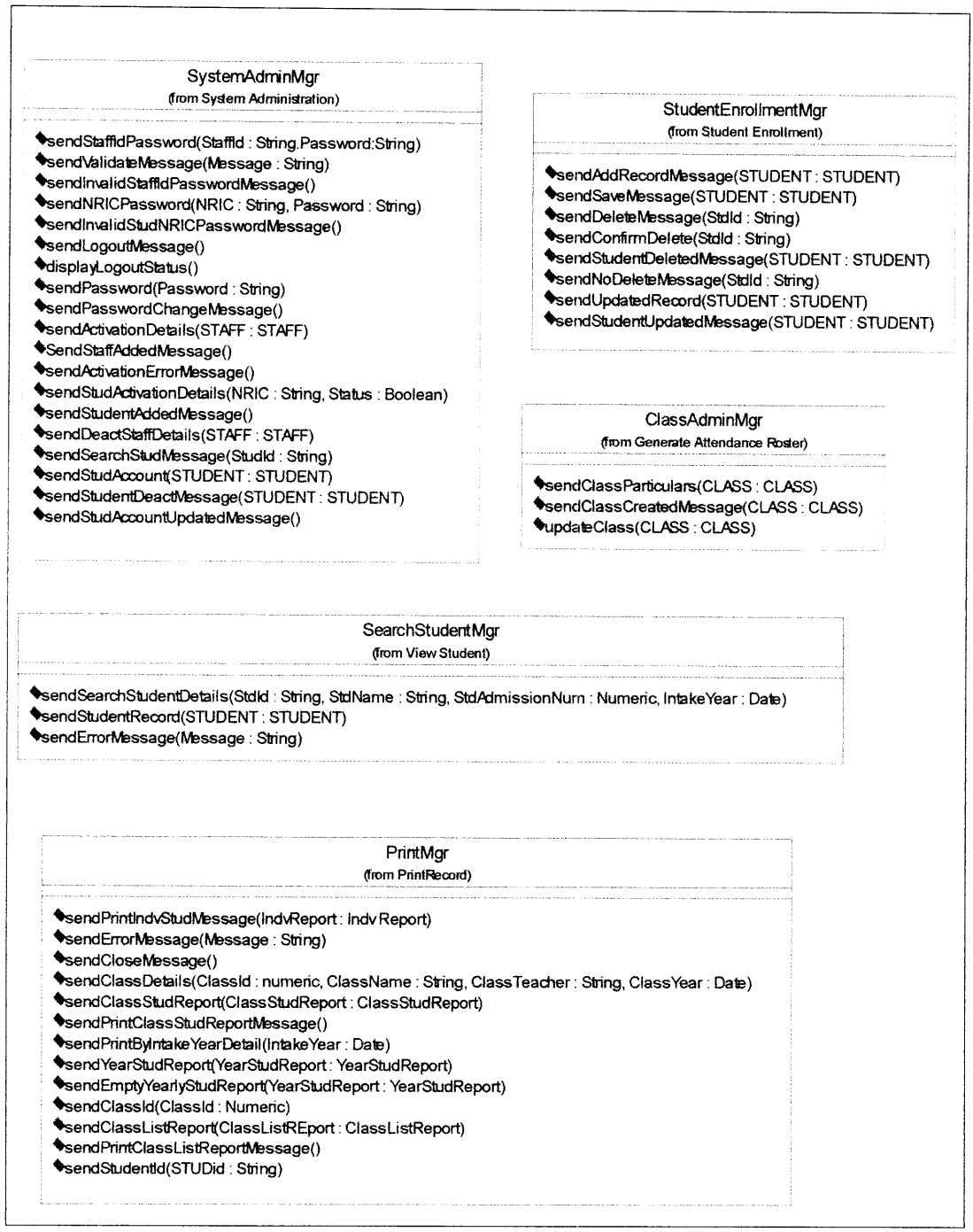


Figure 3.51 Application Logic Layer Package

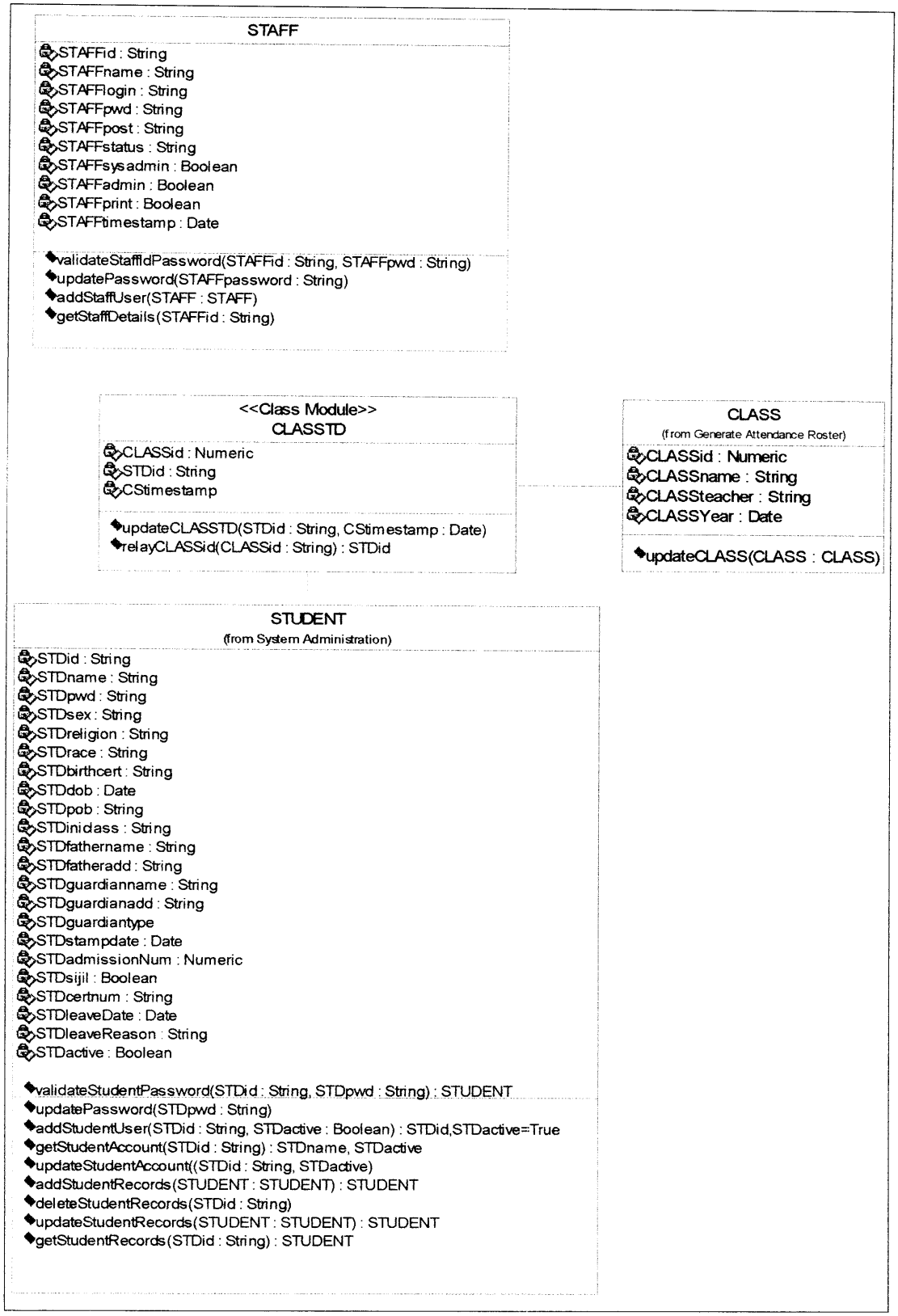


Figure 3.52 Storage Layer package

3.1.5 Data Specifications

The data specifications show the persistant data of the model.

STAFF Class			
Attribute	**Type**	**Initial Value**	**Constraints**
STAFFid	String	"Unknown"	Length=1-15
STAFFname	String	"empty"	Length=0-100
STAFFlogin	String	"empty"	Length=0-15
STAFFpwd	String	"empty"	Length=0-15
STAFFpost	String	"empty"	Length=0-50
STAFFstatus	String	"empty"	Length=0-5
STAFFsysadmin	Boolean	"False"	-
STAFFadmin	Boolean	"False"	-
STAFFprint	Boolean	"False"	-
STAFFtimestamp	datetime	"empty"	-
STUDENT Class			
STDid	String	"unknown"	Length=1-15
STDname	String	"empty"	Length=0-100
STDpwd	String	"empty"	Length=0-100
STDsex	String	"empty"	Length=0-100
STDreligion	String	"empty"	Length=0-100
STDrace	String	"empty"	Length=0-100
STDbirthcert	String	"empty"	Length=0-100
STDdob	datetime	"empty"	Length=0-100
STDpob	String	"empty"	Length=0-100
STDiniclass	String	"empty"	Length=0-15
STDfathername	String	"empty"	Length= 0-100
STDfatheradd	String	"empty"	Length=0-200
STDguardianname	String	"empty"	Length= 0-100
STDguardianadd	String	"empty"	Length=0-200
STDguardiantype	String	"empty"	Length=0-50
STDstampdate	datetime	"empty"	-
STDadmissionNum	numeric	"unknown"	Length=1-18
STDsijil	Boolean	"False"	-
STDcertnum	String	"empty"	Length=0-12
STDleavedate	datetime	"empty"	-
STDleaveReason	String	"empty"	Length=0-100
STDactive	Boolean	"False"	-
CLASSid	numeric	"unknown"	Digit 1-15
CLASSname	String	"Empty"	Length=0-50
CLASSteacher	String	"Empty"	Length=0-50
CLASSyear	datetime	"Empty"	
CLASSTD Class			
CLASSid	numeric	"unknown"	Digit 1-15
STDid	String	"unknown"	Length=1-15
CStimestamp	Datetime	"empty"	-

Table 3.6 Data Specifications

3.1.6 Testing

T E S t No	Test Description/ (condition)	Test Data/ Action	Expected Result	Result
1	Display SystemAdmin Login Form	User has click Administrator Login on main page	System Administrator Login form is displayed.	System Administrator Login form is displayed.
2	Input StaffId and Password	Valid STAFFlogin and STAFFpwd	SystemAdmin Menu is displayed	SystemAdmin Menu is displayed
3	Input StaffId and Password	Invalid STAFFlogin and STAFFpwd	Error messages "Login or password is incorrect". Login Form redisplays.	Error messages "Login or password is incorrect". Login Form redisplays.
4	Display School AdminLogin Form	User has click Staff Login on main page.	School Administrator Login form is displayed	School Administrator Login form is displayed
5	Input StaffId and Password (school administrator)	Valid STAFFlogin and STAFFpwd	School Administrator Menu is displayed	School Administrator Menu is displayed
6	Input StaffId and Password/ (school administrator)	Invalid STAFFlogin and STAFFpwd	Error messages "Login or password is incorrect". School Administrator Login Form redisplays.	Error messages "Login or password is incorrect". School Administrator Login Form redisplays.
7	Display Student Login Form	User has click Student Login on main page.	Student Login form is displayed	Student Login form is displayed
8	Student Display Own Record	Valid STDid, STDpwd	Student's personal record being displayed.	Student's personal record being displayed.
9	Display Student Record	Invalid STDid, STDpwd	Display "Login or Password is incorrect". Student Login Form being redisplay	Display "Login or Password is incorrect". Student Login Form being redisplay
10	Display Enrollment Form/ (details not filled)	Valid STDid, STDpwd	Student enrollment form being displayed.	Student enrollment form being displayed.
11	Display Logout Status	User has pressed the logout button.	Display "Thank you for using Student Enrollment System."	Display "Thank you for using Student Enrollment System."
12	Display Change Password Form	User has pressed change password button.	Change Password Form is displayed	Change Password Form is displayed
13	Edit Password	New Password	Password Changed	Password Changed
14	Display Activate Staff Form	System Administrator has pressed System User Administration button	Activate Staff Form is displayed.	Activate Staff Form is displayed.

T E S t No	Test Description/ (condition)	Test Data/ Action	Expected Result	Result
15	Activate Staff	New STAFF	A Staff System User List including the new user is displayed.	A Staff System User List including the new user is displayed.
16	Activate Staff/ (Staff activated)	Old STAFF	Display "System user already activated".	Display "System user already activated".
17	Display Staff Details	System Administrator clicked on the STAFFid in the staff list.	Staff record is displayed	Staff record is displayed
18	Deactivate User	System Administrator unchecked STAFFsysadmin, STAFFadmin, STAFFprint checkbox.	An edited staff record is displayed.	An edited staff record is displayed.
19	Display Student Account Admin Form	System Administrator has pressed student account admin button	Student Account Admin Form is displayed.	Student Account Admin Form is displayed.
20	Display Student Account	STDid /System Administrator input Student Account Admin Form	Student account is displayed.	Student account is displayed.
21	Activate Student		New student user added.	New student user added.
22	Activate Student	Existing STDid	Error, Student not added.	Error, Student not added.
23	Student Input Own Particulars	STUDENT	Display saved student particulars.	Display saved student particulars
24	Display Search Student Form	School Administrator press Manage Student Record Button	Display search student form.	Display search student form.
25	School Administrator Search Student Record	Valid STDid	Display student record.	Display student record.
26	School Administrator Search Student Record	Invalid STDid	Display "No record found"	Display "No record found"
27	Delete Student Record	School Administrator Pressed Delete Student Button	Display emptied student record.	Display emptied student record.
28	Edit Student Record	School Administrator Overwrite student particulars.	Edited student record is displayed.	Edited student record is displayed.

T E S t No	Test Description/ (condition)	Test Data/ Action	Expected Result	Result
29	Student Edit Student Record	Student edit own record.	Not possible.	Not possible.
30	Display Class Admin Form	School Administrator pressed class admin button.	Class Admin Form is displayed.	Class Admin Form is displayed.
31	Create Class	CLASS	New class list with latest inclusion being displayed.	New class list with latest inclusion being displayed.
32	Add members to class	STDid	Student list with latest inclusion.	Student list with latest inclusion.
33	Add members to class	Invalid STDid	Display "Student not found"	Display "Student not found"
34	Display print student form	STAFFid and STAFFpwd/School Administrator login to print module.	Print Student Form is displayed.	Print Student Form is displayed.
35	Display individual student report	STDid/School Administrator input STDid in print student form.	Individual student report is displayed.	Individual student report is displayed.
36	Print Individual Student Report	School Administrator pressed print button	Individual student report printed.	Individual student report printed.
37	Display class student report	CLASS/School Administrator input CLASS in print student form.	Student report for selected class is displayed.	Student report for selected class is displayed.
38	Print Class Student Report	School Administrator pressed print button	Class Student Report printed.	Class Student Report printed.
39	Display YearStudent report	Intake Year/School Administrator input Intake Year in print student form.	Student report of selected year is displayed.	Student report of selected year is displayed.
40	Print Year Student Report	School Administrator pressed print button	Student report for the whole selected intake year is printed.	Student report for the whole selected intake year is printed.
41	Display Class List	CLASS/School Administrator input CLASS in print student form.	Class List is displayed.	Class List is displayed.
42	Print Class List	School Administrator pressed print button	Selected Class List is printed.	Selected Class List is printed.

Table 3.7 Test Plan and Results

3.2 Discussion

The test results done on the prototype developed based on the proposed model shows that it is not possible to create duplicate student records by users without direct access to the database. It is also not possible for student to update their record once it is submitted. The model thus will guide the design of student information management system that will solve the data redundancies and data integrity problems besides helping in storage and retrieval of student data.

The model also shows the possibility of redistributing burden and roles in managing information in education sector. There are a few reservations and shortcommings though. It is observed that some students faced difficulties in filling up the form. They need the assistants of teachers. If left to enroll from home via Internet, they will not be able to accomplish the task. With low penetration of Internet user and associated cost, it is also feared that many may not enroll via Internet. A possible solution for this category of user would be allowing enrollment from school computer laboratory.

In the effort to enforce control and redistribution of burdens and role, the prototype allows granting of System Administration right, Student and Class Administration right and Printing right. As an afterthought such rights should be further redefined to grant system user more specific rights needed. For example, the right to manage student record should be sub-divided into right to view, right to update, and right to delete. This is also through in granting printing rights as there are four types of reports involve. Not all the system user should be granted full printing rights.

4.0 SIGNIFICANCE OF STUDY

The model suggested has shown the possibility of apportioning student management task in a more equitable basis. It showed that a web based student enrollment system allows better responsibility sharing. Through better responsibility sharing School administrator mainly class teacher can save some valuable time in student management. Basically the prototype shows the following possibilities: (1) Reduce the hassle of first day registration as student can register from home and be verified later, (2) Reduce cost in term of saving valuable teaching time, (3) Improve data integrity, (4) Improve storage and retrieval of records, (5) Allow teacher to manage student data anytime and any place, (6) Facilitate certification and document preparation. This model thus expects to show some significant impact on school information management.

5.0 CONCLUSION AND RECOMENDATION

It is envisage that other areas of school information management should work along the principle of task or responsibility sharing. Obviously this model can be employed in many areas of information management in school scene such co-curriculum registration, staff management and any other personnel management that will need personal particulars to be input. Managers in our education sector should also explore the possibility of increasing adoption of Information Technology in managing School and other education setups as there are still large uncharted water.

APPENDIX A

Software Requirement Specification

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

for

The Web-based Student Enrollment System for Malaysian Government Secondary School

Student Enrollment System (SES)

Prepared For: Graduate School, Universiti Utara Malaysia

Prepared By: Gee Kheng Chuan (83231)

TABLE OF CONTENTS

(SRS) 1
LIST OF FIGURES V
TABLE OF APPENDIX V

1.0 INTRODUCTION 1
1.1 Identification.....1
1.2 System Overview 1
1.3 Document Overview 1

2.0 REFERENCED DOCUMENTS 2

3.0 ENGINEERING REQUIREMENTS 3
3.1 System External Interface Requirements 3
3.2 System Capability Requirements 5
3.2.1 Use Case Activate User [SRS_SES_100] 7
3.2.1.1 Actor 7
3.2.1.2 Brief Description 7
3.2.1.3 Activation Condition 7
3.2.1.3 Pre-Condition (S) 7
3.2.1.4 Description (SRS_REQ 101) 7
3.2.1.5 Alternative Flows 7
3.2.1.6 Exception Flows 7
3.2.1.7 Post-Condition 7
3.2.1.8 Rule(s) 7
3.2.1.9 Constraint(s) 7
3.2.2 Use Case Deactivate User [SRS_SES_200] 7
3.2.2.1 Actor 7
3.2.2.2 Brief Description 7
3.2.2.3 Activation Condition 7
3.2.2.4 Pre Condition (s) 7
3.2.2.5 Description (SRS_REQ_201) 7
3.2.2.6 Alternate Flows 8
3.2.2.7 Exception Flows 8
3.2.2.8 Post-Condition 8
3.2.2.9 Rule(s) 8
3.2.2.10 Constraint(s) 8
3.2.3 Use Case LOG IN [SRS_SES_300] 8
3.2.3.1 Actor 8
3.2.3.2 Brief Description 8
3.2.3.3 Activation Condition 8
3.2.3.4 Pre-Condition (s) 8
3.2.3.5 Description (SRS_REQ_301) 8
3.2.3.6 Alternative Flows 8
3.2.3.7 Exception Flows 8
3.2.3.8 Post-Condition 8
3.2.3.9 Rule(s) 8
3.2.3.10 Constraint(s) 9
3.2.4 Use case CHANGE PASSWORD [SRS_SES_400] 9
3.2.4.1 Actor 9
3.2.4.2 Brief Description 9
3.2.4.3 Activation Condition 9
3.2.4.4 Pre-Condition(s) 9

3.2.4.5 Description (SRS_REQ_401) 9
3.2.4.6 Alternative Flows 9
3.2.4.7 Exception Flows 9
3.2.4.8 Post-Condition 9
3.2.4.9 Rule(s) 9
3.2.4.10 Constraint(s) 9
3.2.5 Use case Logout [SRS_SES_500] 9
3.2.5.1 Actor 9
3.2.5.2 Brief Description 9
3.2.5.3 Activation Condition 9
3.2.5.4 Pre-Condition (s) 10
3.2.5.5 Description (SRS_REQ-501) 10
3.2.5.6 Alternative Flows 10
3.2.5.7 Exception Flows 10
3.2.5.8 Post-Condition 10
3.2.5.9 Rule(s) 10
3.2.5.10 Constraint(s) 10
3.2.6 Use Case Add Student Particulars [SRS_SES_600] 11
3.2.6.1 Actor 11
3.2.6.2 Brief Description 11
3.2.63 Activation Condition 11
3.2.6.4 Pre-Condition (s) 11
3.2.6.5 Description (SRS_SES_601) 11
3.2.6.6 Alternative Flows 11
3.2.6.7 Exception Flows 11
3.2.6.8 Post-Condition 11
3.2.6.9 Rule(s) 11
3.2.6.10 Constraint(s) 11
3.2.7 Use Case Delete Student Record [SRS_SES_700] 12
3.2.7.1 Actor 12
3.2.7.2 Brief Description 12
3.2.7.3 Activation Condition 12
3.2.7.4 Pre-Condition (s) 12
3.2.7.5 Description (SRS_REQ_501) 12
3.2.7.6 Alternative Flows 12
3.2.7.7 Exception Flows 12
3.2.7.8 Post-Condition 12
3.2.7.9 Rule(s) 12
3.2.7.10 Constraint(s) 12
3.2.8 Use Case Update Student Record [SRS_SES_ 800] 12
3.2.8.1 Actor 12
3.2.8.2 Brief Description 12
3.2.8.3 Activation Condition 12
3.2.8.4 Pre-Condition (s) 12
3.2.8.5 Description (SRS_ REQ_801) 12
3.2.8.6 Alternative Flows 13
3.2.8.7 Exception Flows 13
3.2.8.8 Post-Condition 13
3.2.8.9 Rule(s) 13
3.2.8.10 Constraint(s) 13
3.2.9 Use Case Print Student Record [SRS_SES_900] 13
3.2.9.1 Actor 13
3.2.9.2 Brief Description 13
3.2.9.3 Activation Condition 13
3.2.9.4 Pre-Condition (s) 13
3.2.9.5 Description (SRS_REQ_901) 13

3.2.9.6 Alternative Flows 13
3.2.9.7 Exception Flows 13
3.2.9.8 Post-Condition 14
3.2.9.9 Rule(s) 14
3.2.9.10 Constraint(s) 14
3.2.10 Use Case Search Student Record [SRS_SES_1000] 14
3.2.10.1 Actor 14
3.2.10.2 Brief Description 14
3.2.10.3 Activation Condition 14
3.2.10.4 Pre-Condition (s) 14
3.2.10.5 Description (SRS_REQ_1001) 15
3.2.10.6 Alternative Flows 15
3.2.10.7 Exception Flows 15
3.2.10.8 Post Condition 15
3.2.10.9 Rule(s) 15
3.2.10.10Constraint(s) 15
3.2.11 Use Case Generate Class [SRS_SES_1100] 16
3.2.11.1 Actor 16
3.2.11.2 Brief Description 16
3.2.11.3 Activation Condition 16
3.2.11.4 Pre-Condition(s) 16
3.2.11.5Description (SRS_REQ_1101) 16
3.2.11.6 Alternative Flows 16
3.2.11.7 Exception Flows [SRS_REQ_1102] 16
3.2.11.8 Post Condition 16
3.2.11.9 Rule(s) 16
3.2.11.10Constraint(s) 17
3.2.12 Use Case Add Class Member [SRS_SES_1200] 17
3.2.12.1 Actor 17
3.2.12.2 Brief Description 17
3.2.12.3 Activation Condition 17
3.2.12.4 Pre-Condition(s) 17
3.2.12.5 Description [SRS_REQ_1201] 17
3.2.12.6 Alternative Flows [SRS_REQ_1202] 17
3.2.12.7 Exception Flows 17
3.2.12.8 Post Condition 17
3.2.12.9 Rule(s) 17
3.2.12.10Constraint(s) 17
3.2.13 Use Case Print Class List [SRS_SES_1300] 17
3.2.13.1 Actor 17
3.2.13.2 Brief Description 17
3.2.13.3 Activation Condition 18
3.2.13.4 Pre-Condition(s) 18
3.2.13.5 Description [SRS_REQ_1301] 18
3.2.13.6 Alternative Flows [SRS_REQ_1302] 18
3.2.13.7 Exception Flows 18
3.2.13.8 Post Condition 18
3.2.13.9 Rule(s) 18
3.2.13.10Constraint(s) 18
3.2.14 Use Case Print Class Member Particulars [SRS_SES_1400] 18
3.2.14.1 Actor 18
3.2.14.2 Brief Description 18
3.2.14.3 Activation Condition 18
3.2.14.4 Pre-Condition (s) 18
3.2.14.5 Description [SRS_REQ_1401] 18
3.2.14.6 Alternative Flows [SRS_REQ_1402] 19

3.2.14.7 Exception Flows 19
3.2.14.8 Post Condition 19
3.2.14.9 Rule(s) 19
3.2.14.10Constraint(s) 19
3.3 System Internal Requirements 19
3.4 System Data Elements Requirements 21
3.4.1 System Panel Class 21
3.4.2 AdminMgr Class 21
3.4.3 EnrollmentMgr Class 21
3.4.4 ClassMgr 21
3.4.5 SearchMgr 21
3.4.6 User Class 22
3.4.7 StudentRec Class 22
3.4.8 ClassRec Class 22
3.5 Sizing and Timing Requirements 22
3.5.1 Timing 22
3.5.2 Sizing 22
3.6 Security Requirements 22
3.7 Design Constraints 22
3.7.1 Programming Language 22
3.7.2 Programming Practices 22
3.8 Software Quality Factors 22
3.8.1 Internal Consistency 22
3.8.2 Understandability 22
3.8.3 Testability 22
3.8.4 Traceability 23
3.8.5 Consistency 23

4.0 PREPARATION FOR DELIVERY 23

5.0 NOTES 23

LIST OF FIGURES

Figure 3.1 Context Diagram 3
Figure 3.2 Use Case Package Diagram for SES 5
Figure 3.3 Use Case Diagram for System Administration 6
Figure 3.4 Use Case Diagram for Student Enrollment Use Case Package 10
Figure 3.5 Use Case Diagram for Search Student Record Use Case Package 14
Figure 3.6 Use Case Diagram for Generate Attendance Roster Use Case ackage 15
Figure 3.7: Class Diagram Package for Use Case System Administration 19
Figure 3.8: Class Diagram Package Use Case Student Enrollment 20
Figure 3.9: Class Diagram Package Use Case View Student Record 20
Figure 3.10: Class Diagram Package Use Case Generate Attendance Roster 21

TABLE OF APPENDIX

Appendix A: Sequence Diagram
Appendix B: State Diagram
Appendix C: Class Diagram

1.0 INTRODUCTION

1.1 Identification

System Name : < Web-based Student Enrollment System >
System Abbreviation : < SES >

1.2 System Overview

This project is a web based student enrollment system for Malaysian secondary schools. This web based system is mean to manage the student enrollment data and generate student data for attendance roster. This web based system is important for government secondary school as it takes away most of the student personal data entry role from teacher and administrators and pass it to the would be students. This leaves the administrator and teachers to do much lesser work of maintenance. The main objectives are to store student's enrollment data in order to minimize the data redundancies; to search and retrieve student records based on the National Registration Identity Card (NRIC) or birth certificate number and; to generate attendance roster with intention to improve data integrity of students attendance roster.

1.3 Document Overview

The purpose of this system requirement specification document is to file up all the user requirement towards the system to be developed. The developer can refer to this document time to time during development period. It is to make sure that the developed system does fulfill the user needs. In this case the percentage of develop the wrong system may be low. Besides that, this document can give a brief picture about the function of the system and actor involve in the system to the developer. It also uses to specify the scope of the system. The content of the system included a list of diagram like context diagram, use case diagram, list of table and appendixes (Appendix A: Sequence Diagram, Appendix B: State Diagram, Appendix C: Class Diagram and Appendix D: Summary of Requirement)

The first content is the introduction, which included identification, system overview and document overview. The second content is the reference documents. Engineering Requirement is the third content. It included System External Interface Requirements (3.1), and System Capability Requirements (3.2), which explain the use case. System Data Elements Requirements also included in this document. (3.3). besides the system requirements this document also included sizing and timing requirements (3.4). All the time constraint and size of the system need to be determined.
The security of the system needed to be documented (3.5); sometimes the designer will be facing design constraint like in choosing which programming language to use so this also needed to be documented. (3.6). The software quality factors, which in section (3.7) is to evaluate the quality of this document and the quality of the development process. The fourth content included the delivery items. Lastly, all the abbreviation used the document also need to be listed down.

2.0 REFERENCED DOCUMENTS

Bennett,S.& McRobb,S., Farmer,R. *Object-Oriented System Analysis and Design.* McGraw-Hill Publishing Company.

Boggs, Wendy., & Michael (1999). *Mastering UML with Rational Rose.* United States of America: SYBEX Inc.

Graham, Ian.(2001). *Object-oriented Method Principles and Practice.* 3rd Edition. Addisson Wesley, Pearson Education.

Simon Bennett, Steve McRobb & Ray Farmer (2002). *Object-oriented System Analysis and Design* Using UML. McGraw-Hill Publishing Company.

Frank Armour, Granville Miller (2001). *Advanced Use Case Modeling Software System.* Addison Wesley Publishing.

Oestereich, Bernd (1999). *Developing Software with UML: Object Oriented Analysis and Design in* Practice. Addison Wesley Publishing.

Roger S. Pressman (2001). *Software Engineering A Practitioner's Approach.* McGraw Hill

3.0 ENGINEERING REQUIREMENTS

3.1 System External Interface Requirements

Below is the overview of the use case external interface requirement for Administration Management System.

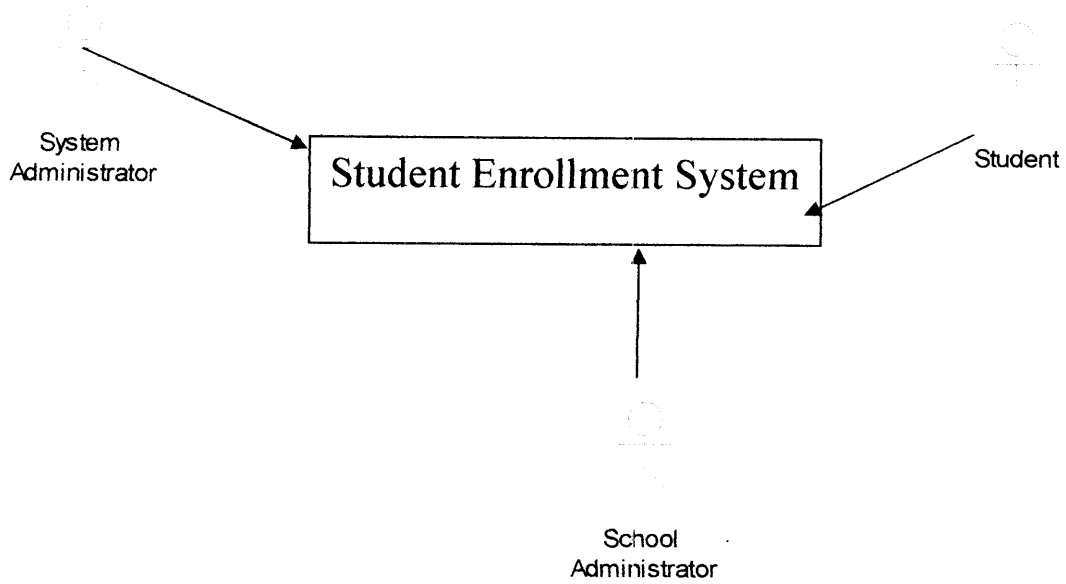


Figure 3.1 Context Diagram

Interface System Administrator/SES
Interface Identification : System Administrator
Interface Type : Person

Description:
The task of the system administrator is log in system, activate user, deactivate user and check password. The system administrator uses password to log in. They control authorization of the system user.

Association:

SYSTEM ADMINISTRATION USE CASE PACKAGE

1. System Administrator will communicate with the Log In System use case. System Administrator has their password to log in the administration system. After log in, administration menu will show.

2. System Administrator will communicate with the Activate User use case. System administrator may deactivate user in order to disable them to use the system.

3. System Administrator will communicate with the Deactivate User use case. System administrator may deactivate user in order to disable them to use the system.

Interface School Administrator/SES
Interface Identification : School Administrator
Interface Type : Person

Description:
The task of the school administrator is log in, log out, change password, update student record, view student record, delete student record, print student record, generate class, print Class Member Particulars and print class list.

Association:

SYSTEM ADMINISTRATION USE CASE PACKAGE

1. The School Administrator will communicate with the Log In use case.
   Every School Administrator has their user ID and password to log in. After log in, interface menu will show.

2. The School Administrator will communicate with the Log Out use case.
   Every School Administrator must log out to exit their web site and exit from system.

3. The School Administrator will communicate with the Change Password use case.
   Every School Administrator can change their password, which is used when log in system.

STUDENT ENROLLMENT USE CASE PACKAGE

1. The School Administrator will communicate with the Delete Student Record use case. The School Administrator deletes student's record using this use case

2. The School Administrator will communicate with the Update Student Record use case. The School Administrator updates student's record using this use case.

3. The School Administrator will communicate with the Print Student Record use case. The School Administrator print student record by using this use case.

VIEW STUDENT RECORD USE CASE PACKAGE

1. The School Administrator will communicate with the View Student Record use case. The School Administrator search and view student's record using this use case.

GENERATE ATTENDANCE ROSTER USE CASE PACKAGE

1. The School Administrator will communicate with the Generate Attendance Roster use case. The School Administrator generates classes and adds members to the class.

Interface Student/Ses
Interface Identification : Student
Interface Type : Person

Description:
The task of the student is to log in, log out, change password, add personal particulars, and view student record.

Association:
SYSTEM ADMINISTRATION USE CASE PACKAGE

1. The Student will communicate with the Log In use case.
   Every student has their own user ID and password to log in. After log in, interface menu will show.
2. The Student will communicate with the Log Out use case.
   Every Student must log out to exit their web site and exit from system.

3. The Student will communicate with the Change Password use case.
Every Student can change their password, which is used when log in system.

STUDENT ENROLLMENT USE CASE PACKAGE

1. The Student will communicate with the Add Student Particular use case. The Student add their personal particulars using this use case

VIEW STUDENT RECORD USE CASE PACKAGE

1. The Student will communicate with the View Student Record use case. The Student views their record using this use case.

3.2 System Capability Requirements

Below is the use case package diagram for Student Enrollment System. The use case package will show the overall system overview.

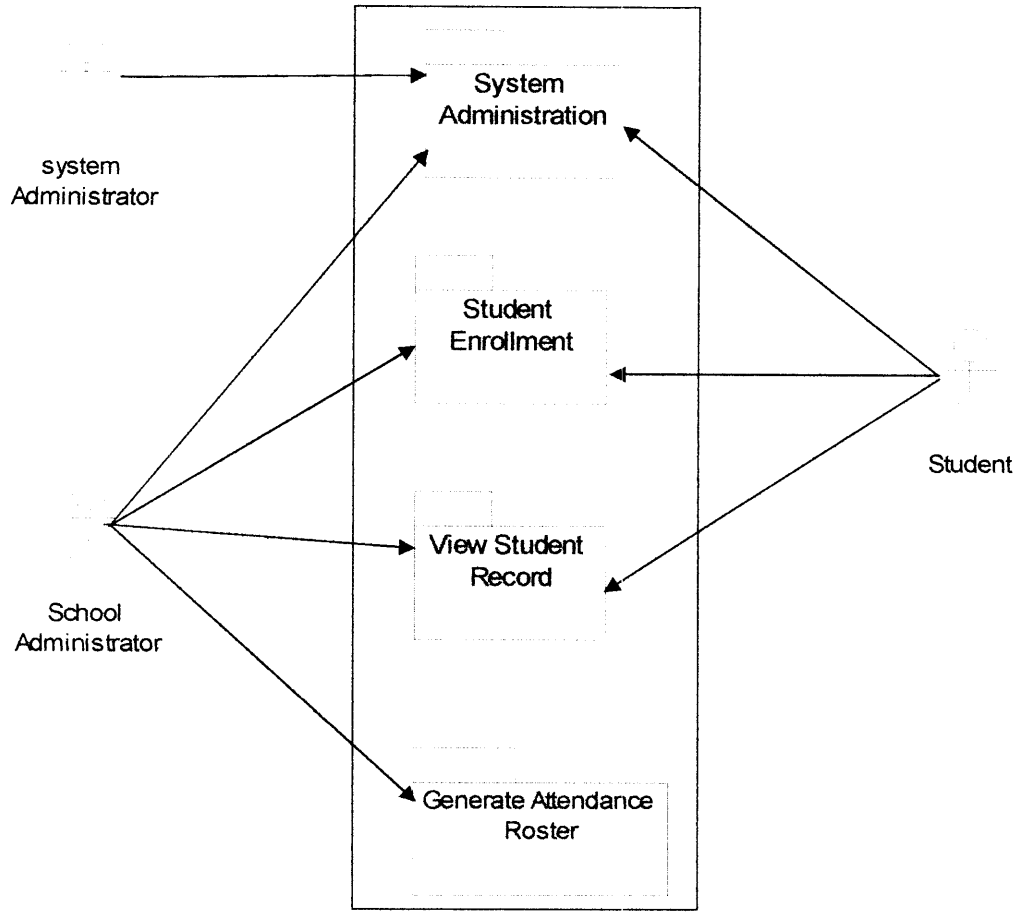


Figure 3.2 Use Case Package Diagram for SES

Description:
SES provide the following services to the user

- System Administration
- Student Enrollment
- View Student Record
- Generate Attendance Roster

Below is the use case diagram for System Administration Use Case Package. The use cases give an overall view of the package.

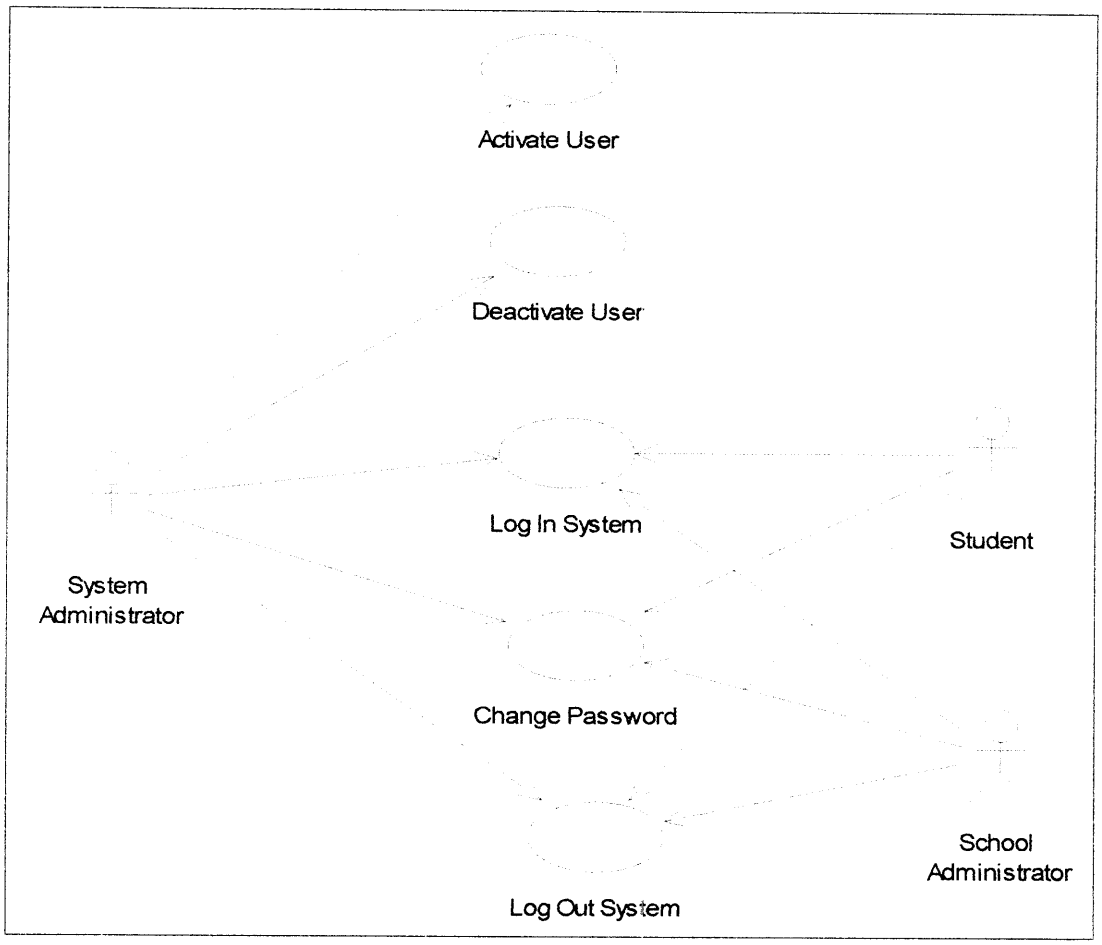


Figure 3.3 Use Case Diagram for System Administration

Use case	Description
Activate User [SRS_SES_l00]	The System Administrator log in system to activate the User by input user name and user id.
Deactivate User [SRS_SES_200]	The System Administrator log in to deactivate user that are longer having any linkage with the school.
Login System [SRS_SES_300]	The System Administrator, the School Administrator, and the Student log into their web site when they input their password and user name.
Change Password [SRS_SES_400]	For security reason, each user can change their password which in default was the NRID or Birth Certificate Number.
Logout System [SRS_SES_500]	The System Administrator, the School Administrator, and the Student log out of the web site after they have finish using the web site. This will exit the user from the system.

Table 1: Description Use Case System Administration

3.2.1 Use Case Activate User [SRS_SES_100]

3.2.1.1 Actor
System Administrator

3.2.1.2 Brief Description
This use case is used by the System Administrator to add user to the system.

3.2.1.3 Activation Condition
Event driven (On System Administrator demand)

3.2.1.3 Pre-Condition (S)
The System Administrator has log in using valid user name and password.

3.2.1.4 Description (SRS_REQ 101)
1. The system will display input box for the System Administrator to input user name and NRID/Birth Certificate Number.
2. System Administrator enter student name and NRID/Birth Certificate Number
3. The system display the name and NRID/Birth Certificate Number entered and prompt for confirmation.
4. System Administrator confirms the entry. [E1_if confirmation denied]
5. At any time the System Administrator can cancel the transaction. [A1-user cancels transaction]

3.2.1.5 Alternative Flows
[A1-transaction cancels]
1. Use case closes.

3.2.1.6 Exception Flows
[E1- if Confirmation Denied] [SRS _REQ_102]
1. The system will prompt for new entry.

3.2.1.7 Post-Condition
Users added.

3.2.1.8 Rule(s)
Not applicable.

3.2.1.9 Constraint(s)
1. Time for showing message box should be done less than 2 seconds.

3.2.2 Use Case Deactivate User [SRS_SES_200]

3.2.2.1 Actor
System Administrator

3.2.2.2 Brief Description
This use case is used by the system to deactivate the user.

3.2.2.3 Activation Condition
Event driven (on System Administrator demand)

3.2.2.4 Pre Condition (s)
The system Administrator log into the administration system.

3.2.2.5 Description (SRS_REQ_201)
1. This use case begins when administrator enter a deactivate user request.
2. The system will prompt for NRID/Birth Certificate Number of user to be deactivated.
3. The system will display User Name and NRID/Birth Certificate Number.

4. The System Administrator will press **Deactivate User** button. [E1-if confirmation denied]
5. The system will update personal record. [E2-if user has been deactivate]
6. The System Administrator can cancel the transaction at any time. [A1_user cancel transaction]

3.2.2.6 Alternate Flows

[A1_user cancel transaction]

1. Use case closes. System display menu.

3.2.2.7 Exception Flows

[E1_if confirmation denied]

1. System will prompt for new entry.

[E2-if user has been deactivate]

1. System will display error message showing that the user already deactivate

3.2.2.8 Post-Condition

User record updated.

3.2.2.9 Rule(s)

Not applicable

3.2.2.10 Constraint(s)

1. Time for showing message box should be done less than 2 seconds.

3.2.3 Use Case LOG IN [SRS_SES_300]

3.2.3.1 Actor

System Administrator, School Administrator and Student

3.2.3.2 Brief Description

This use case is used by System Administrator, School Administrator and Student to log in their web site.

3.2.3.3 Activation Condition

Event driven (On System Administrator, School Administrator and Student's demand)

3.2.3.4 Pre-Condition (s)

User Name and password exist.

3.2.3.5 Description (SRS_REQ_301)

1. This use case begins when user enter a Login request.
2. System will prompt for User Name and Password.
3. User press ENTER button to log in.
4. The system will check User Name and Password.
5. Display user menu. [E1- User Name and Password invalid]

3.2.3.6 Alternative Flows

No alternative flow.

3.2.3.7 Exception Flows

[E1- User Name and Password invalid]

1. The System shall show error message that User Name and Password incorrect and blank the text box.
2. The use case finish.

3.2.3.8 Post-Condition

User Menu is shown.

3.2.3.9 Rule(s)

Not applicable.

3.2.3.10 Constraint(s)

1. Time for showing message box should be done less than 2 sec.

3.2.4 Use case CHANGE PASSWORD [SRS_SES_400]

3.2.4.1 Actor

System Administrator, School Administrator, Student

3.2.4.2 Brief Description

This use case is used by System Administrator, School Administrator and Student to change their password.

3.2.4.3 Activation Condition

Event driven (On System Administrator, School Administrator and Student's demand)

3.2.4.4 Pre-Condition (s)

1. User Name and Password exist.
2. User has log in.

3.2.4.5 Description (SRS_REQ_401)

1. This use case begins when user press **Change Password** button.
2. The system display change password form.
3. The user input user name and the old password. This is followed by input new password.
5. User press ENTER button.
6. The system will validate the new input password. [E1- If password invalid]
7. System show message password changed.
8. The use case finish.

3.2.4.6 Alternative Flows

No alternative flow.

3.2.4.7 Exception Flows

[E1- If password invalid] [SRS REQ_402]

1. The System shall show error message that User Name or Password is incorrect and blank the text box.
2. The use case finish.

3.2.4.8 Post-Condition

Password record updated.

3.2.4.9 Rule(s)

Not applicable.

3.2.4.10 Constraint(s)

1. User ID is worker number.
2. Password is six alphanumeric

3.2.5 Use case Logout [SRS_SES_500]

3.2.5.1 Actor

System Administrator, School Administrator, and Student.

3.2.5.2 Brief Description

This use case is used by System Administrator, School Administrator and Student to log out their web site.

3.2.5.3 Activation Condition

Event driven (On System Administrator, School Administrator, and Student's demand)

3.2.5.4 Pre-Condition (S)

User Name and Password of System Administrator, School Administrator, and Student exist.

3.2.5.5 Description (SRS_REQ-501)

1. This use case begins when user press LOG OUT button to log out.
2. The system shall show message box to ensure that user really exit from system.
3. The system exit.
4. The use case finish.

3.2.5.6 Alternative Flows

Nil

3.2.5.7 Exception Flows

Nil

3.2.5.8 Post-Condition

User exit from system.

3.2.5.9 Rule(s)

Not applicable.

3.2.5.10 Constraint(s)

Nil

Below is the use case diagram for Student Enrollment Use Case Package. The use cases give an overall view of the package.

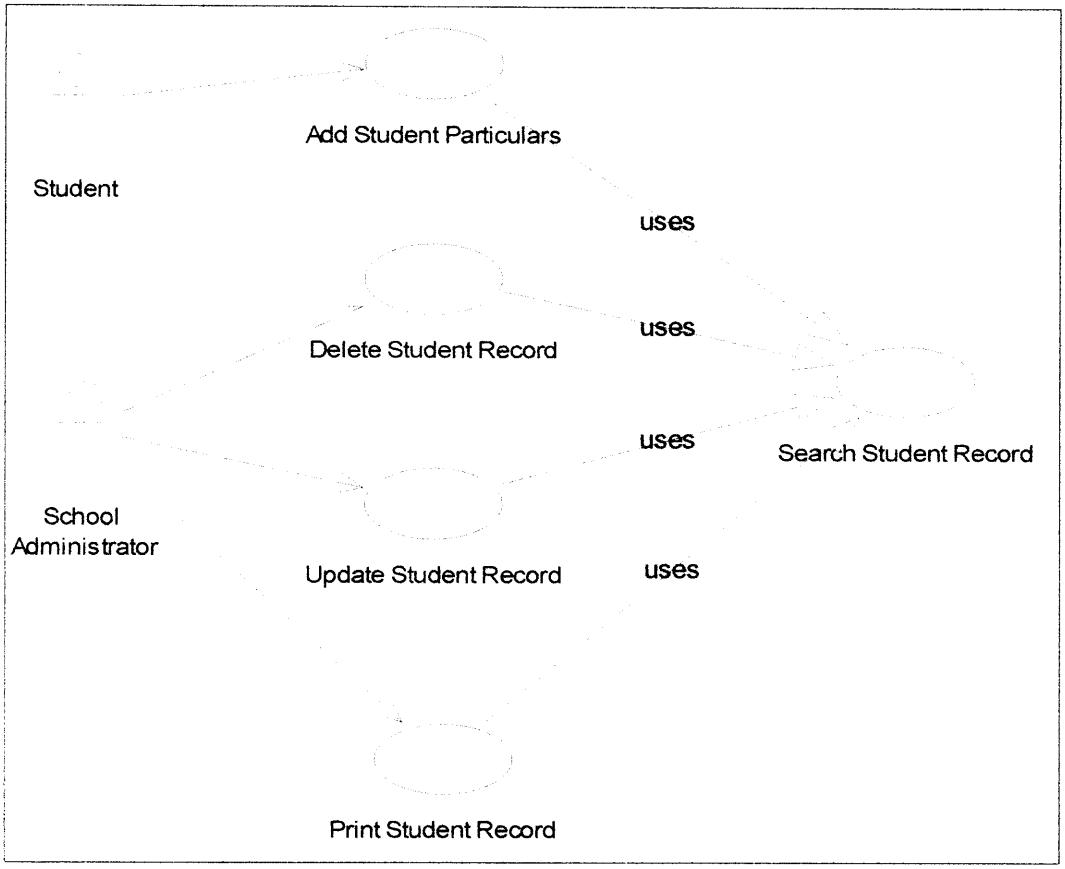


Figure 3.4 Use Case Diagram for Student Enrollment Use Case Package.

Use Case	Description
Add Student Particulars [SRS_SES_600]	New Student enters their personal particulars and saved as a new record.
Delete Student Record [SRS_SES_700]	When a new student fail to report physically after entering his/her particulars, the School Administrator may delete his/her record completely in the database.
Update Student Record [SRS_SES_800]	There is a need to edit and update a Student's record if there are changes in their particulars such as changes of address, phone number etc.
Print Student Record [SRS_SES_900]	The School Administrator may of various reason wish to print student record. The School Administrator can choose to print individual record, by intake or by class.

Table 2: Description Use Case Student Enrollment.

3.2.6 Use Case Add Student Particulars [SRS_SES_600]

3.2.6.1 Actor
Student

3.2.6.2 Brief Description
This use case is used by new Student. The new student fills up the particulars needed in enrolling himself/herself.

3.2.63 Activation Condition
Event driven (On Student's demand)

3.2.6.4 Pre-Condition (S)
1. Already log in the system.
2. NRID/Birth Certificate Number exists.

3.2.6.5 Description (SRS_SES_601)
1. This use case begins when the user enter a request to add personal particulars.
2 Then system displays an enrollment form. [E1-If record already exist]
3 After finished, system will save the record.
4 System will display progress bar while saving the record.

3.2.6.6 Alternative Flows
No alternative flow.

3.2.6.7 Exception Flows
[E1-If record already exist] [SRS_REQ_602]
1. System shall show message indicating that record of the Student already exists.
2. The use case closes.

3.2.6.8 Post-Condition
Database has changed.

3.2.6.9 Rule(s)
Not applicable.

3.2.6.10 Constraint(s)
1. Time for saving record should be done less than 1 minute.
2. Time for showing message box should be done less than 2 sec.

3.2.7 Use Case Delete Student Record [SRS_SES_700]

3.2.7.1 Actor
School Administrator

3.2.7.2 Brief Description
This use case used by School Administrator. The School Administrator can delete a Student record after viewing the record to be deleted.

3.2.7.3 Activation Condition
Event driven (On School Administrator's demand)

3.2.7.4 Pre-Condition (s)
The School Administrator has log into the system.

3.2.7.5 Description (SRS_REQ_501)
1. The use case begin when the school administrator enter a Delete Student Record request.
2. The use case will search for record to be deleted by using Search Student Record use case.
3. The system displays the Student Record and prompts for confirmation.
4. Administrator confirms the entry. [E1_if confirmation denied]
5. System show message that record has been deleted.
6. At any point, the administrator can cancel the transaction. [A1-Transaction Cancel]

3.2.7.6 Alternative Flows
[A1-Transaction Cancel]
1. Use case closes.

3.2.7.7 Exception Flows
[E1-if confirmation denied]
1. The system will prompt for new entry.

3.2.7.8 Post-Condition
Database had changed.

3.2.7.9 Rule(s)
Not applicable.

3.2.7.10 Constraint(s)
1. Time for delete the record should be done less than 1 minute.
2. Time for showing message box should be done less than 2 sec.

3.2.8 Use Case Update Student Record [SRS_SES_ 800]

3.2.8.1 Actor
School Administrator

3.2.8.2 Brief Description
This use case used by System Administrator. The System Administrator can edit and update a record for a Student.

3.2.8.3 Activation Condition
Event driven (On School Administrator's demand)

3.2.8.4 Pre-Condition (s)
The School Administrator has log into the system.

3.2.8.5 Description (SRS_ REQ_801)
1. This use case is initiated when the School Administrator enter an Update Sudent Record request.

2. The use case will search for record to be updated by using Search Student Record use case.
3. The system display full student record to be edited. [E1-Record Not Found]
4. The School Administrator clicks Update upon completion of edition.
5. The system prompts for another edit.
6. At any point, the administrator or form teacher can cancel the transaction. [A1-User Cancel Transaction]

3.2.8.6 Alternative Flows

[A1-User Cancel Transaction]

1. Use case closes. The system will resume to the main menu

3.2.8.7 Exception Flows

[E1-Record Not Found]

1. The system shows error message and prompt for new entry.

3.2.8.8 Post-Condition

Database had changed.

3.2.8.9 Rule(s)

Not applicable.

3.2.8.10 Constraint(s)

1. Time for update the record should be done less than 1 minute.
2. Time for showing message box should be done less than 2 sec.

3.2.9 Use Case Print Student Record [SRS_SES_900]

3.2.9.1 Actor

School Administrator

3.2.9.2 Brief Description

This use case used by School Administrator. The School Administrator can request the system to print individual Student Record, Student Record by Class or Student Records by Intake.

3.2.9.3 Activation Condition

Event driven (On School Administrator's demand)

3.2.9.4 Pre-Condition (s)

The School Administrator has already log in to the system.

3.2.9.5 Description (SRS_REQ_901)

1. This use case is initiated when the administrator enter a Print Student Record request.
2. The system will prompt for Intake Year, Class Name or NRID/Birth Certificate Number
3. Administrator enters the appropriate input.
4. The system prompts for confirmation to print. [E1_if entry not found]
5. Administrator confirms to print. [E2_if confirmation denied]
6. The system will print all records and prompt for next print request upon completion.
7. At any point before confirmation to print, the administrator can cancel the transaction. [A1-User Cancel Transaction]

3.2.9.6 Alternative Flows

[A1-User Cancel Transaction].

1. Use case closes. The system will resume to the main menu

3.2.9.7 Exception Flows

[E1_if entry not found].

1. The system will display error message and prompt for new entry.

[E2_if confirmation denied].

1. The system prompts for another print request.

3.2.9.8 Post-Condition
Record(s) printed.

3.2.9.9 Rule(s)
Not applicable.

3.2.9.10 Constraint(s)
1. Time for printing the report should be done less than 15 sec.

Below is the use case diagram for View Student Record Use Case Package. The use cases give an overall view of the package.

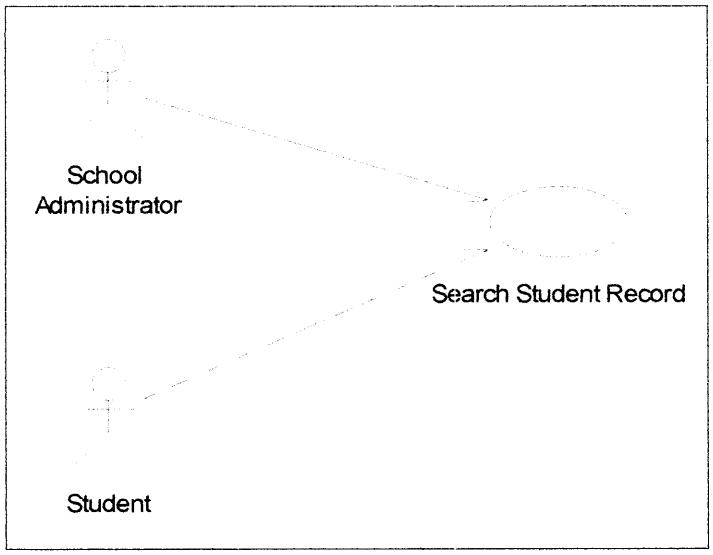


Figure 3.5 Use Case Diagram for Search Student Record Use Case Package.

Use Case	Description
Search Student Record [SRS_SES_1000]	When the School Administrator and students needs to view the Student Record, they may search the record by enter Student NRID or Birth Certificate Number. If the record exists, system will display the record on screen or otherwise shows error message.

Table 3: Description Use Case Search Student Record

3.2.10 Use Case Search Student Record [SRS_SES_1000]

3.2.10.1 Actor
School Administrator and Student

3.2.10.2 Brief Description
This use case is used by School Administrator and Students. They can search a Student Record by entering NRID/Birth Certificate Number. This is particularly to check the accuracy of documents (hard copy) submitted to be verified.

3.2.10.3 Activation Condition
Event driven (On System Administrator or Student's demand)

3.2.10.4 Pre-Condition (S)
System Administrator or Student has log into the system.

3.2.10.5 Description (SRS_REQ_1001)

1. This use case is initiated when the School Administrator or Student enters a Search Student Record request.
2. The system will prompt for student NRID/Birth Certificate Number.
3. The School Administrator or Student enters student NRID/Birth Certificate Number.
4. The system display full student record to be viewed. [E 1-If record does not exist]
5. The School Administrator or Student clicks Done upon completion of viewing.
6. The system prompts for another entry.
7. At any point, the administrator or form teacher can cancel the transaction. [A1_User Cancel Transaction]

3.2.10.6 Alternative Flows

[A1_User Cancel Transaction]

1. Use case closes. The system will resume to the main menu.

3.2.10.7 Exception Flows

[E1-if record does not exist] [SRS_REQ_1002]

1. System shall show message indicating that record of that staff does not exist.
2. The system will prompt for new entry.

3.2. 10.8 Post Condition

Nil

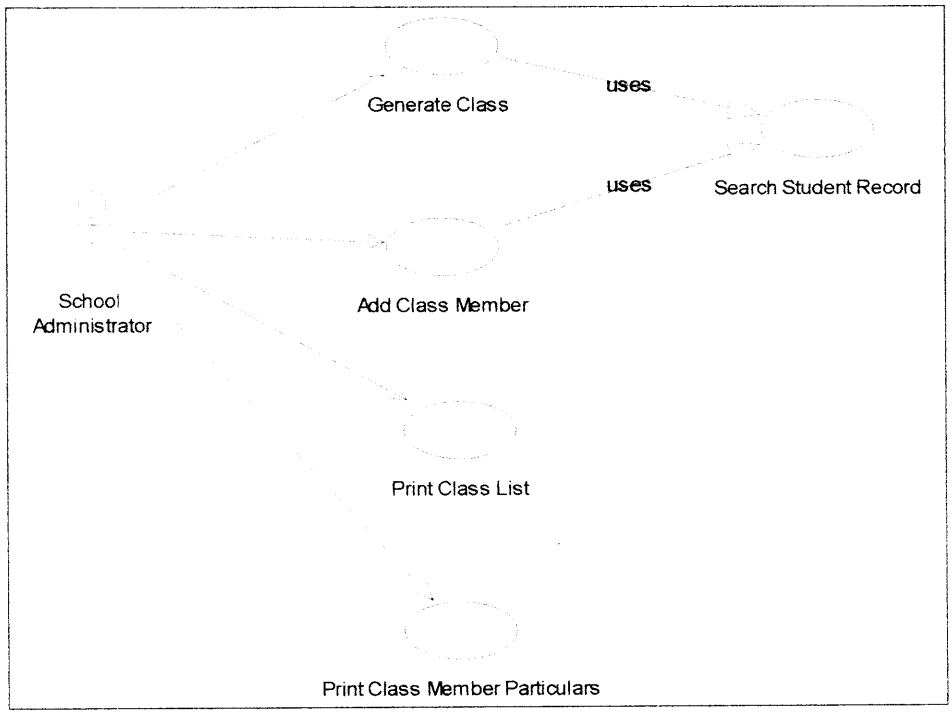
3.2.10.9 Rule(s)

Not applicable.

3.2.10.10 Constraint(s)

1. Time for search the record should be done less than 1 minute.
2. Time for showing message box should be done less than 2 sec.

Below is the use case diagram for Generate Attendance Roster Use Case Package. The use cases give an overall view of the package.



3.6 Use Case Diagram for Generate Attendance Roster Use Case Package.

Use Case	Description
Generate Class Member [SRS_SES_1100]	School Administrator includes class members to respective class.
Add Class Member [SRS_SES_1200]	School Administrator add new class member to class after the class member has been initially generated.
Print Name List [SRS_SES_1300]	School Administrator print class member list.
Print Member Particulars [SRS_SES_1400]	School Administrator prints the particulars of student in an academic class.

Table 4: Description Use Case Generate Attendance Roster.

3.2.11 Use Case Generate Class [SRS_SES_1100]

3.2.11.1 Actor
School Administrator

3.2.11.2 Brief Description
This use case is activated by the School Administrator to include class members for the new academic year.

3.2.11.3 Activation Condition
Event driven (On School Administrator's demand)

3.2.11.4 Pre-Condition (s)
System Administrator has log into the system.

3.2.11.5 Description (SRS_REQ_1101)
1. This use case is initiated when the School Administrator enter a Generate Class request.
2. The system will prompt for name of class to be created.
3. Administrator enter class name.
4. System confirms class created. [E1_ if class already exists]
4. The system will prompt for NRID/Birth Certificate Number.
5. The School Administrator enters NRID/Birth Certificate Number
6. The system displays the name Student Record and prompts for confirmation. [E2_if record not found]
7. The School administrator confirms the entry. The system will prompt for next entry. [E3_if confirmation denied]
8. At any point, the administrator can cancel the transaction. [A1_User Cancel Transaction]

3.2.11.6 Alternative Flows
[A1_User Cancel Transaction]
1. Use case closes. The system will resume to the main menu.

3.2.11.7 Exception Flows [SRS_REQ_1102]
[E1_if class already exist]
1. The system will prompt for new entry.
E2_if record not found].
1. The system will prompt for new entry.
[E3_if confirmation denied]
1. System prompts for reentry.

3.2. 11.8 Post Condition
Database has changed.

3.2.11.9 Rule(s)
Not applicable.

3.2.11.10 Constraint(s)

1. Time for search the record should be done less than 1 minute.
2. Time for showing message box should be done less than 2 sec.

3.2.12 Use Case Add Class Member [SRS_SES_1200]

3.2.12.1 Actor

School Administrator

3.2.12.2 Brief Description

This use case is activated by the School Administrator to add new members to existing class.

3.2.12.3 Activation Condition

Event driven (On School Administrator's demand)

3.2.12.4 Pre-Condition (S)

System Administrator has log into the system.

3.2.12.5 Description [SRS_REQ_1201]

1. This use case is initiated when the School Administrator enter an Add Class Member request.
2. The system will prompt for class name and year.
3. The School Administrator makes the required entries.
4. The system will prompt for NRID/Birth Certificate Number. [E1_if class not found]
5. The School Administrator enters NRID/Birth Certificate Number
6. The system displays the Student Record and prompts for confirmation. [E2_if record not found]
7. The School administrator confirms the entry. The system will prompt for next entry. [E3_if confirmation denied]
8. At any point, the administrator can cancel the transaction. [A1_User Cancel Transaction]

3.2.12.6 Alternative Flows [SRS_REQ_1202]

[A1_User Cancel Transaction]

1. Use case closes. The system will resume to the user menu.

3.2.12.7 Exception Flows

[E1_if class not found]

1. The use case closes, system display user menu.

[E2_if record not found].

1. The system will prompt for new entry.

[E3_if confirmation denied]

1. System prompts for new entry.

3.2.12.8 Post Condition

Database has changed.

3.2.12.9 Rule(s)

Not applicable.

3.2.12.10 Constraint(s)

1. Time for search the record should be done less than 1 minute.
2. Time for showing message box should be done less than 2 sec.

3.2.13 Use Case Print Class List [SRS_SES_1300]

3.2.13.1 Actor

School Administrator

3.2.13.2 Brief Description

This use case is activated by the School Administrator to print class name list.

3.2.13.3 Activation Condition
Event driven (On School Administrator's demand)

3.2.13.4 Pre-Condition (s)
System Administrator has log into the system.

3.2.13.5 Description [SRS_REQ_1301]
1. This use case is initiated when the School Administrator enter a Print Class List request.
2. The system will prompt for class name and year.
3. The School Administrator makes the required entries.
4. The system will prompt for confirmation to print. [E1_if class not found]
5. The system will print the class list [E2_if confirmation denied]
6. At any point before confirmation to print, the School Administrator can cancel the transaction. [A1_User Cancel Transaction]

3.2.13.6 Alternative Flows [SRS_REQ_1302]
[A1_User Cancel Transaction]
1. Use case closes. The system will resume to the main menu.

3.2.13.7 Exception Flows
[E1_if class not found]
1. The system will prompt for new entry.

[E2_if confirmation denied]
1. System prompts for new entry.

3.2.13.8 Post Condition
Class List printed.

3.2.13.9 Rule(s)
Not applicable.

3.2.13.10 Constraint(s)
1. Time for showing message box should be done less than 2 sec.

3.2.14 Use Case Print Class Member Particulars [SRS_SES_1400]

3.2.14.1 Actor
School Administrator

3.2.14.2 Brief Description
This use case is activated by the School Administrator to print student particulars.

3.2.14.3 Activation Condition
Event driven (On School Administrator's demand)

3.2.14.4 Pre-Condition (s)
System Administrator has log into the system.

3.2.14.5 Description [SRS_REQ_1401]
1. This use case is initiated when the School Administrator enter a Print Class Member Particular request.
2. The system will prompt for class name and year.
3. The School Administrator makes the required entries.
4. The system will prompt for confirmation to print. [E1_if class not found]
5. The system will print the Student Particulars [E2_if confirmation denied]
6. At any point before confirmation to print, the School Administrator can cancel the transaction. [A1_User Cancel Transaction]

3.2.14.6 Alternative Flows [SRS_REQ_1402]

[A1_User Cancel Transaction]

1. Use case closes. The system will resume to the main menu.

3.2.14.7 Exception Flows

[E1_if class not found]

1. The system will prompt for new entry.

[E2_if confirmation denied]

1. System prompts for reentry.

3.2.14.8 Post Condition

Class List printed.

3.2.14.9 Rule(s)

Not applicable.

3.2.14.10 Constraint(s)

1. Time for showing message box should be done less than 2 sec.

3.3 System Internal Requirements

SES has eight classes in Class Diagram. There is System Panel class, AdminMgr class, EnrollmentMgr class, ClassMgr class, SearchMgr class, User class, StudentRec class, and ClassRec class. Refer APPENDIX C (Class Diagram)

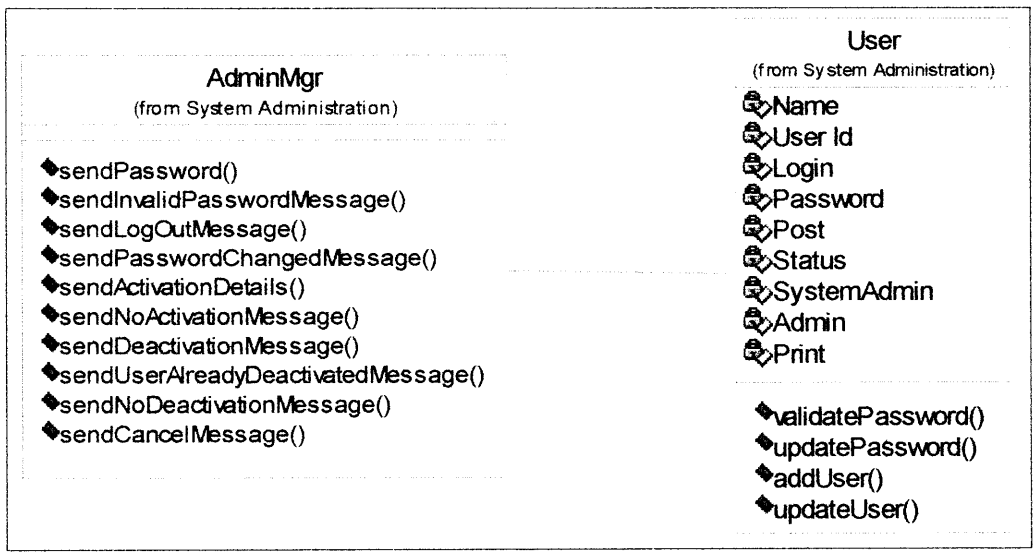


Figure 3.7 : Class Diagram Package for Use Case System Administration

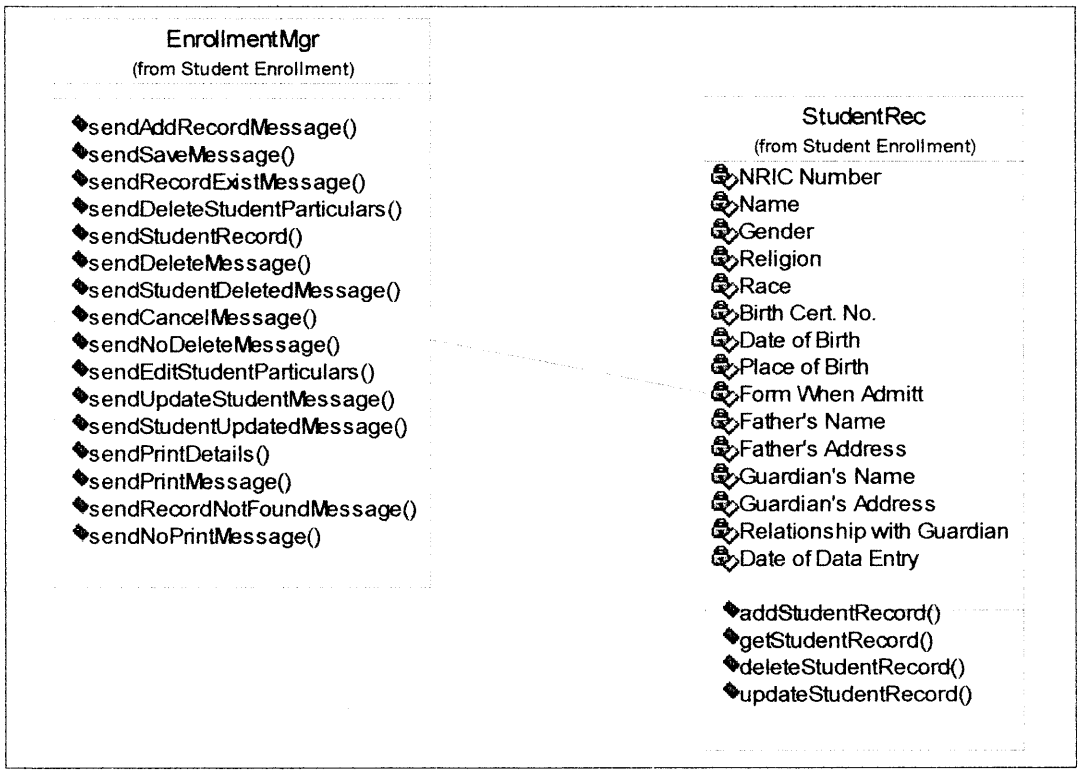


Figure 3.8: Class Diagram Package Use Case Student Enrollment

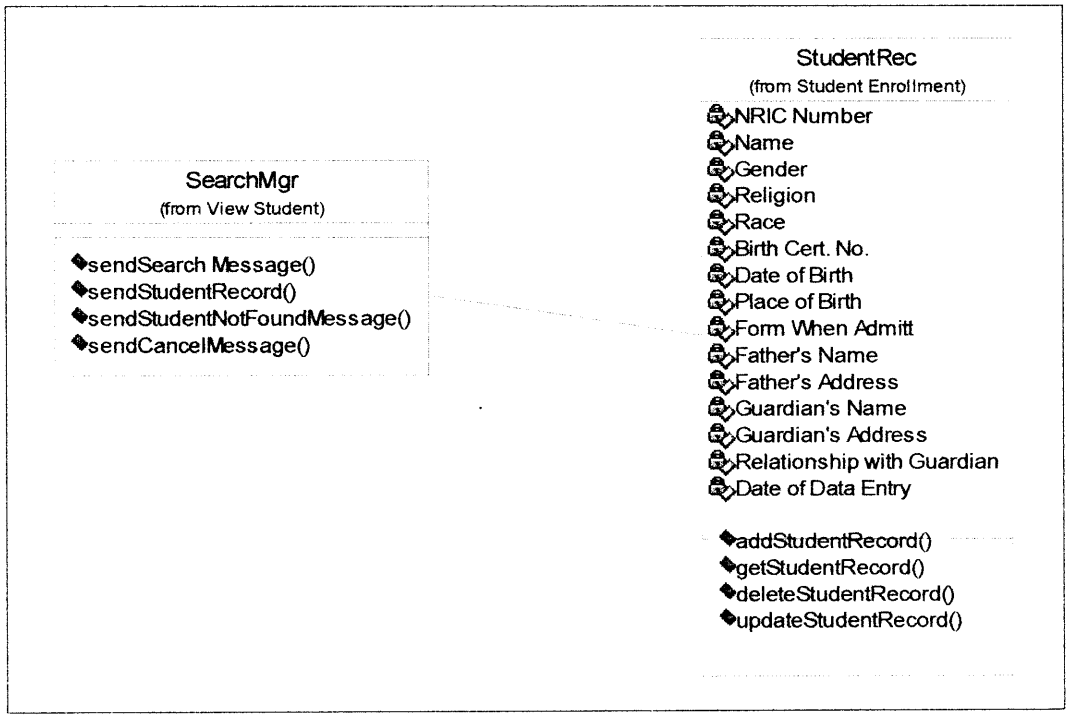


Figure 3.9: Class Diagram Package Use Case View Student Record

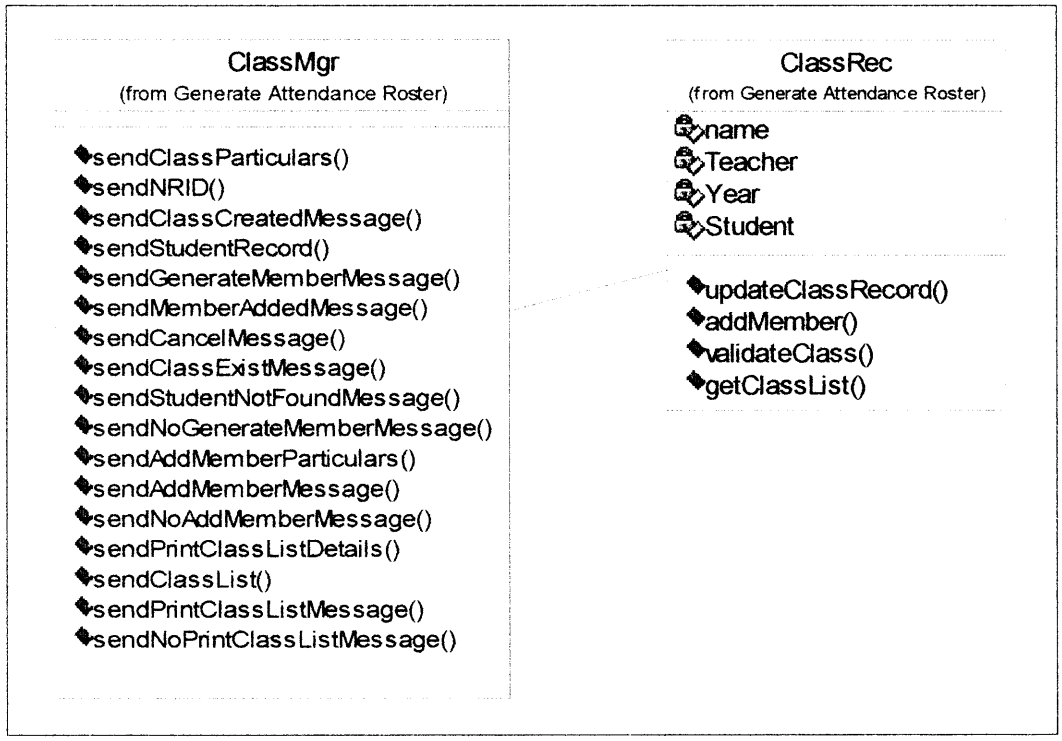


Figure 3.10: Class Diagram Package Use Case Generate Attendance Roster

3.4 System Data Elements Requirements

Please refer APPENDIX A (Sequence Diagram)

3.4.1 System Panel Class

It implements the use case Log In .
Received Events: displayLoginForm() from AdminMgr
Sent Events: sendUserName AndPassword () to User
Attributes: to be defined in SDD

3.4.2 AdminMgr class

It implements the use case Activate User.
Received Events: sendActivationDetails () from System Panel
Sent Events: addUser() to User
Attributes : to be defined in SDD

3.4.3 EnrollmentMgr class

It implements the use case Update Student Record.
Received Events: sendUpdateStudentMessage() from System Panel
Sent Events: updateStudentRecord to StudentRec
Attributes: to be defined in SDD

3.4.4 ClassMgr

It implements the use case Generate Class.
Received Events: sendClassParticulars() from System Panel
Sent Events: addClassRecord() to ClassRec
Attributes: to be defined in SOD

3.4.5 SearchMgr

It implements the use case Search Student Record.
Received Events: sendSearchMessage() from System Panel
Sent Events: getStudentRecord() to StudentRec
Attributes: to be defined in SOD

3.4.6 User Class

It implements the use case Activate User.

Received Events: addUser() from AdminMgr
Sent Events: sendUserAddedMessage() to AdminMgr
Attributes: to be defined in SDD

3.4.7 StudentRec Class

It implements the use case updateStudent Record.

Received Events: updateStudentRecord() from EnrollmentMgr
Sent Events: sendStudentUpdatedMessage () to EnrollmentMgr
Attributes: to be defined in SOD

3.4.8 ClassRec Class

It implements the use case Generate Class.

Received Events: addClassRecord() from ClassMgr
Sent Events: sendClassCreatedMessage() to ClassMgr
Attributes: to be defined in SOD

3.5 Sizing and Timing Requirements

3.5.1 Timing

1. Time for saving record should be done less than 1 minute.
2. Time for showing message box should be done less than 2 sec.

3.5.2 Sizing

The size of project is approximately 10000 lines of codes.

3.6 Security Requirements

1. User number as staff number and password needed to log in the system. It is a private and confidential item. All of the changes and application can only be done by the appropriate person

3.7 Design Constraints

3.7.1 Programming Language

1. Visual Basic
2. ASP

3.7.2 Programming Practices

During the programming activities, all developers will be following the Visual Basic Answers, Tech Support At Your Fingertips. (McGraw Hill) book.

3.8 Software Quality Factors

The evaluation of document quality and development process quality will be accordance with the criteria DoD-Std-2167A, DoD-Std-2168 and parts of Rational conventions are the responsibility of the ordering agency.

3.8.1 Internal Consistency

The software must be comply and consistent with its state transition diagram, class diagram and sequence diagram upon with the specifications and meet the user's requirements.

3.8.2 Understandability

The software must be readable and understandable by any software quality manager or user.

3.8.3 Testability

The software must be testable by any test engineer or user.

3.8.4 Traceability
The software must be traceable in between the user's requirements and its allocation reference.

3.8.5 Consistency
The software must be comply and consistent with SRS, SDD, STD, STR.

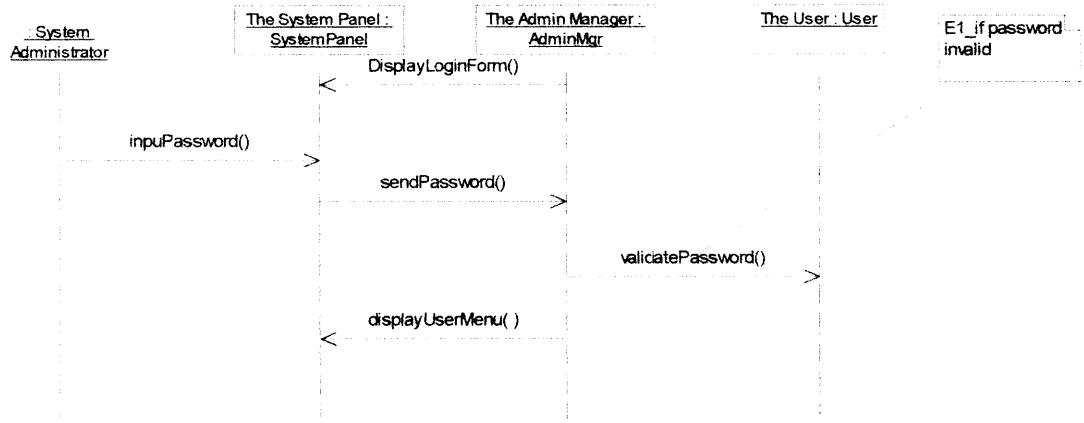
4 PREPARATION FOR DELIVERY
The software shall be delivered in a 650 megabyte optical disk and all the documentations shall be kept accordingly in a hard cover file.

5 NOTES
SRS Software Requirement Specification
REQ Requirement
SES Student Enrollment System

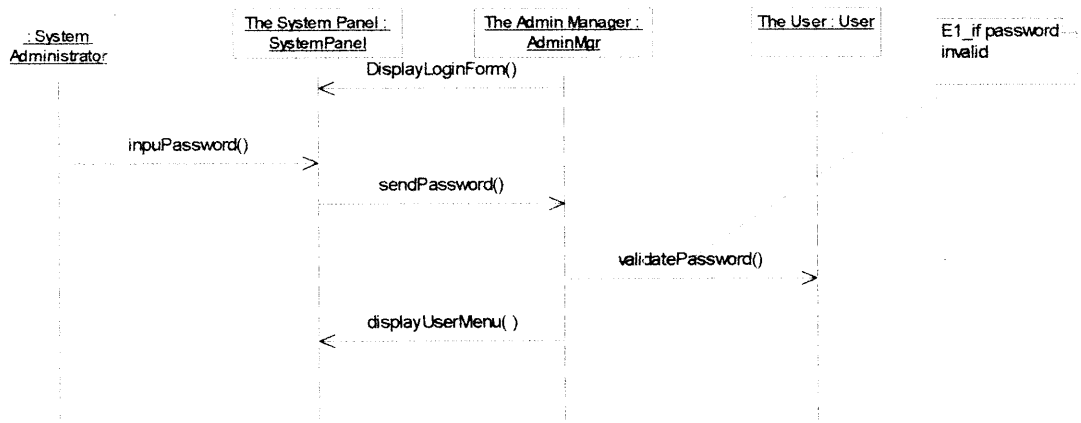
Appendix A:

Sequence Diagrams

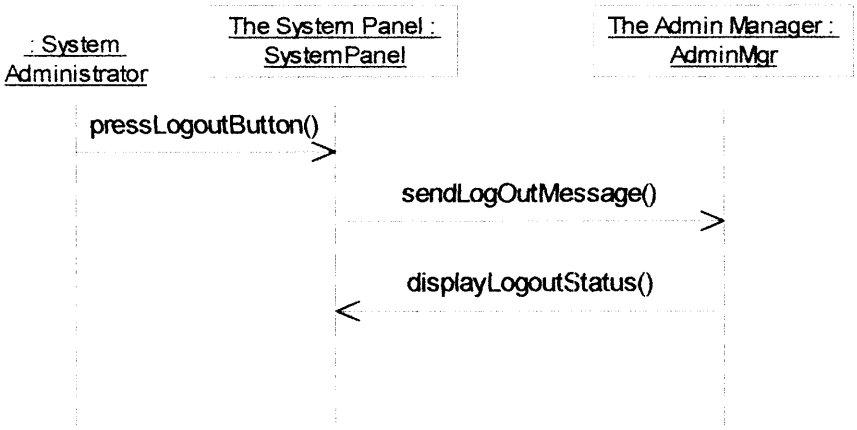
System Administration Package Sequence Diagrams



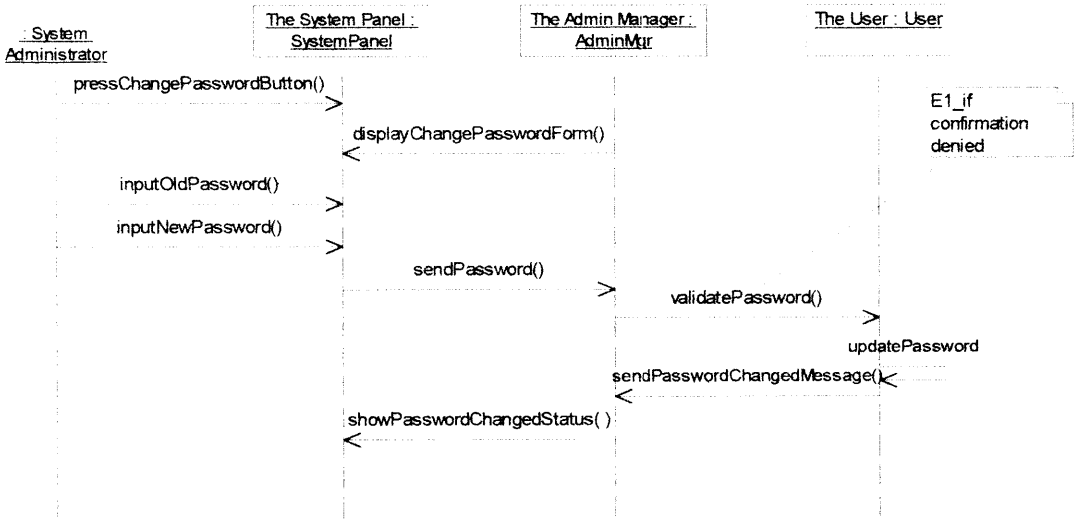
Log in System



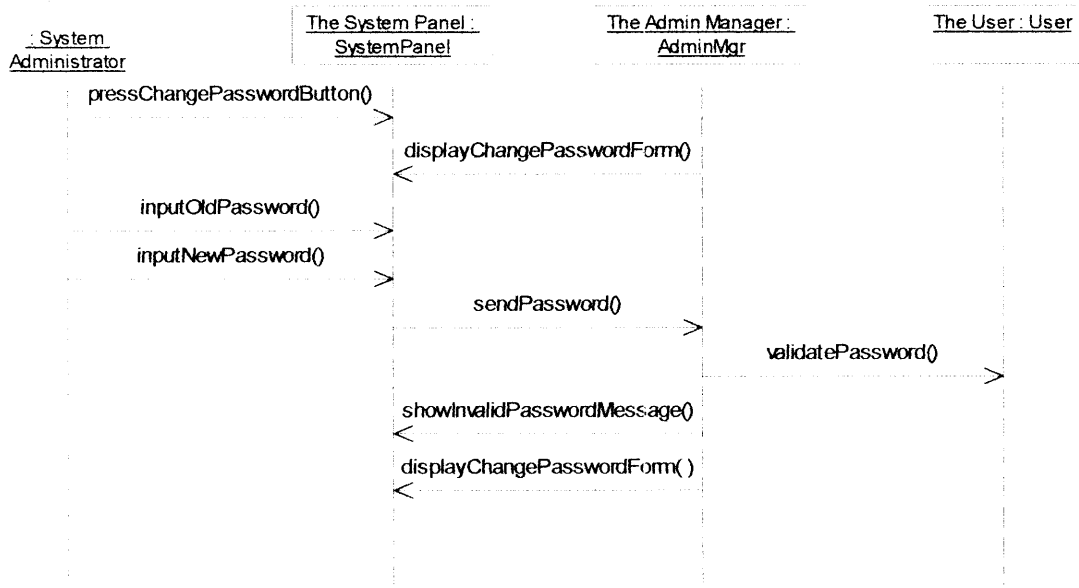
Log In System/E1_if Password Invalid



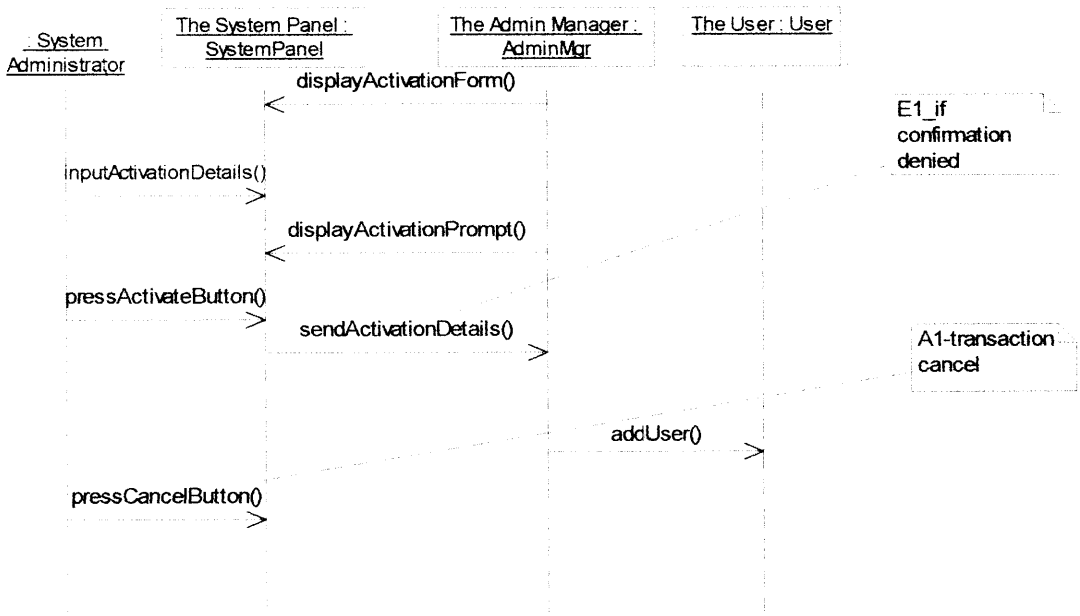
Log Out System



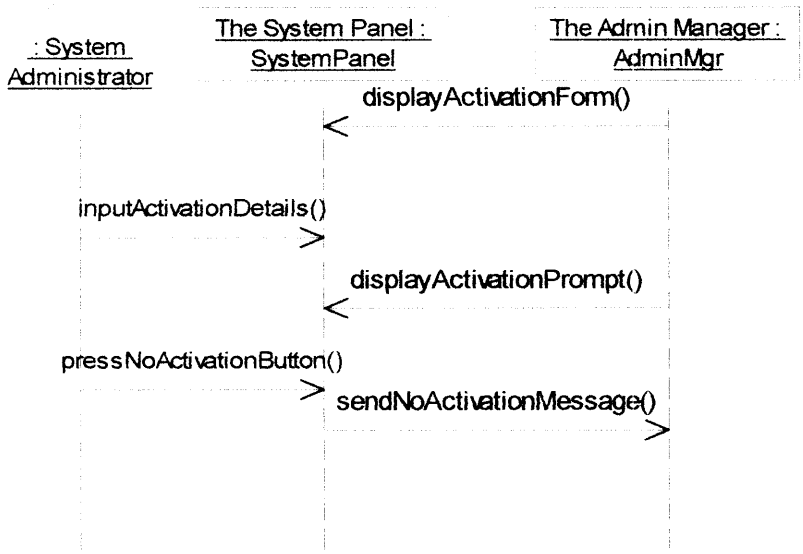
Change Password



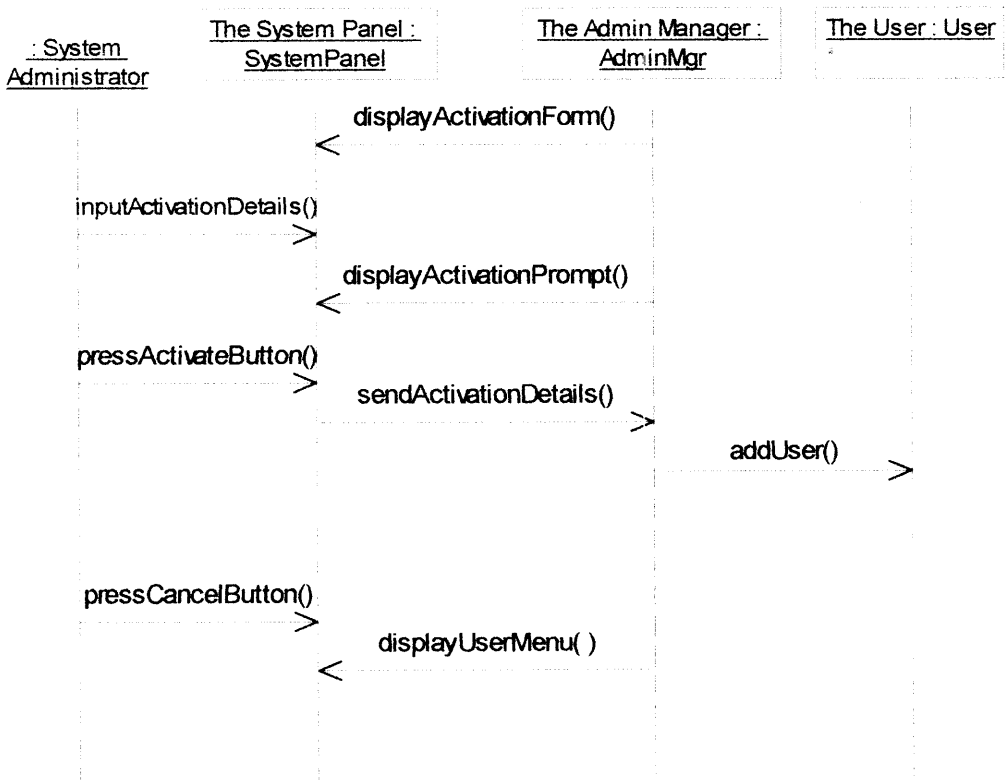
Change Password/E1_if password in valid



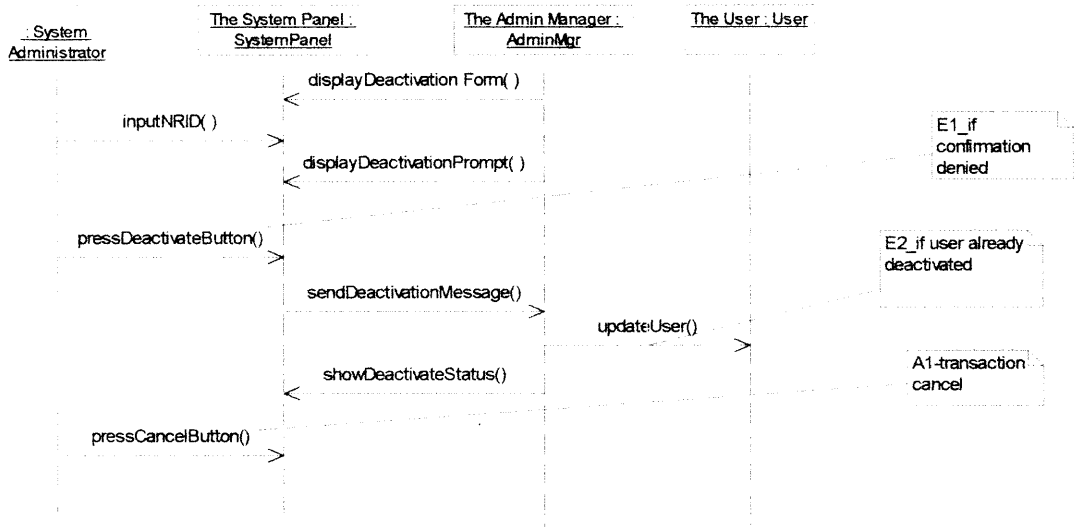
Activate User



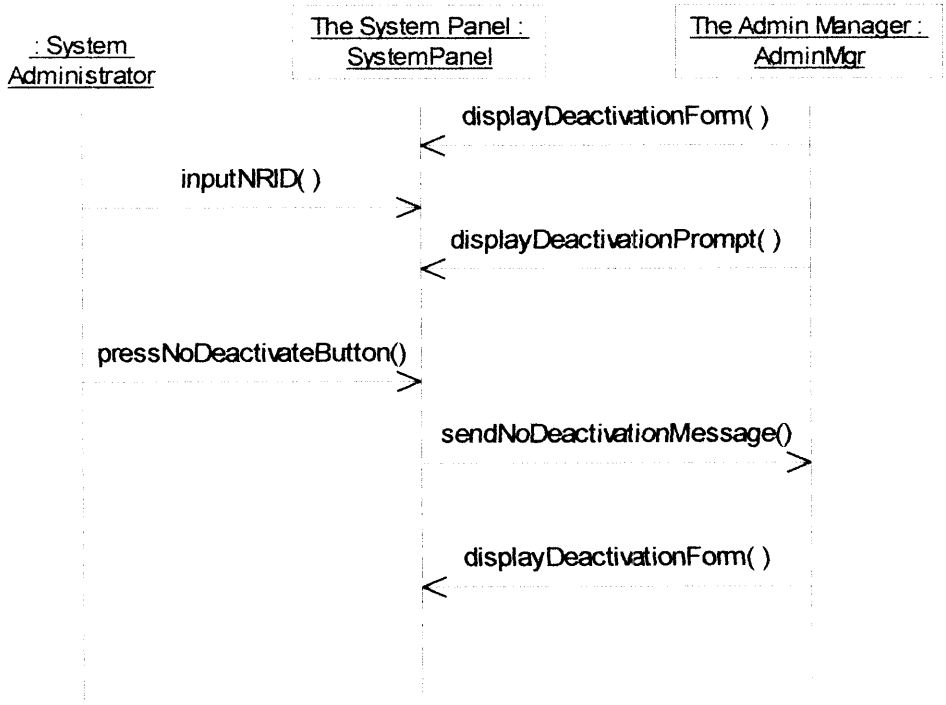
Activate User/E1_Activation Not Confirm



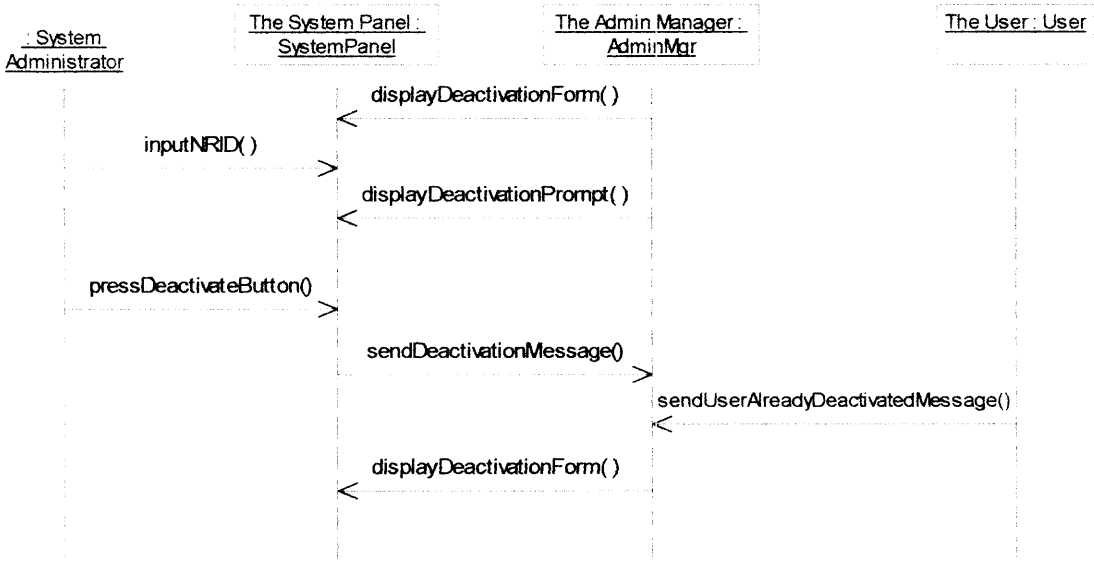
Activate User/A1_transaction cancel



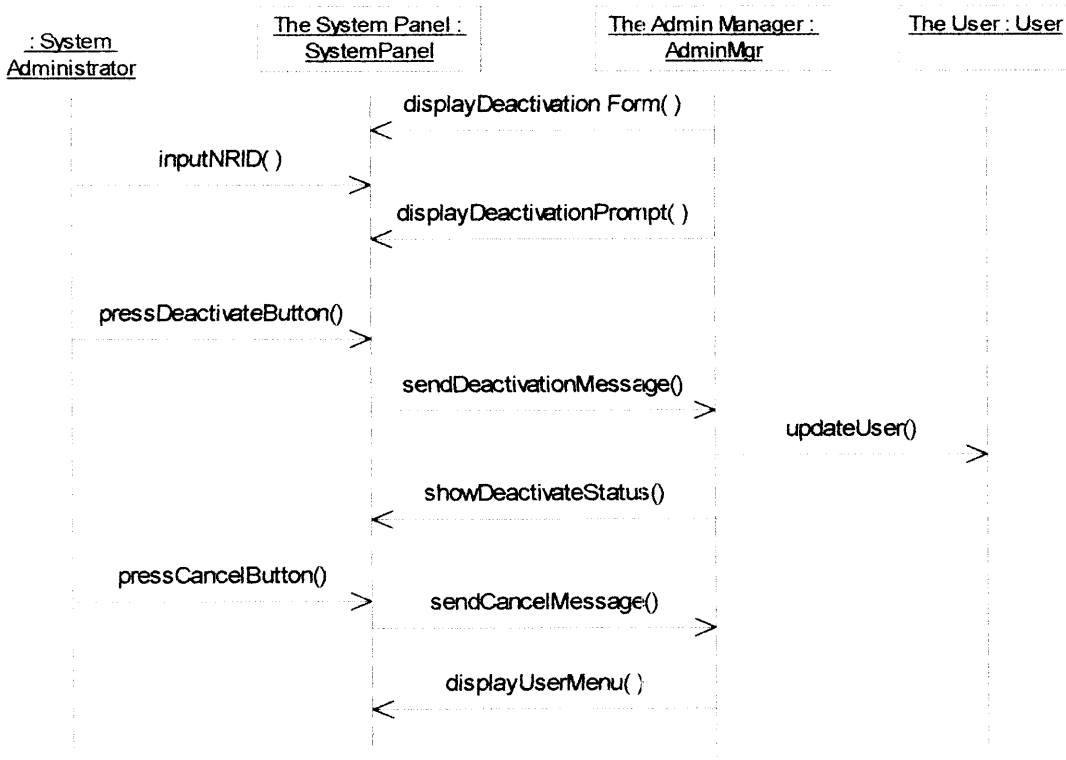
Deactivate User



Deactivate User/E1_if confirmation denied.

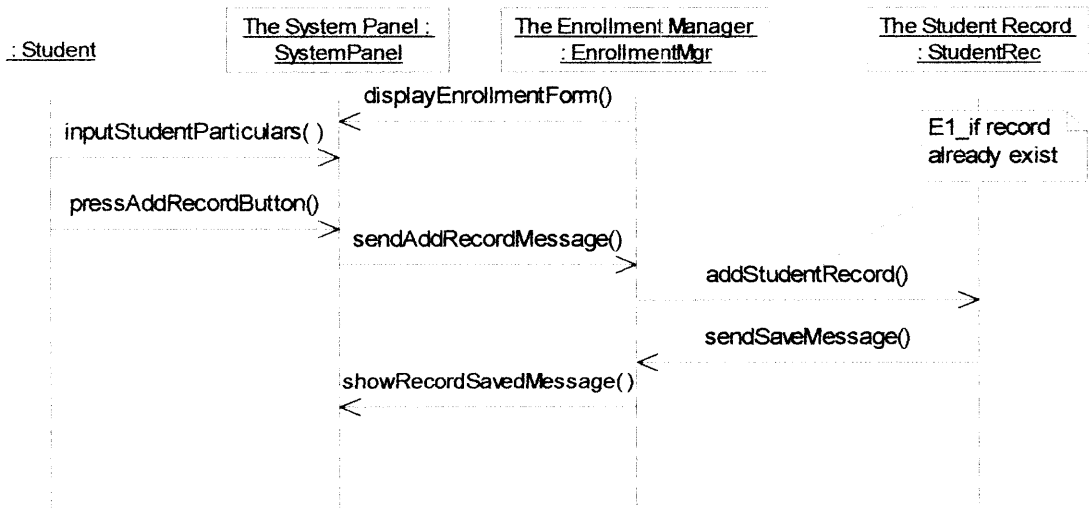


Deactivate User/E2_if user already deactivated

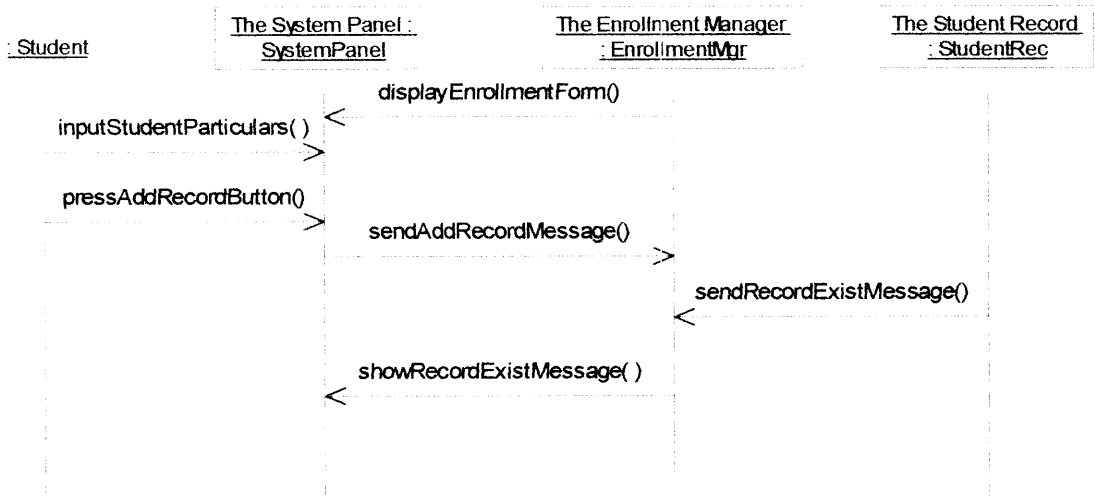


Deactivate User/A1_user cancel transaction

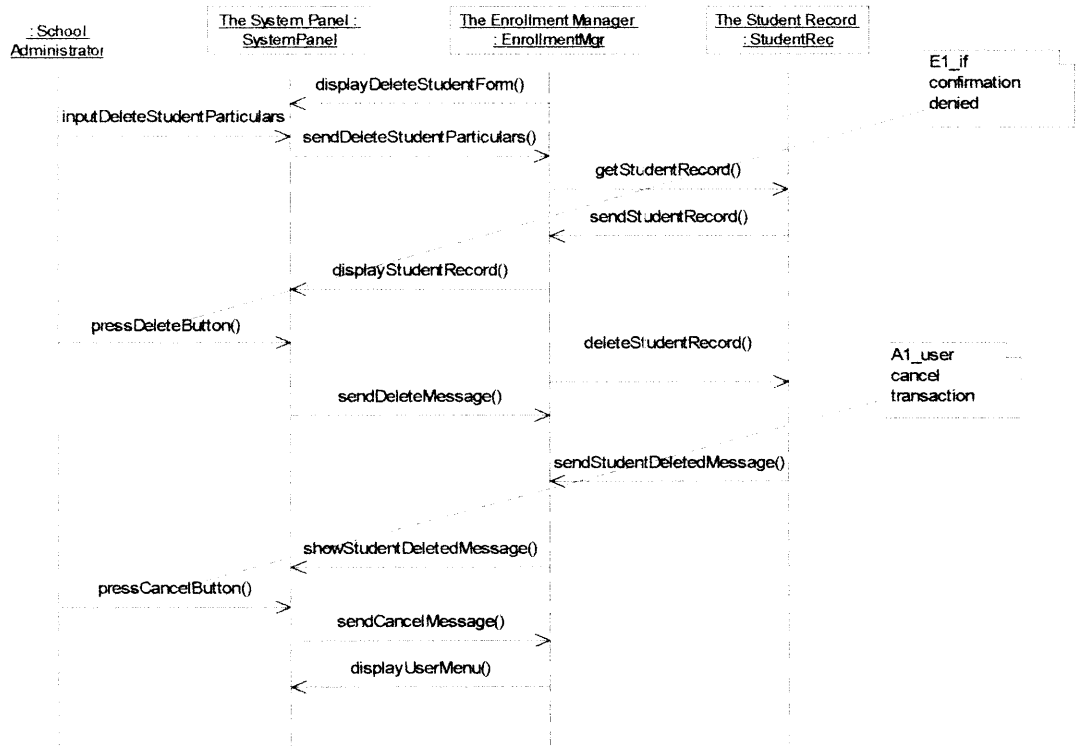
Student Enrollment Package Sequence Diagram



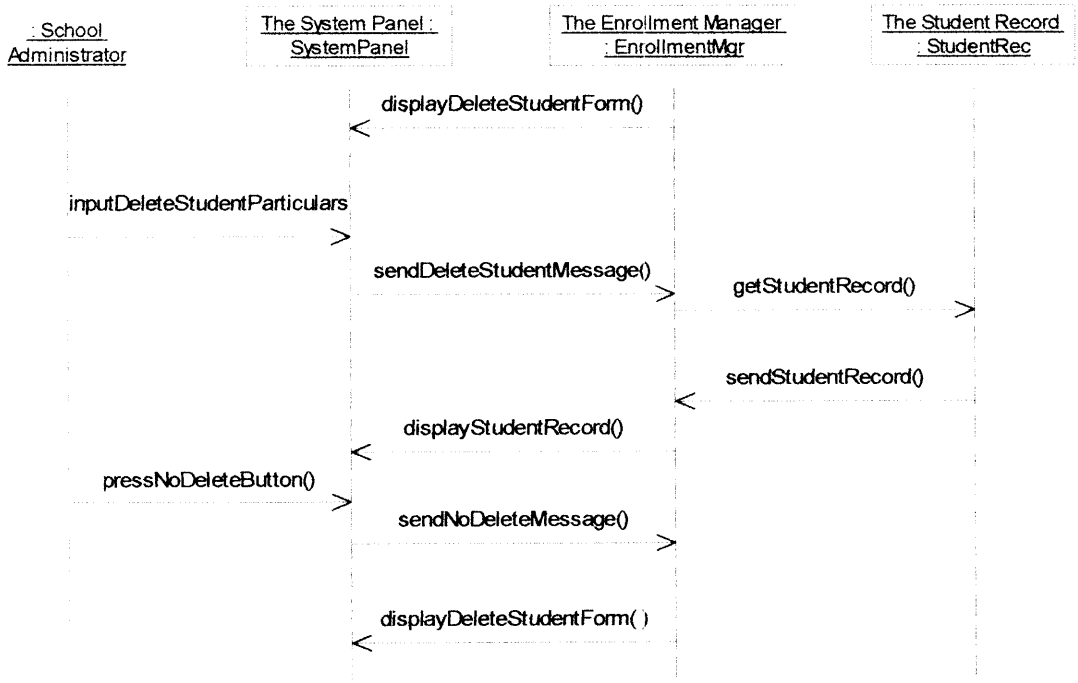
Add Student Particulars



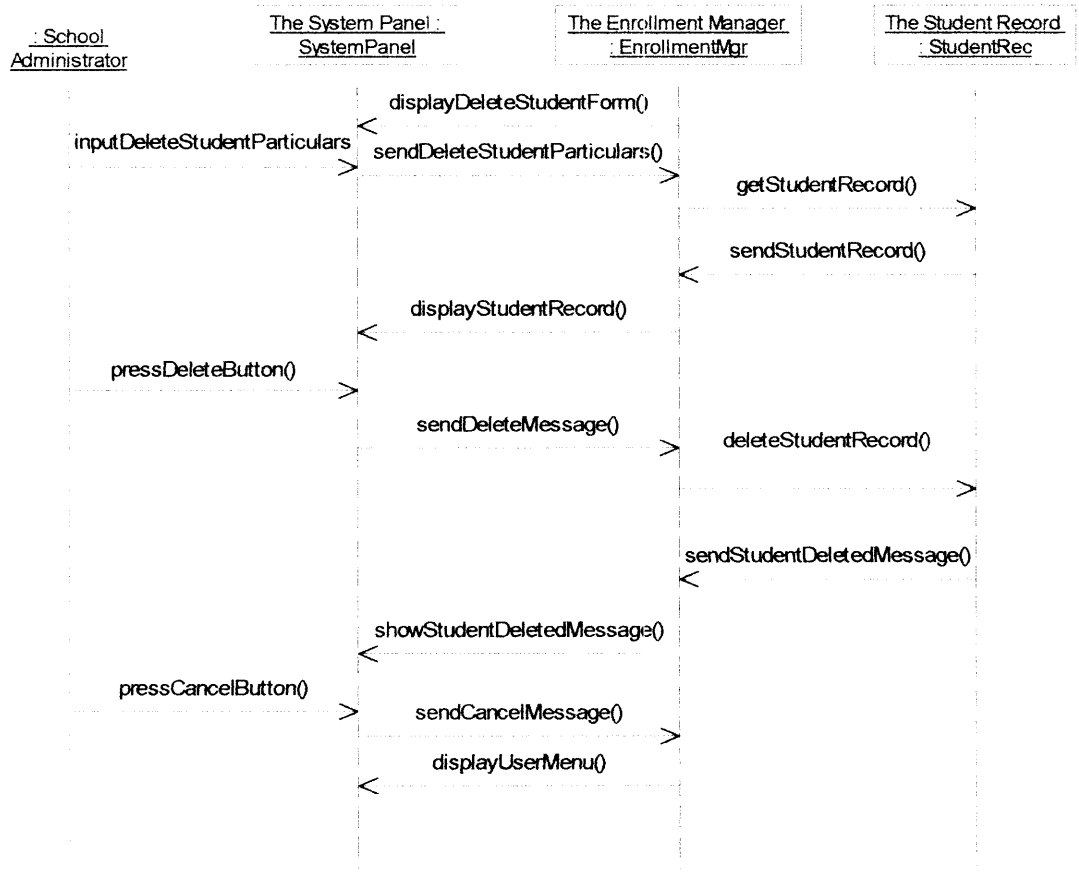
Add Student Particulars/E1_if record already exist.



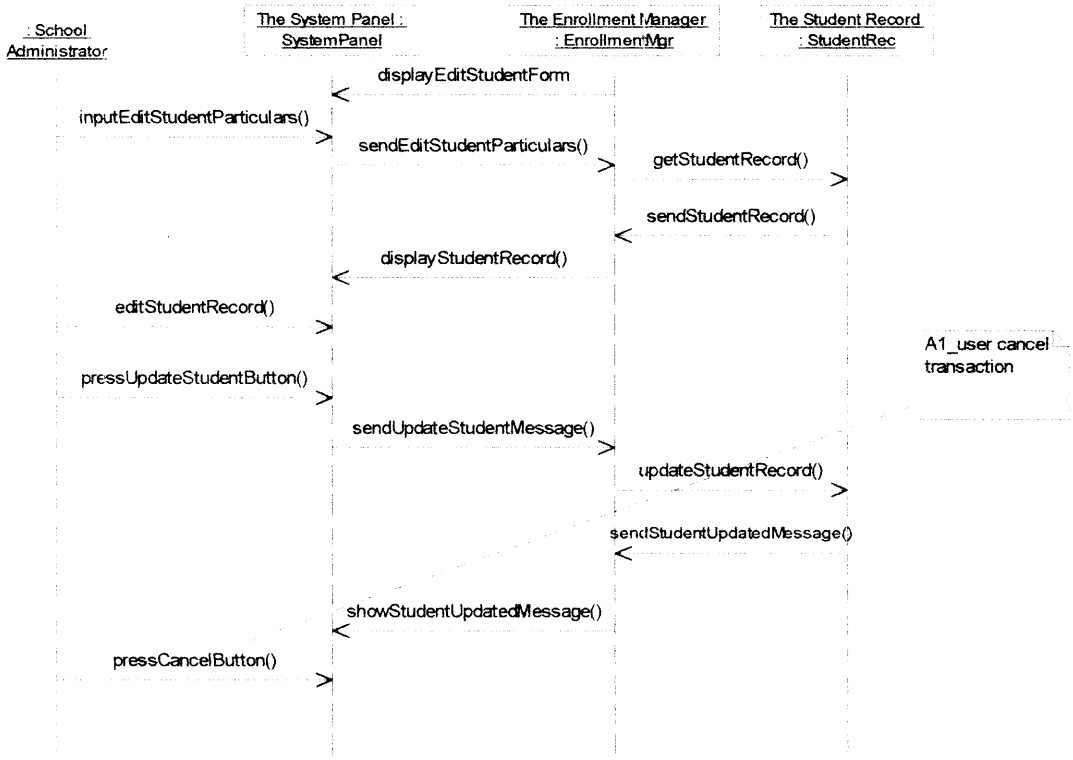
Delete Student Record



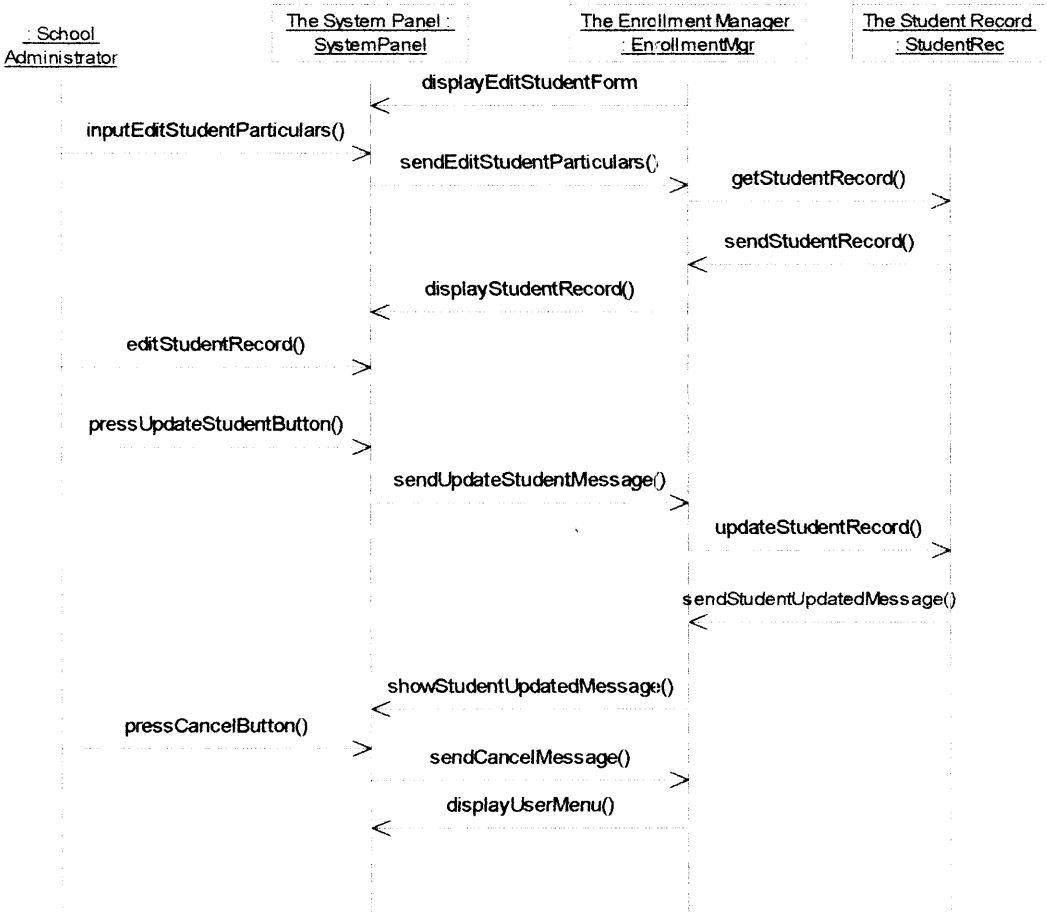
Delete Student Record/E1_confirmation denied



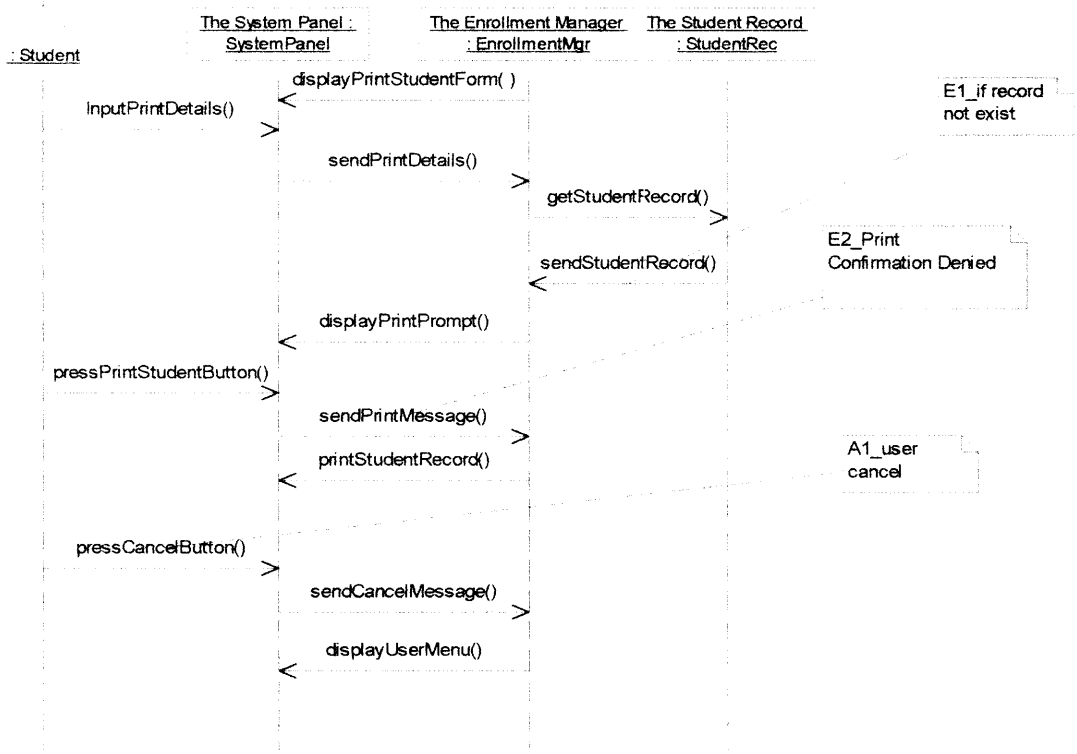
Delete Student Record/A1_user cancel transaction



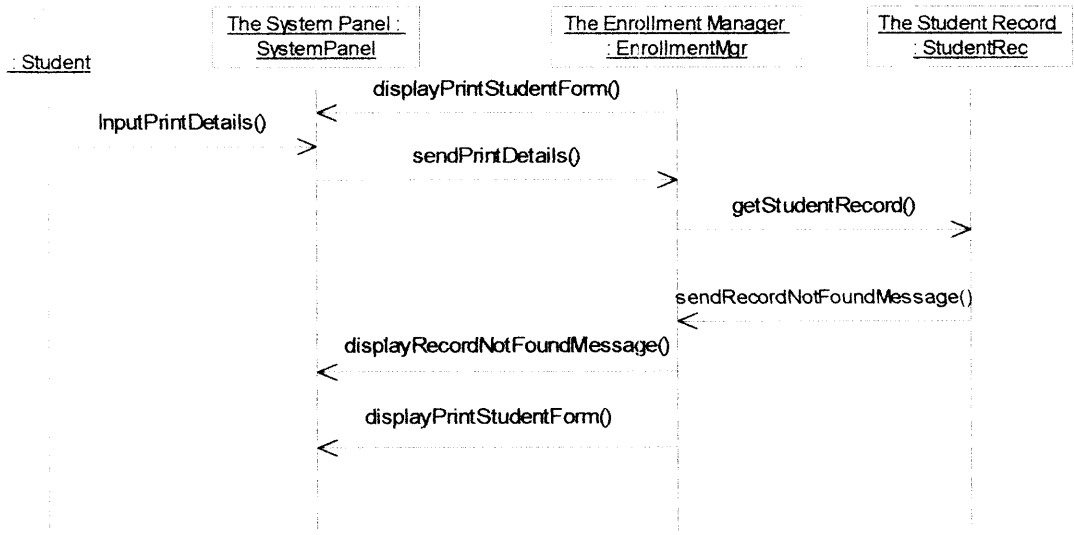
Update Student Record



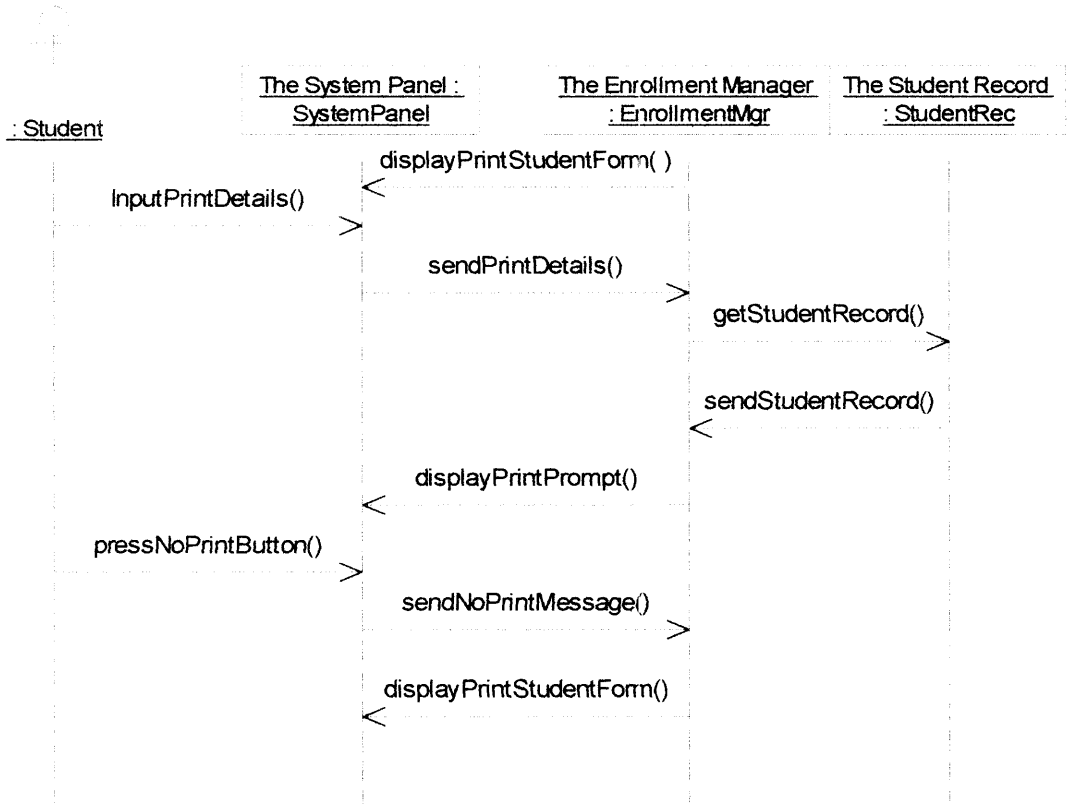
Update Student Record/A1_user cancel transaction



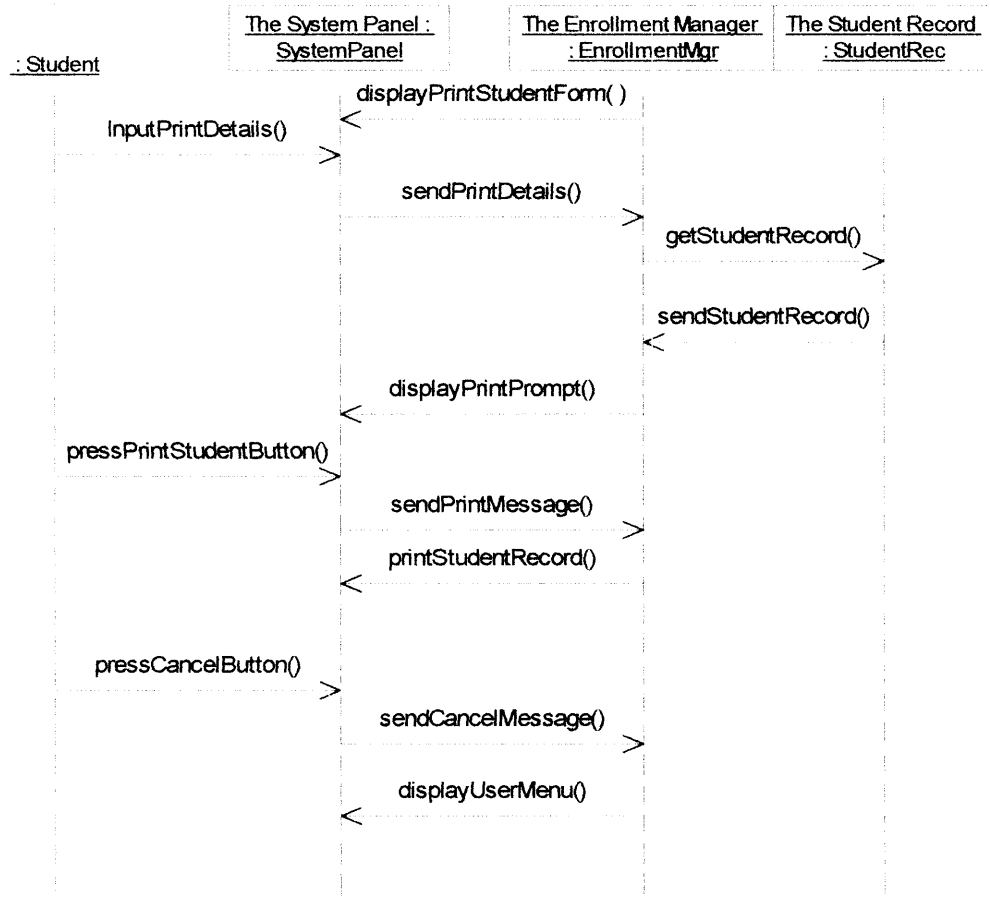
Print Student Record



Print Student Record/E1_if record not found.

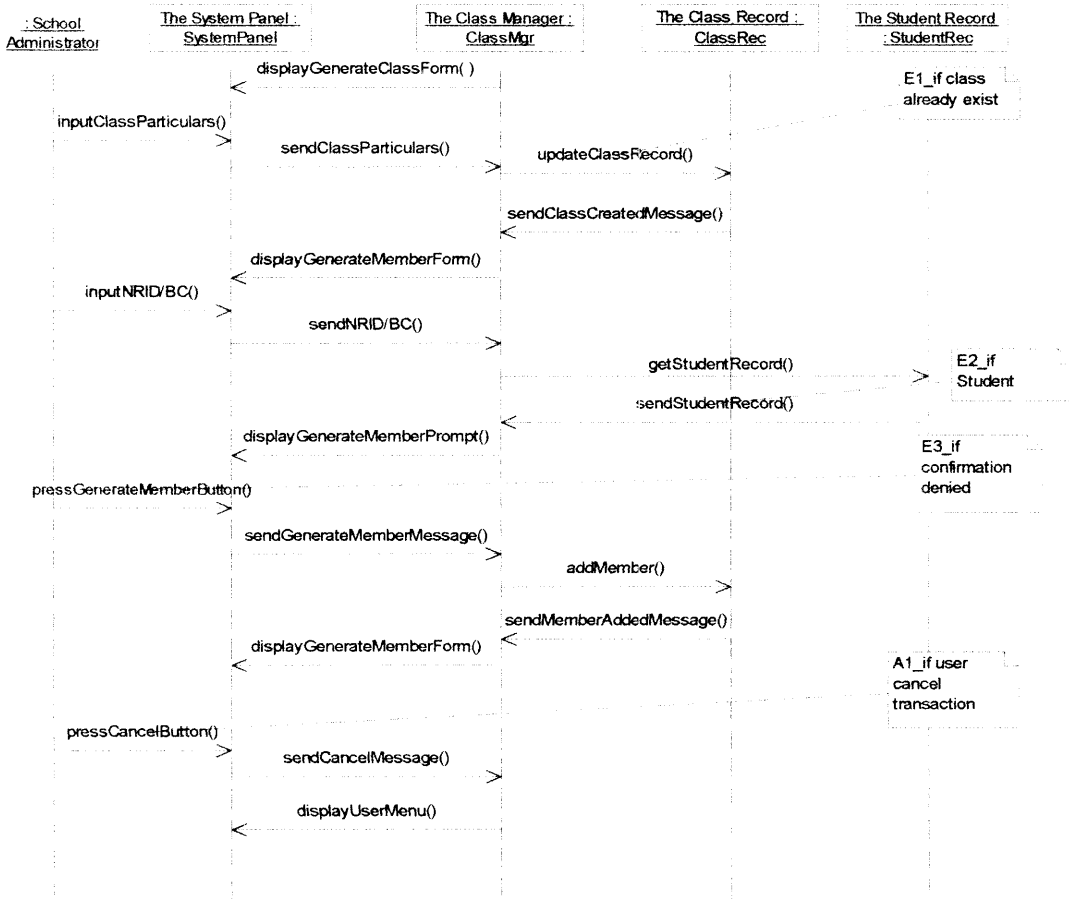


Print Student Record/E2_confirmation denied.

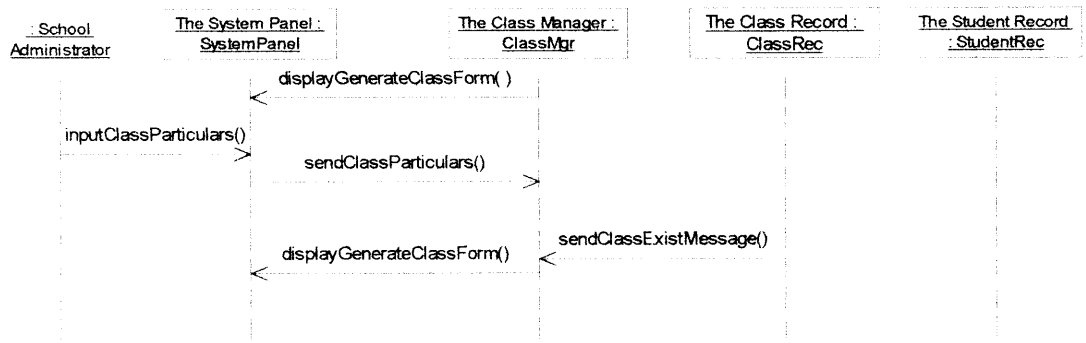


Print Student Record/A1_user cancels transaction

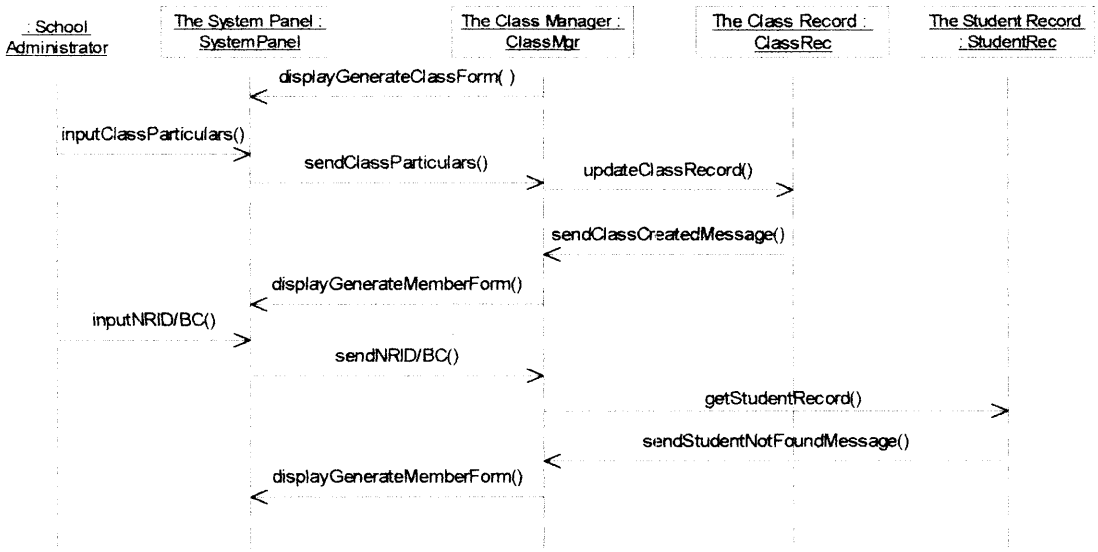
Generate Attendance Roster Sequence Diagram



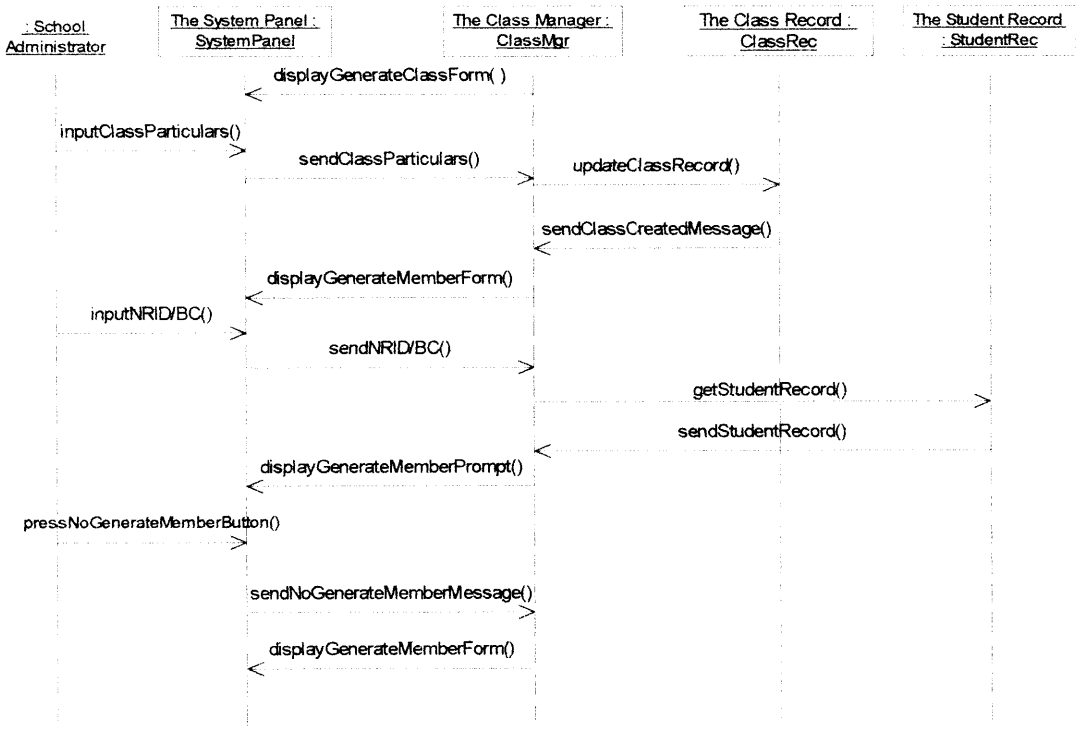
Generate Class



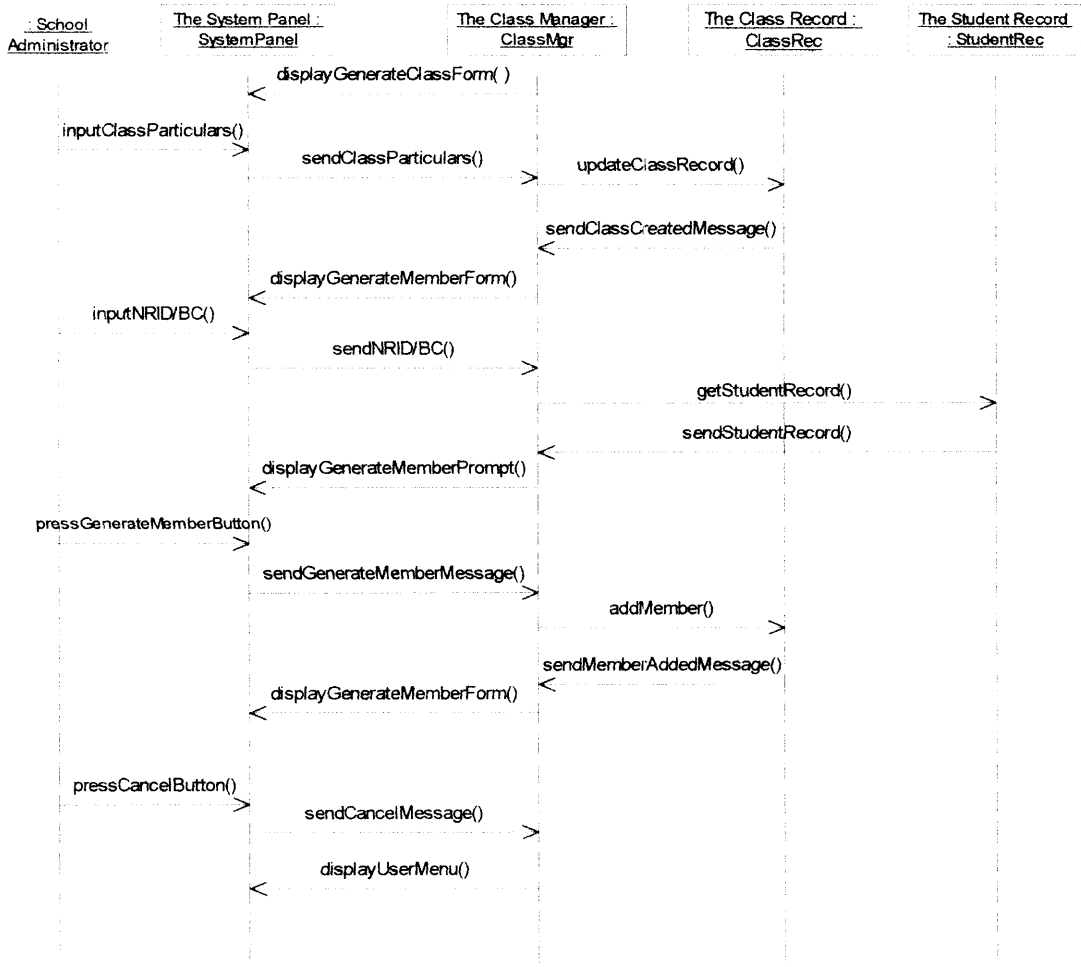
Generate Class/E1_if class already exist



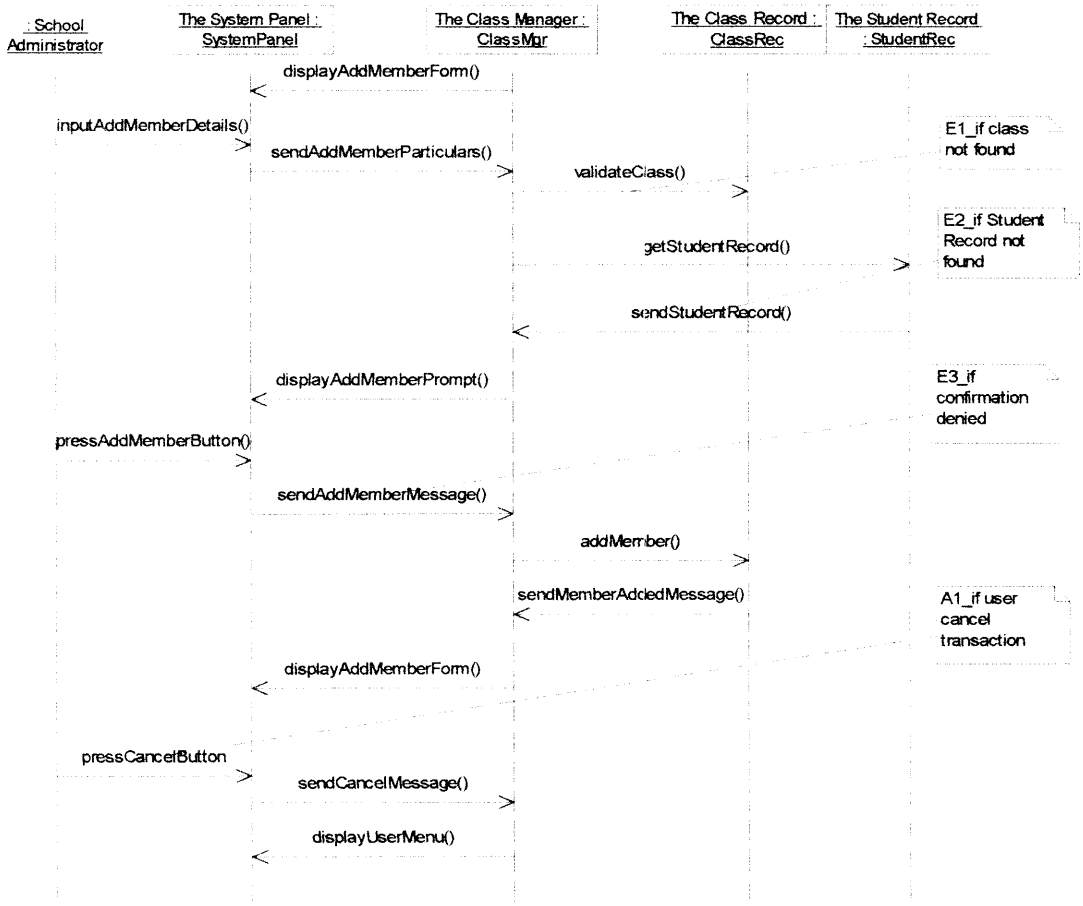
Generate Class/E2_if student record not found.



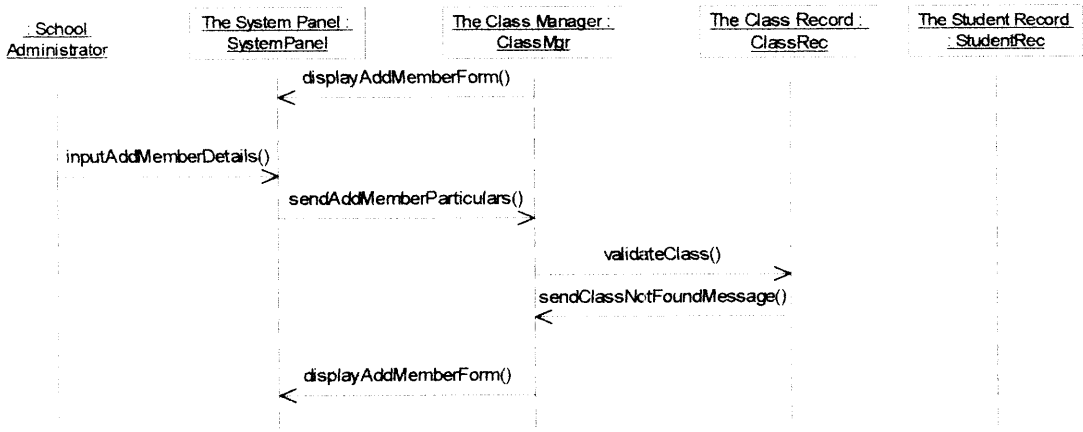
Generate Class/E3_if confirmation denied



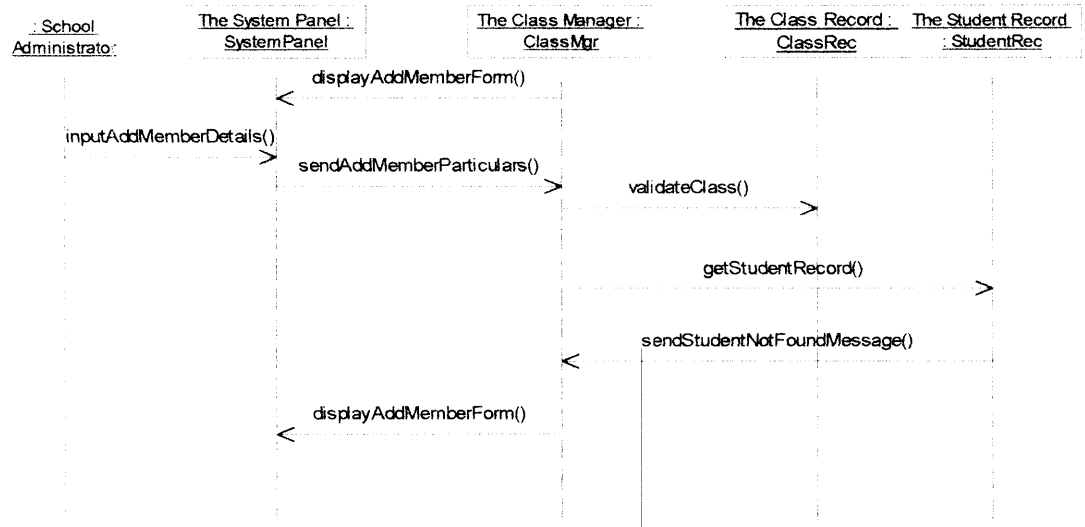
Generate Class/A1_user cancel transaction



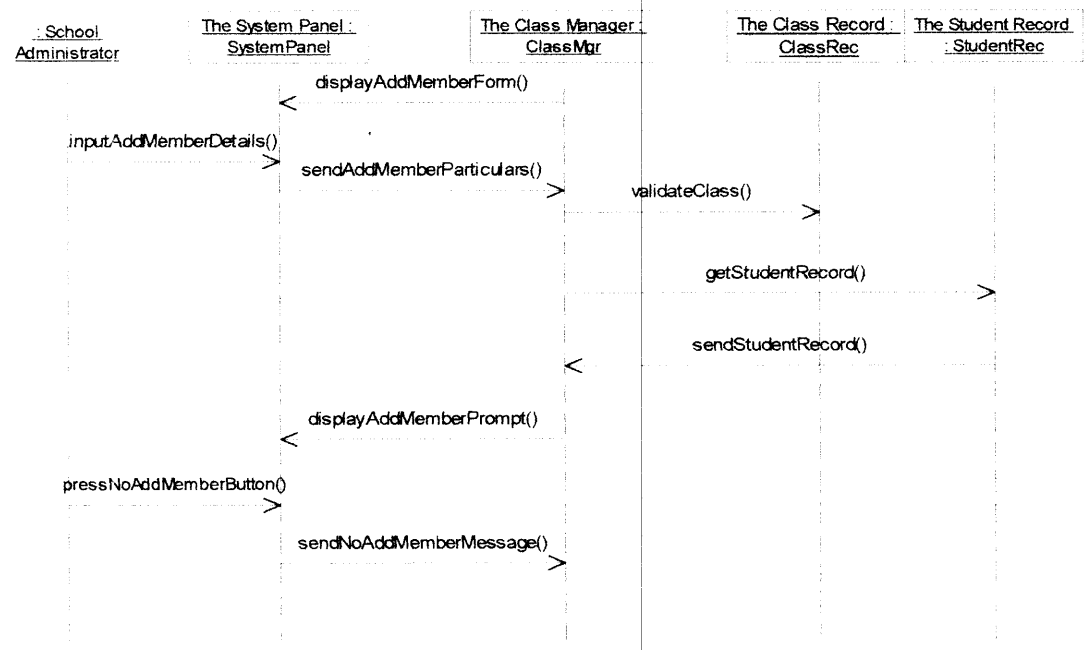
Add Class Member



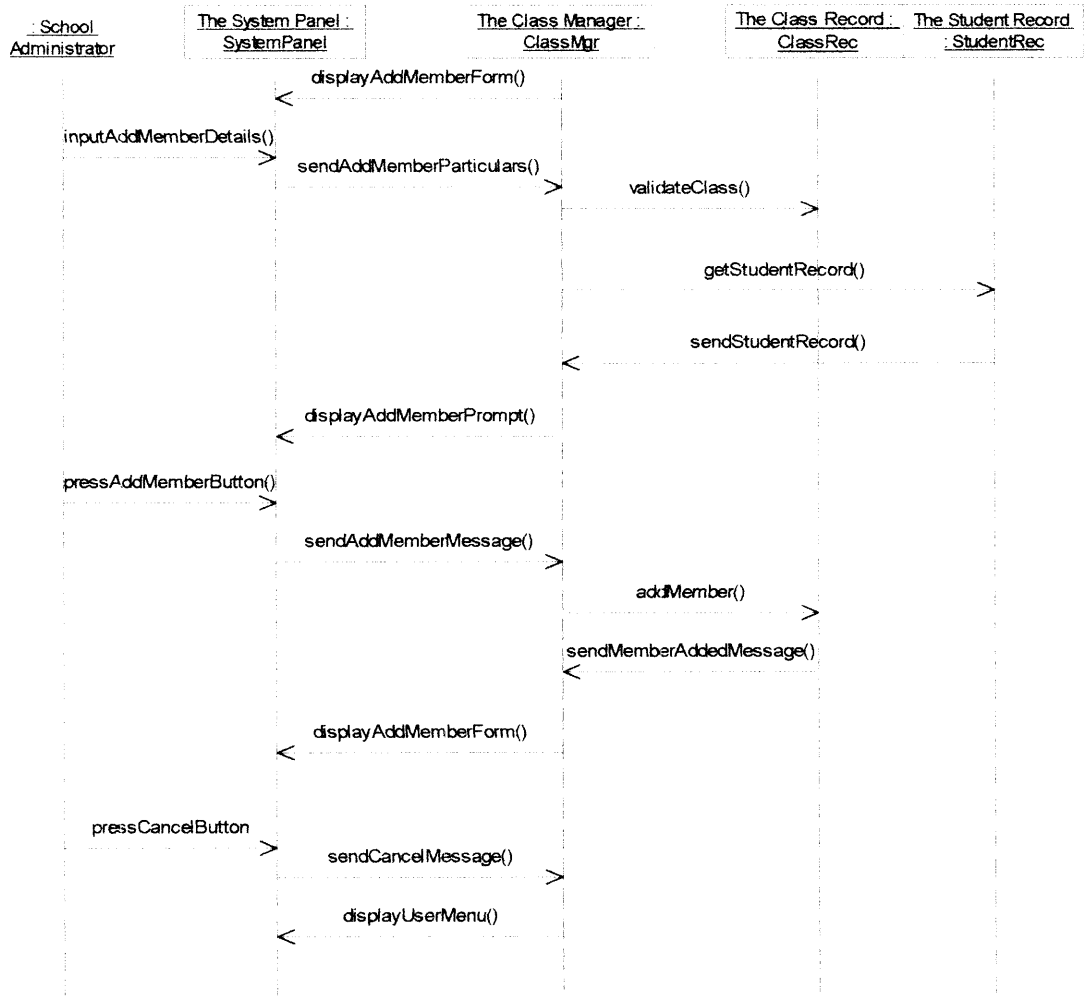
Add Class Member/E1_if class not found



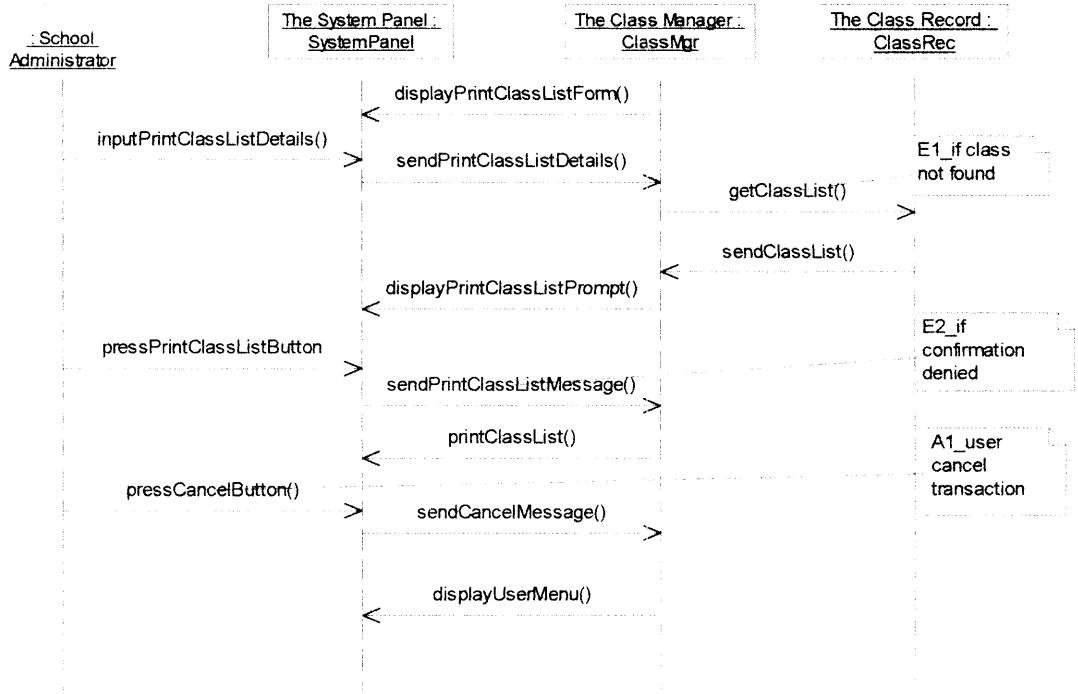
Add Class Member/E2_if student record not found



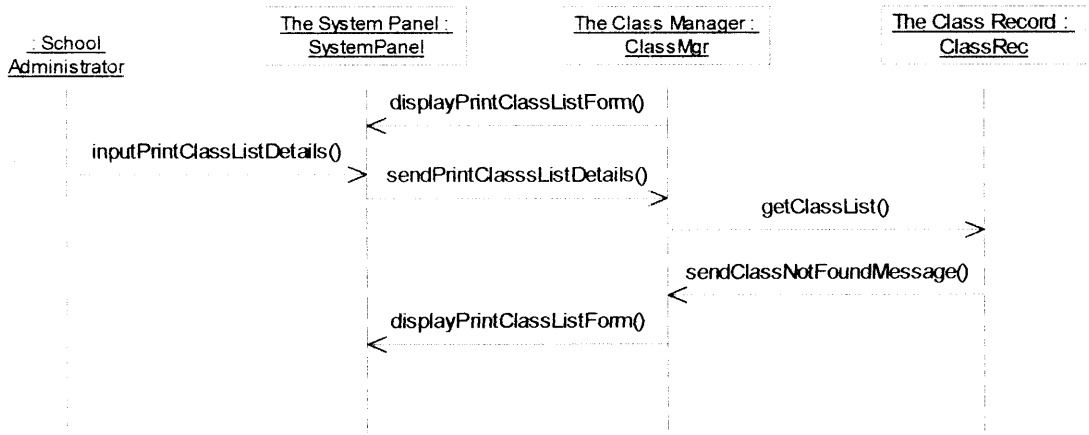
Add Class Member/E3_if confirmation denied



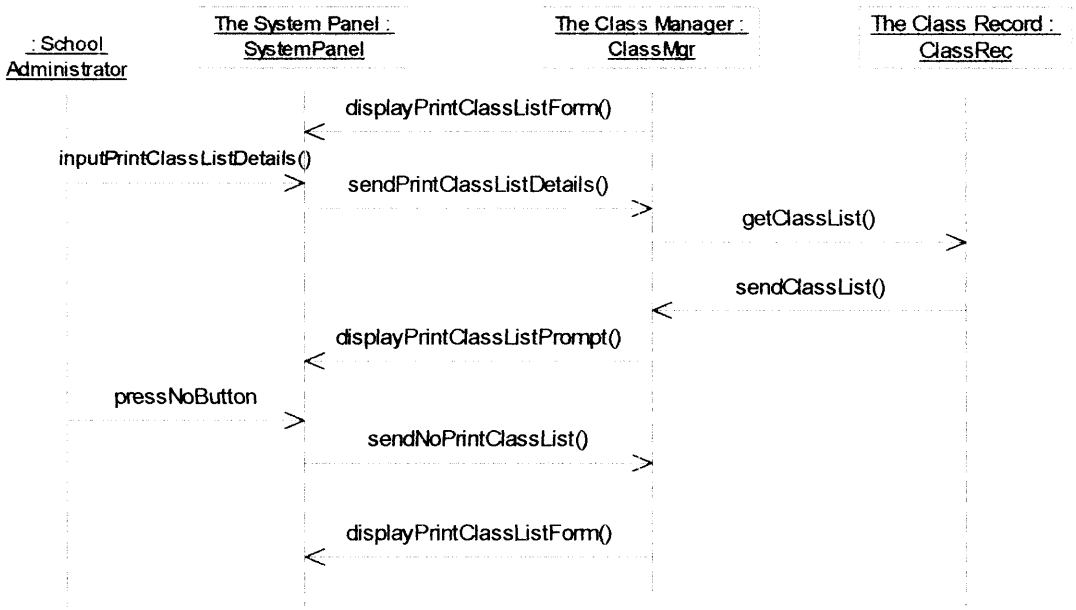
Add Class Member/A1_user cancel transaction



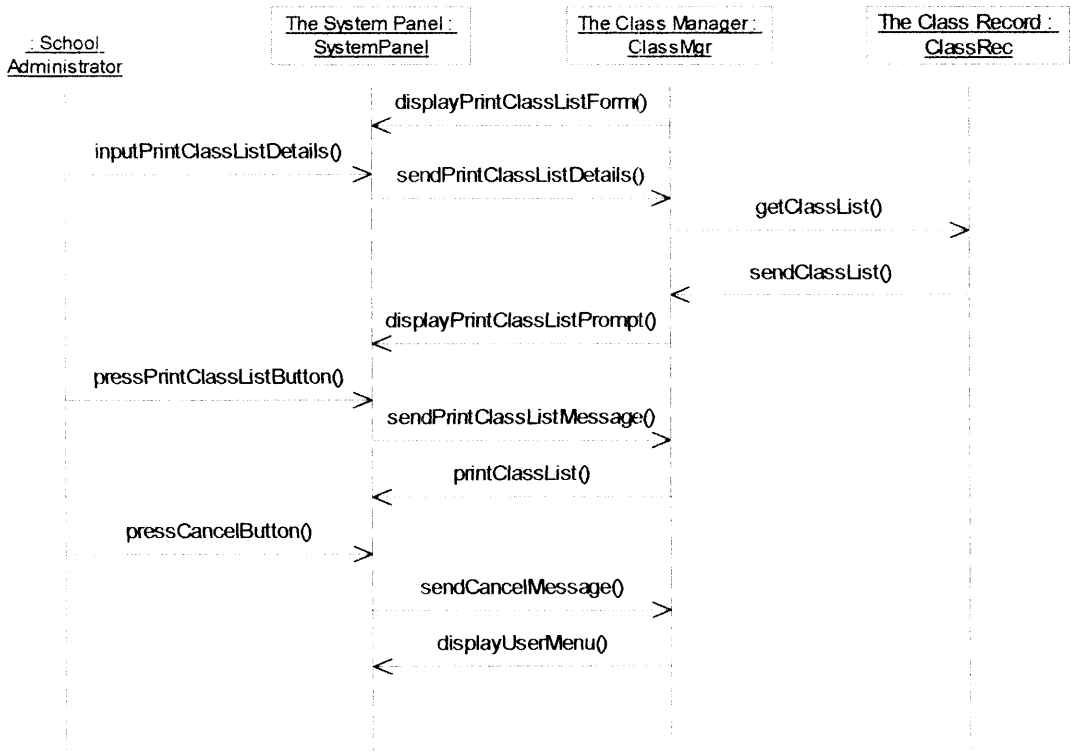
Print Class List



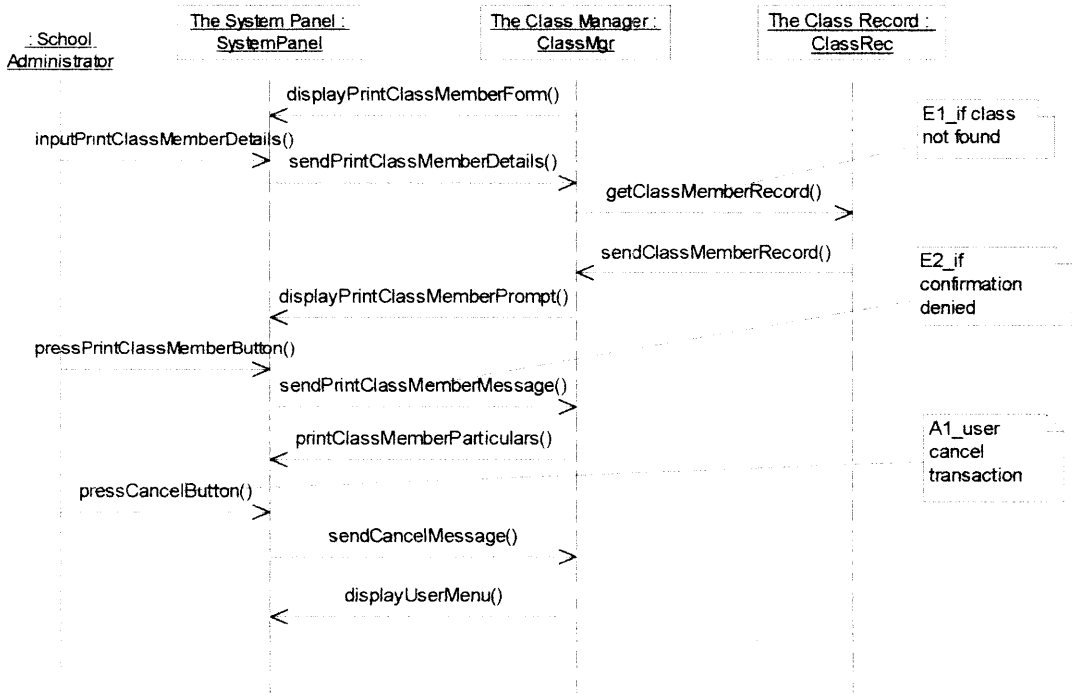
Print Class List/E1_if class not found



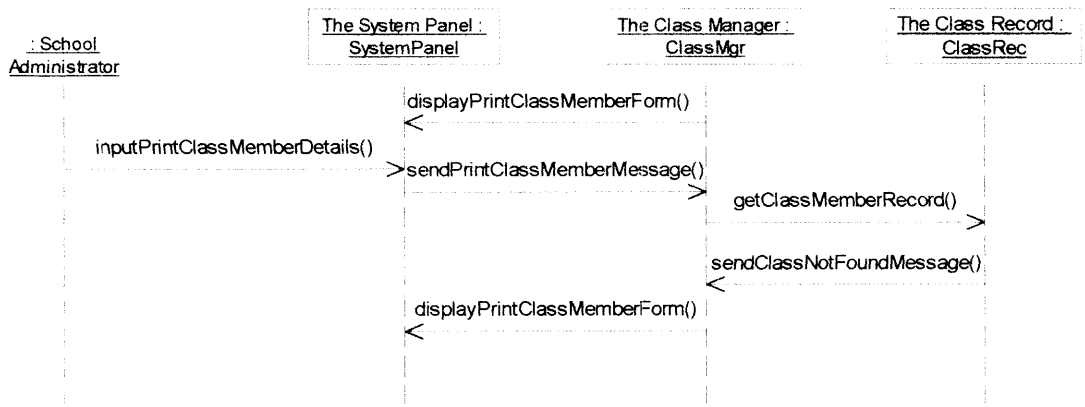
Print Class List/E2_if confirmation denied



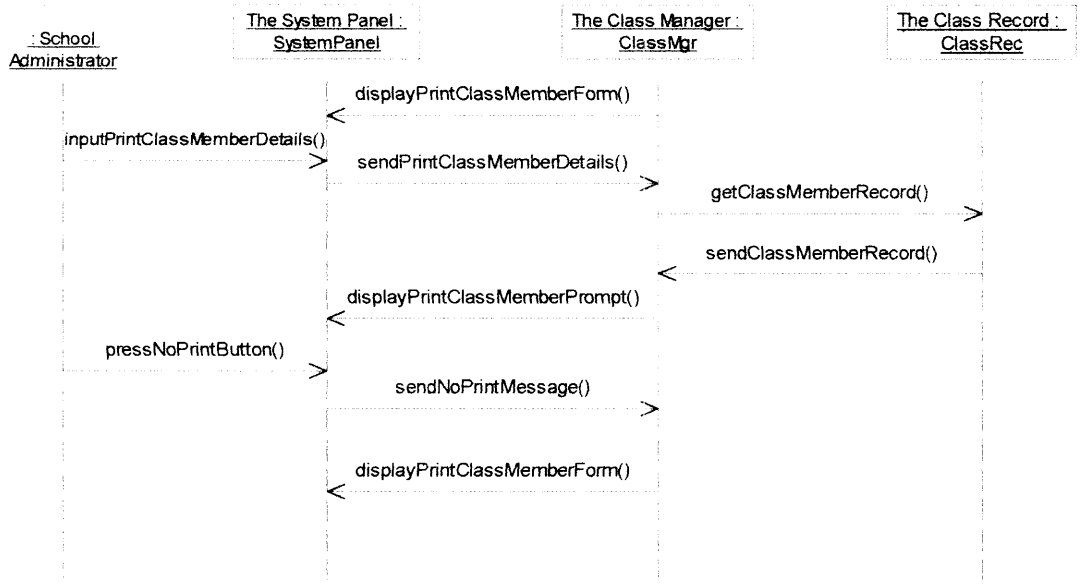
Print Class List/A1_user cancels transaction



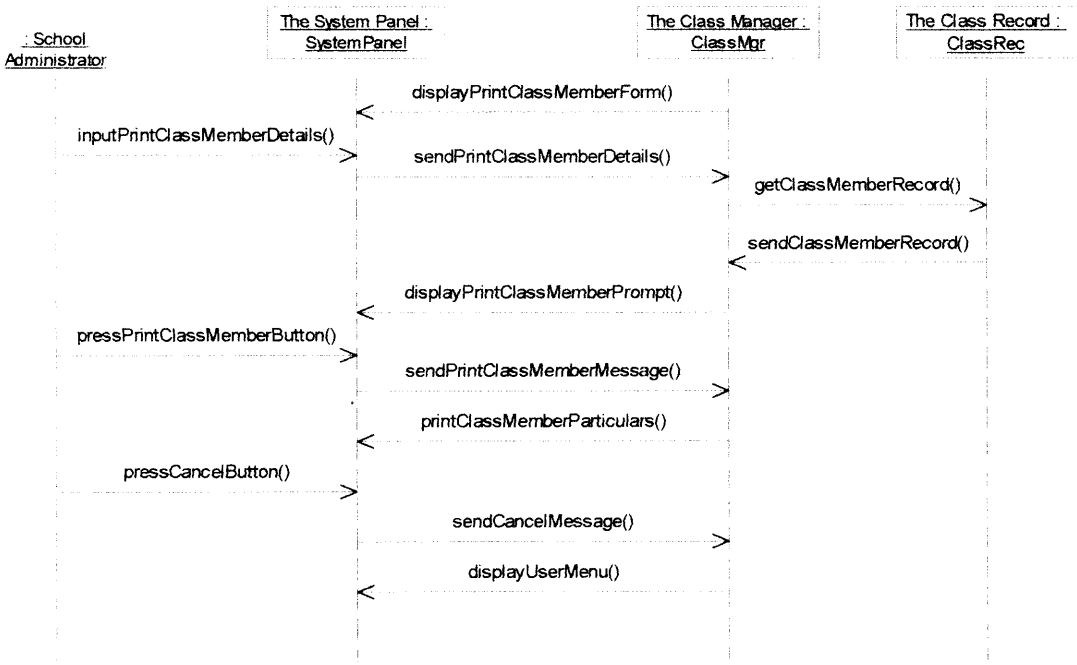
Print Class Member Particulars



Print Class Member Particulars/E1 if class not Found

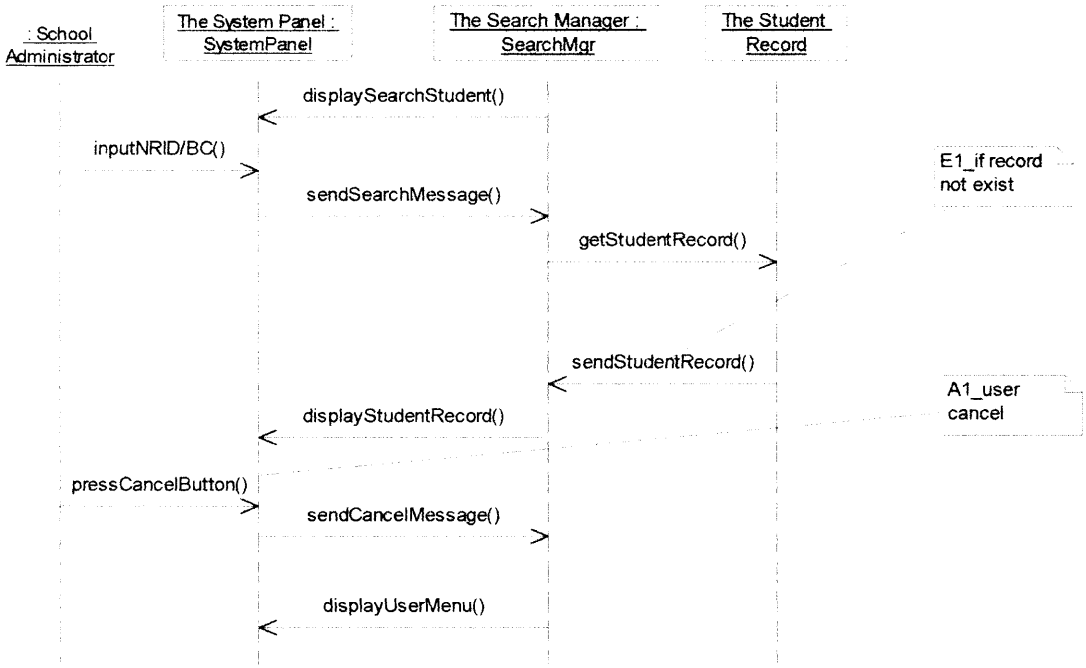


Print Class Member Particulars/E2 if confirmation denied

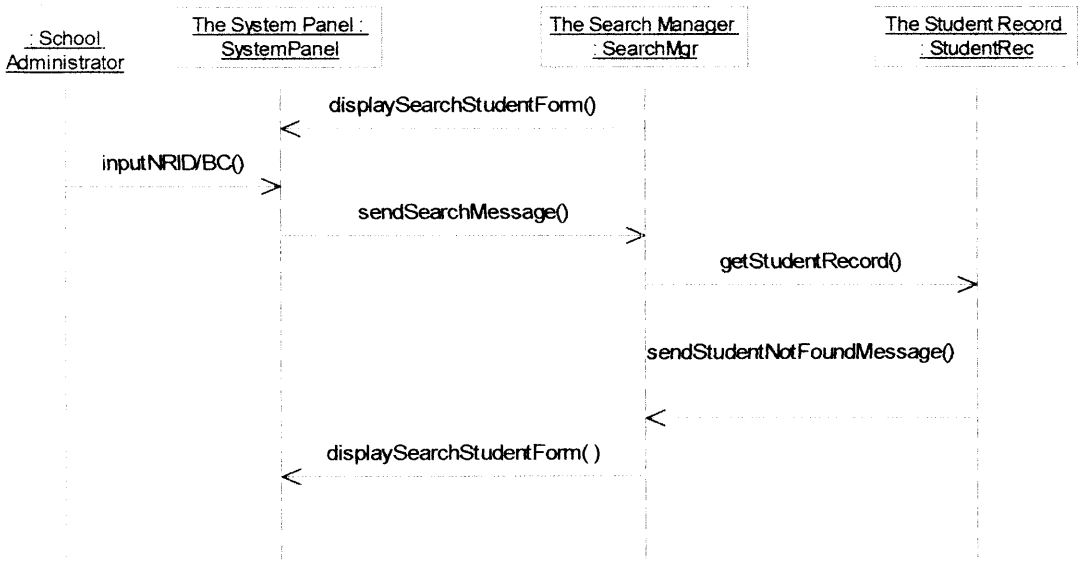


Print Class Member Particulars/A1-user cancel transaction

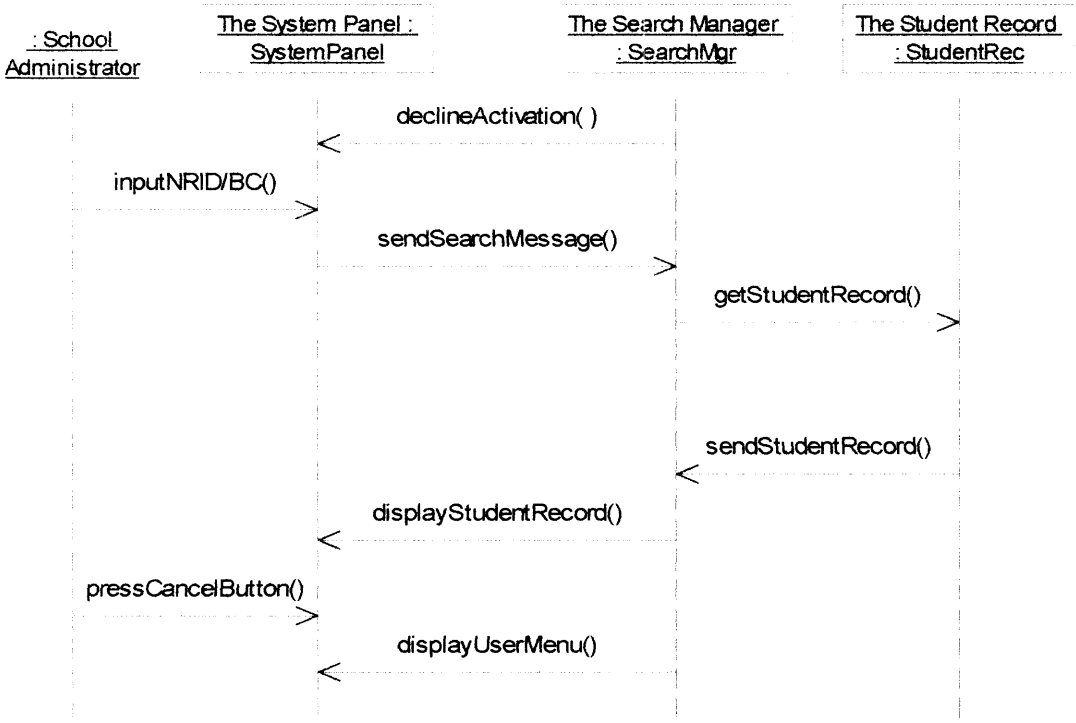
View Student Package Sequence Diagram



Search Student Record



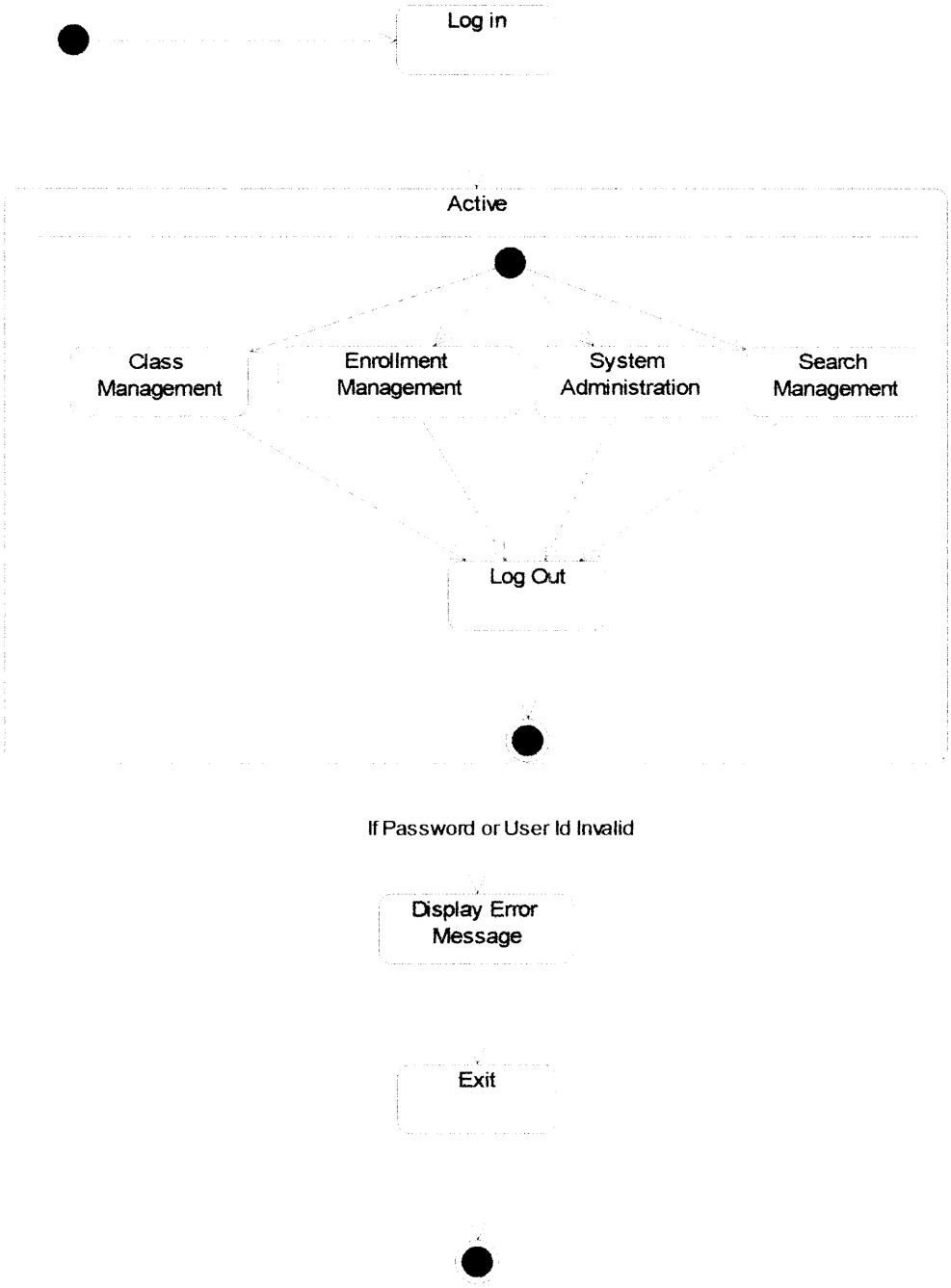
Search Student Record/E1_if student record not found



Search Student Record/A1_user cancel transaction

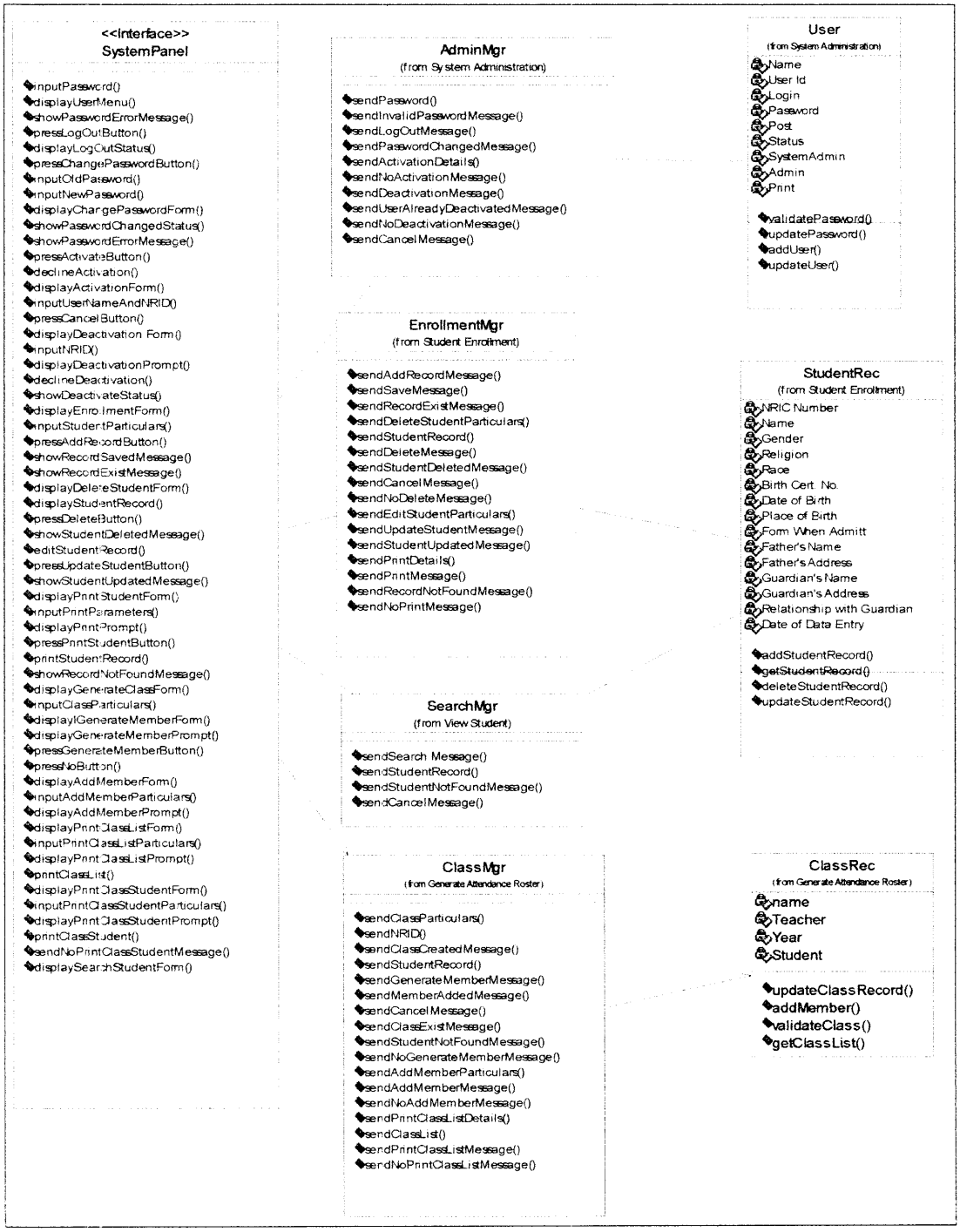
APPENDIX B

STATE DIAGRAM


Log in
Active
Class Management
Enrollment Management
System Administration
Search Management
Log Out
If Password or User Id Invalid
Display Error Message
Exit

APPENDIX C

CLASS DIAGRAM


<<Interface>>
SystemPanel
inputPassword()
displayUserMenu()
showPasswordErrorMessage()
pressLogOutButton()
displayLogOutStatus()
pressChangePasswordButton()
inputOldPassword()
inputNewPassword()
displayChangePasswordForm()
showPasswordChangedStatus()
showPasswordErrorMessage()
pressActivateButton()
declineActivation()
displayActivationForm()
inputUserNameAndNRID()
pressCancelButton()
displayDeactivation Form()
inputNRID()
displayDeactivationPrompt()
declineDeactivation()
showDeactivateStatus()
displayEnrolmentForm()
inputStudentParticulars()
pressAddRecordButton()
showRecordSavedMessage()
showRecordExistMessage()
displayDeleteStudentForm()
displayStudentRecord()
pressDeleteButton()
showStudentDeletedMessage()
editStudentRecord()
pressUpdateStudentButton()
showStudentUpdatedMessage()
displayPrintStudentForm()
inputPrintParameters()
displayPrintPrompt()
pressPrintStudentButton()
printStudentRecord()
showRecordNotFoundMessage()
displayGenerateClassForm()
inputClassParticulars()
displayGenerateMemberForm()
displayGenerateMemberPrompt()
pressGenerateMemberButton()
pressNoButton()
displayAddMemberForm()
inputAddMemberParticulars()
displayAddMemberPrompt()
displayPrintClassListForm()
inputPrintClassListParticulars()
displayPrintClassListPrompt()
printClassList()
displayPrintClassStudentForm()
inputPrintClassStudentParticulars()
displayPrintClassStudentPrompt()
printClassStudent()
sendNoPrintClassStudentMessage()
displaySearchStudentForm()
AdminMgr
(from System Administration)
sendPassword()
sendInvalidPasswordMessage()
sendLogOutMessage()
sendPasswordChangedMessage()
sendActivationDetails()
sendNoActivationMessage()
sendDeactivationMessage()
sendUserAlreadyDeactivatedMessage()
sendNoDeactivationMessage()
sendCancelMessage()
User
(from System Administration)
Name
User Id
Login
Password
Post
Status
SystemAdmin
Admin
Print
validatePassword()
updatePassword()
addUser()
updateUser()
EnrollmentMgr
(from Student Enrollment)
sendAddRecordMessage()
sendSaveMessage()
sendRecordExistMessage()
sendDeleteStudentParticulars()
sendStudentRecord()
sendDeleteMessage()
sendStudentDeletedMessage()
sendCancelMessage()
sendNoDeleteMessage()
sendEditStudentParticulars()
sendUpdateStudentMessage()
sendStudentUpdatedMessage()
sendPrintDetails()
sendPrintMessage()
sendRecordNotFoundMessage()
sendNoPrintMessage()
StudentRec
(from Student Enrollment)
NRIC Number
Name
Gender
Religion
Race
Birth Cert. No.
Date of Birth
Place of Birth
Form When Admitt
Father's Name
Father's Address
Guardian's Name
Guardian's Address
Relationship with Guardian
Date of Data Entry
addStudentRecord()
getStudentRecord()
deleteStudentRecord()
updateStudentRecord()
SearchMgr
(from View Student)
sendSearch Message()
sendStudentRecord()
sendStudentNotFoundMessage()
sendCancelMessage()
ClassMgr
(from Generate Attendance Roster)
sendClassParticulars()
sendNRID()
sendClassCreatedMessage()
sendStudentRecord()
sendGenerateMemberMessage()
sendMemberAddedMessage()
sendCancelMessage()
sendClassExistMessage()
sendStudentNotFoundMessage()
sendNoGenerateMemberMessage()
sendAddMemberParticulars()
sendAddMemberMessage()
sendNoAddMemberMessage()
sendPrintClassListDetails()
sendClassList()
sendPrintClassListMessage()
sendNoPrintClassListMessage()
ClassRec
(from Generate Attendance Roster)
name
Teacher
Year
Student
updateClassRecord()
addMember()
validateClass()
getClassList()

APPENDIX B

Software Design Document

SOFTWARE DESIGN DOCUMENT
(SDD)

for the

Web-based Student Enrollment System for Malaysian Government Secondary School

Student Enrollment System
(SES)

Prepared For: Graduate School, Universiti Utara Malaysia

Prepared By: Gee Kheng Chuan (83231)

TABLE OF CONTENTS

(SDD) ..... **I**
LIST OF FIGURES ..... **III**
TABLE OF APPENDIXES ..... **III**

1.0 INTRODUCTION **1**
1.1 Identificati **1**
1.2 System Overview **1**
1.3 Document Overview **1**

2.0 REFERENCED DOCUMENTS **2**

3.0 PRELIMINARY DESIGN **2**
3.1 System Overview **2**
3.1.1 System architecture 2
3.1.1.1 Static Organization 3
3.1.1.1.1 Data Specification 3
3.1.1.1.2 Class Diagram 4
3.1.1.1.3 Component Diagram 5
3.1.1.1.4 Deployment Diagram 8
3.1.1.2 Dynamic Organization 8
3.1.1.2.1 Sequence Diagram 8
3.1.1.2.2 Collaboration Diagram 8
3.1.1.2.3 State Diagram 8
3.1.1.3 Package (s) Interfaces 9
3.1.2 System States and Modes 10
3.1.3 Memory and Processing Time Allocation 10
3.2 System Design Description **10**
3.2.1 Presentation Layer (SDD_ SES_1000) 10
3.2.2 Application Logic Layer (SDD_ SES_ 2000) 10
3.2.3 Storage Layer (SDD_SES_3000) 11

4.0 DETAILED DESIGN **12**
4.1 Presentation Layer Package **12**
4.1.1 SystemAdmin Panel Class 12
4.1.2 StudentEnrollment Panel Class 18
4.1.3 ClassAdmin Panel Class 20
4.1.4 SearchStudent Panel Class 21
4.1.5 Print Panel Class 22
4.2 Application Layer Package **25**
4.2.1 SystemAdmin Manager Class 25
4.2.2 StudentEnrollment Manager Class 29
4.2.3 ClassAdmin Manager Class 31
4.2.4 SearchStudent Manager Class 32
4.2.5 Print Manager Class 33

4.3 Storage Layer Package **35**
4.3.1 STAFF Class 35
4.3.2 STUDENT class 36
4.3.3 CLASS class 38
4.3.4 CLASSTD class 38

5.0 NOTES **39**

LIST OF FIGURES

Figure 3.1: System Architecture Diagram 3
Figure 3.2: Component Diagram for SES System 5
Figure 3.3: Main component diagram for SES System 6
Figure 3.4: Boundary package component diagram 6
Figure 3.5: Control package component diagram 7
Figure 3.6: Entity package component diagram 7
Figure 3.7: Deployment Diagram for SES System 8
Figure 3.8: System Packages Diagram 9
Table 3.9: Roles of SES Packages 9
Figure 3.9: Visibility of Presentation Layer With Other Package 10
Figure 3.10: Visibility of Application Logic Layer With Other Packages 11
Figure 3.11: Visibility of Storage Layer With Other Packages 11

LIST OF TABLES

Table 1a & 1b: Data Specifications
Table 2: Streotypes Applying to Components On UML Component Diagram of SES System
Table 3: Roles of SES Packages

TABLE OF APPENDIXES

Appendix A **Class Diagram**
Appendix B **Sequence Diagram**
Appendix C **Collaboration Diagram**
Appendix D **State Diagram**

1.0 INTRODUCTION

1.1 Identification

System Name:	Web-based Student Enrollment System for a Malaysian Government Secondary School
System Abbreviation:	SES

1.2 System Overview

This project is a web based student enrollment system for Malaysian government secondary schools. This web based system is mean to manage the student enrollment data and generate student data for attendance roster. This web based system is important for government secondary school as it takes away most of the student personal data entry role from teacher and administrators and pass it to the would be students. This leaves the administrator and teachers to do much lesser work of maintenance. The main objectives are to store student's enrollment data in order to minimize the data redundancies; to search and retrieve student records based on the National Registration Identity (NRID) or birth certificate number and; to generate attendance roster with intention to improve data integrity of student's attendance roster.

1.3 Document Overview

The purpose of this Software Design Document (SDD) is to represent the requirements that the writer referred to Software Requirement Specification (SRS). The developer can refer to this document time to time during development period. It is to make sure that the developed system does fulfill the user needs. In this case the percentage of develop the wrong system may be low.

Besides that, this document can give a brief picture about design of the system and interface involve in the system to the developer. The content of the system included a list of figure like External Interface Diagram, System Package Diagram, list of table and appendixes (Appendix A: Class Diagram, Appendix B: Sequence Diagram, Appendix C: Collaboration Diagram, Appendix D: State Diagram and Appendix E: Activity Diagram.)

The first content is the introduction, which included identification, system overview and documents overview. The second content is the reference documents. Preliminary Design would be the third content. It describes the highest level of interaction between external objects and the system. It included System Architecture (3.1.1), System States and Modes (3.1.2), and Memory and Processing Time Allocation (3.1.3). System Architecture describes the internal organization structure of the system. It has Static Organization (3.1.1.1), Dynamic Organization (3.1.1.2) and Packages Interfaces (3.1.1.3). Static Organization describes the packages of the system and the relationships or dependency between the package diagrams meanwhile Dynamic Organization describes the process involved in the system. Relationship between the packages will explain in Package Interface. System States and Modes describe the package name and the related use cases. Memory and Processing Time Allocation System will list the constraints on the memory and processing time allocation. Design Description (3.2), which explains the internal classes inside a package and also its visibility to external package.

The Detailed Design is given in part 4.0. It describes the detailed of the components structure, the algorithms, low-level requirements, and the control and data structure completely.

Lastly, all the abbreviation used in the document is listed in part 5.0.

2.0 REFERENCED DOCUMENTS

Bennett,S., McRobb,S., & Farmer,R. (2002).Object-*Oriented System Analysis and Design Using UML*. London.McGraw-Hill

Boggs, Wendy, & Michael (1999). *Mastering UML with Rational Rose*. United States of America: SYBEX Inc.

Graham, Ian.(2001). *Object-oriented Method Principles and Practice*. 3rd Edition. Addisson Wesley, Pearson Education.

Frank Armour, Granville Miller (2001). *Advanced Use Case Modeling Software System*. Addison Wesley Publishing

Roger S. Pressman (2001). *Software EngineeringA Practitioner's Approach*. McGraw Hill

3.0 PRELIMINARY DESIGN

3.1 System Overview

SES System uses Visual Basic Script which is a server-side scripting language. The scripts are basically html pages with Visual Basic language constructs contained within the pages. A server-side interpreter interprets the VB within each script and emits the resulting html and dynamic content to a client. The presentation, application logic and storage layers are merge into one layer within the web application architecture. Basic object orientation support provided by VB script makes it possible to modularize web applications. These modules are of three types: application specific, database manipulation and presentation modules.

3.1.1 System Architecture

SES Systems web applications can be model using three layers. The layers are presentation, application logic, and storage layers. The presentation layer is the set of html pages with which the user request services from the application. This layer maps the presentation entities within the html pages onto application logic layer services. The application layer contains the "logic" which translates user requests into data manipulation and presentation tasks. These tasks include the reformatting of data from the storage layer into an appropriate form for the presentation layer and the mapping of service requests from the presentation layer into a set of tasks for the storage layer. The storage layer is used to connect to data entities with a data-store or database. This layer translates data manipulation tasks into database tasks.

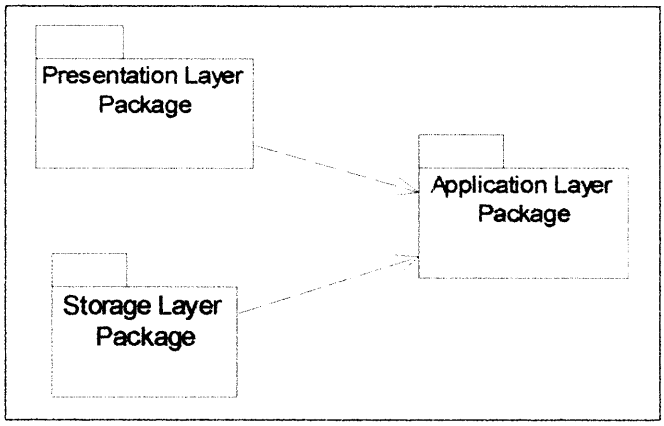


Figure 3.1 System Architecture Diagram

3.1.1.1 Static Organization

Static organization shows the static structure of the model, includes classifiers, relationship, attributes and operations. It does not show temporal information. Types of static organization are class diagram, component diagram, deployment diagram, object diagram and use case.

3.1.1.1.1 Data Specification

Data specification is all attributes in SES system specify it type, initial value and constraints. Attributes are part of the essential description of a class. They belong to the class, unlike objects, which instantiate the class. Attributes are the common structure of what a member of the class can "know".

STAFF Class			
Attribute	**Type**	**Initial Value**	**Constraints**
STAFFid	String	"Unknown"	Length=1-15
STAFFname	String	"empty"	Length=0-100
STAFFlogin	String	"empty"	Length=0-15
STAFFpwd	String	"empty"	Length=0-15
STAFFpost	String	"empty"	Length=0-50
STAFFstatus	String	"empty"	Length=0-5
STAFFsysadmin	Boolean	"False"	-
STAFFadmin	Boolean	"False"	-
STAFFprint	Boolean	"False"	-
STAFFtimestamp	datetime	"empty"	-

Table 1a: Data Secifications

STUDENT Class			
Attribute	**Type**	**Initial Value**	**Constraints**
STDid	String	"unknown"	Length=1-15
STDname	String	"empty"	Length=0-100
STDpwd	String	"empty"	Length=0-100
STDsex	String	"empty"	Length=0-100
STDreligion	String	"empty"	Length=0-100
STDrace	String	"empty"	Length=0-100
STDbirthcert	String	"empty"	Length=0-100
STDdob	datetime	"empty"	Length=0-100
STDpob	String	"empty"	Length=0-100
STDiniclass	String	"empty"	Length=0-15
STDfathername	String	"empty"	Length= 0-100
STDfatheradd	String	"empty"	Length=0-200
STDguardianname	String	"empty"	Length= 0-100
STDguardianadd	String	"empty"	Length=0-200
STDguardiantype	String	"empty"	Length=0-50
STDstampdate	datetime	"empty"	-
STDadmissionNum	numeric	"unknown"	Length=1-18
STDsijil	Boolean	"False"	-
STDcertnum	String	"empty"	Length=0-12
STDleavedate	datetime	"empty"	-
STDleaveReason	String	"empty"	Length=0-100
STDactive	Boolean	"False"	-
CLASS Class			
Attribute	**Type**	**Initial Value**	**Constraints**
CLASSid	numeric	"unknown"	Digit 1-15
CLASSname	String	"Empty"	Length=0-50
CLASSteacher	String	"Empty"	Length=0-50
CLASSyear	datetime	"Empty"	
CLASSTD Class			
Attribute	**Type**	**Initial Value**	**Constraints**
CLASSid	numeric	"unknown"	Digit 1-15
STDid	String	"unknown"	Length=1-15
CStimestamp	Datetime	"empty"	-

Table 1b: Continuation Data Specifications

3.1.1.1.2 Class Diagram

Class Diagram in details design is all classes with the types of attributes specified and the operation parameters defined, which explains in data specification (3.1.1.1.1) and method specification (4.0). Please refer Appendix A (Class Diagram in Details Design)

3.1.1.1.3 Component Diagram

In a project there will be many files that make up the system. These files will have dependencies on one another. Component diagram show these dependencies between software components in the system. As you can see in figure 3.2 are modeled as rectangles with two smaller rectangles jutting out from the left-hand side. Components realize one or more interfaces, modeled using the lollipop notation- and may have dependencies on other components - as you can see the SES System' component has a dependency on the Java Script component. Stereotypes can be used to show the types of different components. Table below summarizes stereotypes that applying to components on UML Component diagram SES system.

Stereotype	Indicates
<<application>>	A "front-end" of your system, such as the collection of HTML pages and ASP that work with them for a browser-based system or the collection of screens and controller classes for a GUI-based system.
<<database>>	A hierarchical, relational, object-relational, network or object-oriented database.
<<file>>	A source code file, such as a .vb file or a .js file.
<<Framework>>	One or more web services.

Table 2: Stereotypes that applying to components on UML Component diagram of SES system.

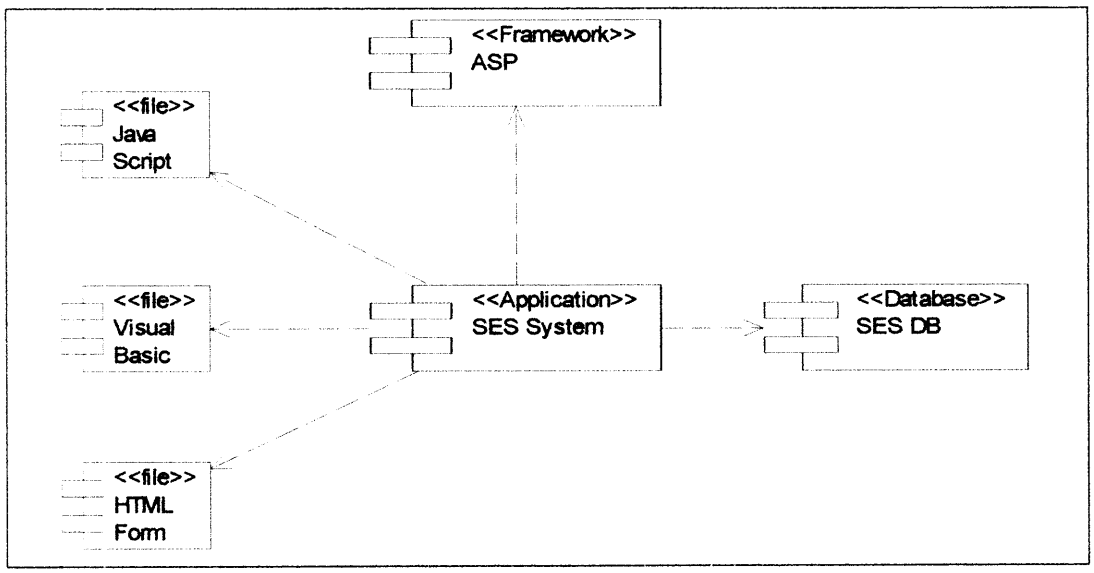


Figure 3.2: Component Diagram for SES System

Beside that, a component diagram can bid to show dependencies between large scale components within a system. SES systems uses package diagram to show the logical grouping of class diagrams in component diagram. It means packages contain components as well as other packages. There is boundary, control and entity packages in component diagram of SES.

Figure 3.3 shows the main component diagram for the entire system. This main diagram focuses on the packages of components that have been created.

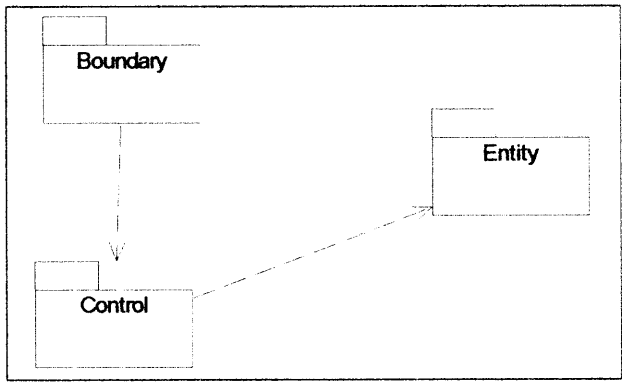


Figure 3.3: Main component diagram for SES System

Figure 3.4 includes the components in the Boundary package. These components will contain the classes in the Presentation Layer Package in the logical view.

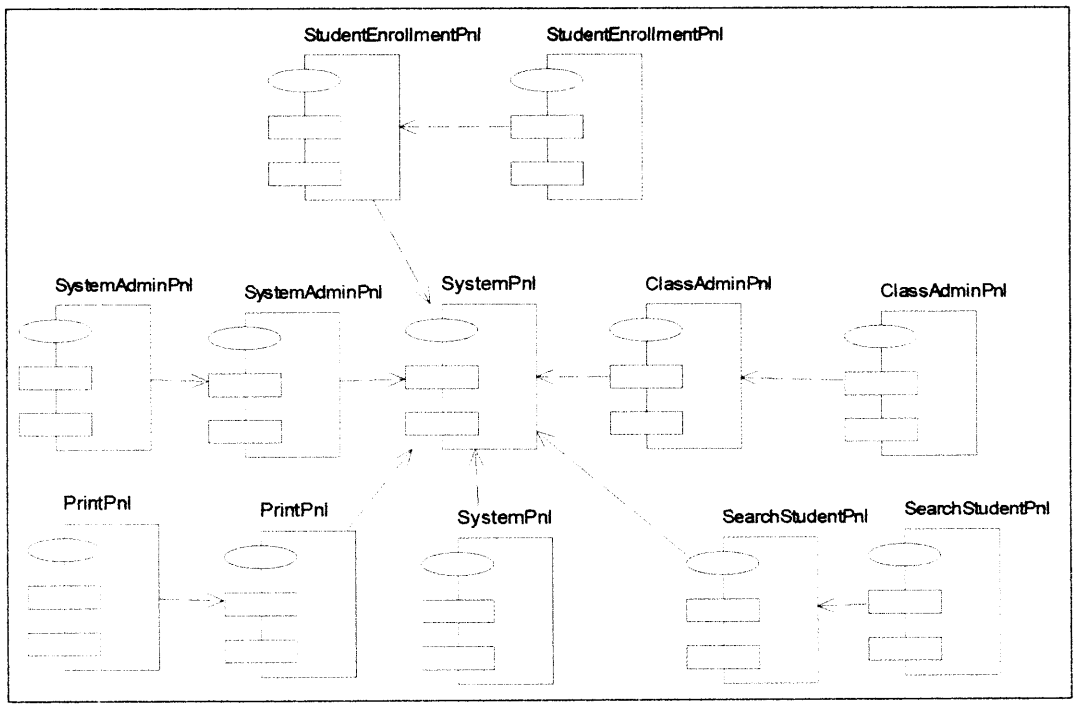


Figure 3.4: Boundary package component diagram

Figure 3.5 includes all of the components in the Control package. These components will contain the classes in the Application Logic Layer Package in the logical view.

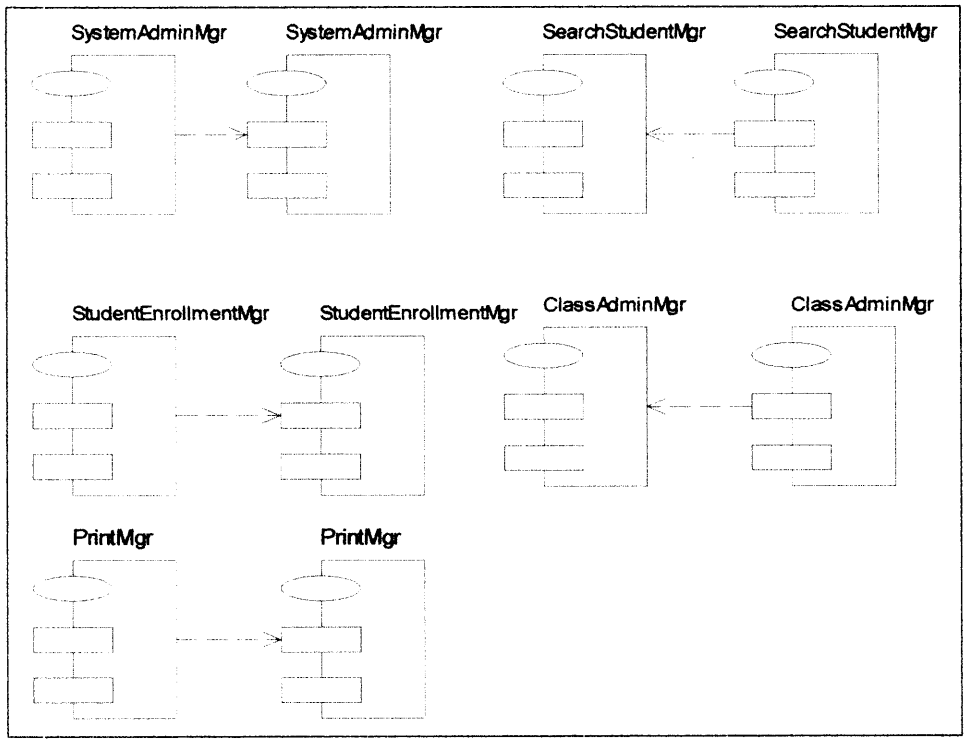


Figure 3.5: Control package component diagram

Figure 3.6 includes all the components in the Entity package. These are the components that will contain the classes in the Storage Layer Package in the logical view.

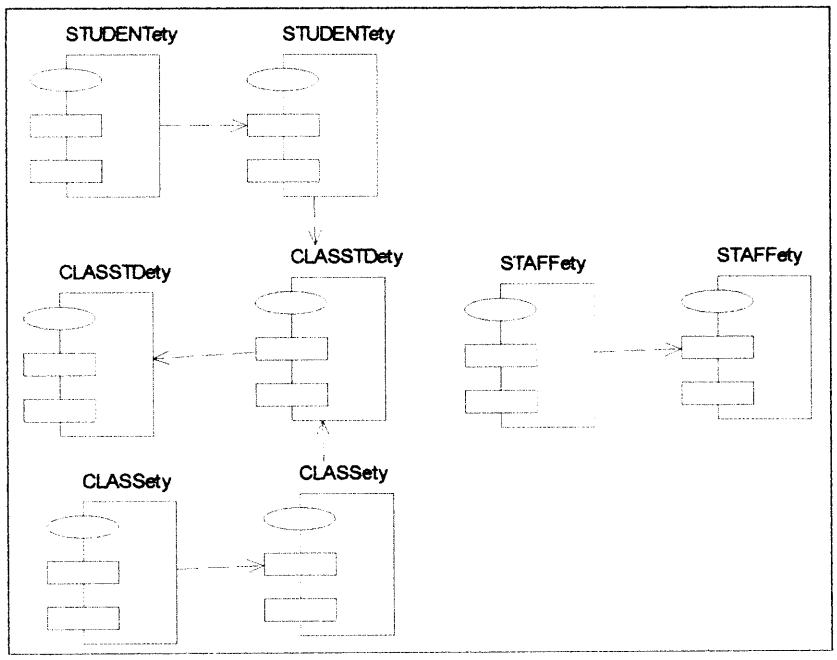


Figure 3.6: Entity package component diagram

3.1.1.1.4 Deployment Diagram

Deployment Diagram is used to show the configuration of run-time processing elements and the software components and processes that are located on them. It is modeling the processors and devices that from the topology of an embedded, client/server, or distributed system. SES is a client/server system using TCP/IP to connect client with server, which describes in figure below.

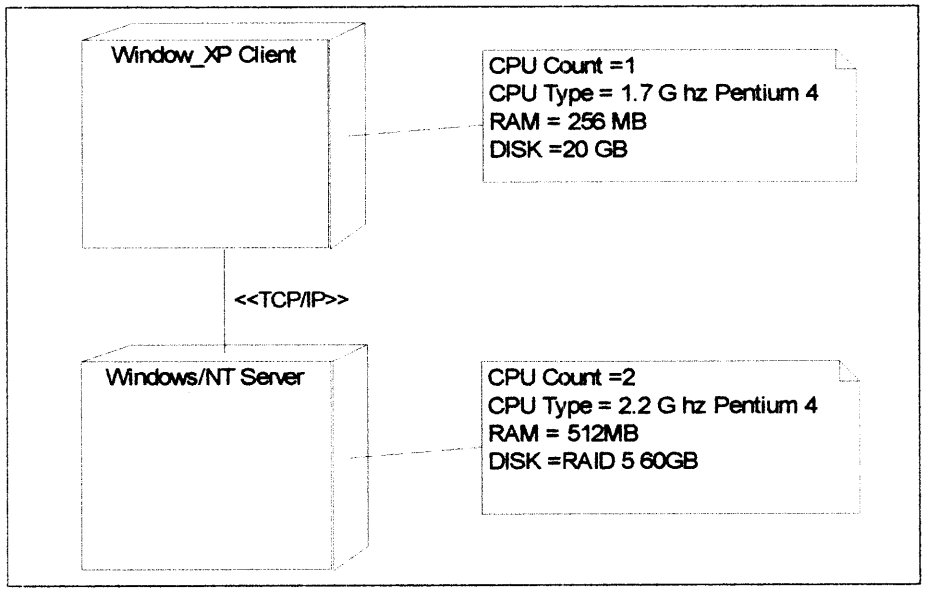


Figure 3.7 Deployment Diagram for SES System.

3.1.1.2 Dynamic Organization

Dynamic organization is collection of operations or procedures, which represent operation system based on time. It is important during design and implementation phase. Types of dynamic organization are sequence diagram, collaboration diagram, state diagram and activity diagram.

3.1.1.2.1 Sequence Diagram

A sequence diagram shows an interaction between objects arranged in a time sequence. Sequence diagram can be drawn at different level of details and to meet different purpose at several stages in the development life cycle. Please refer appendix B (Sequence Diagram)

3.1.1.2.2 Collaboration Diagram

Collaboration diagram have many similarities to sequence diagram. They express the same information in a different format, and, like sequence diagram, they can be drawn at various levels of detail and during different stages in the system development process. Please refer Appendix C (Collaboration Diagram)

3.1.1.2.3 State Diagram

A State diagram shows the static structure of the interface, but it provides no information about how it will behave in response to the user. All objects have a state. The current state of an object is a result of the events that have occurred to the object, and is determined by the current value of the object's attributes and the links that it has with other objects. Please refer Appendix D (State Diagram)

3.1.1.3 Package (s) Interfaces

The class diagram retailed design will group in package. The SES is organised into three packages (using UML definition), which are shown in Figure 3.8. The criteria for grouping classes follow semantic rules. Each package defined a set of role, which results in a minimum of coupling and a maximum of cohesion within the package.

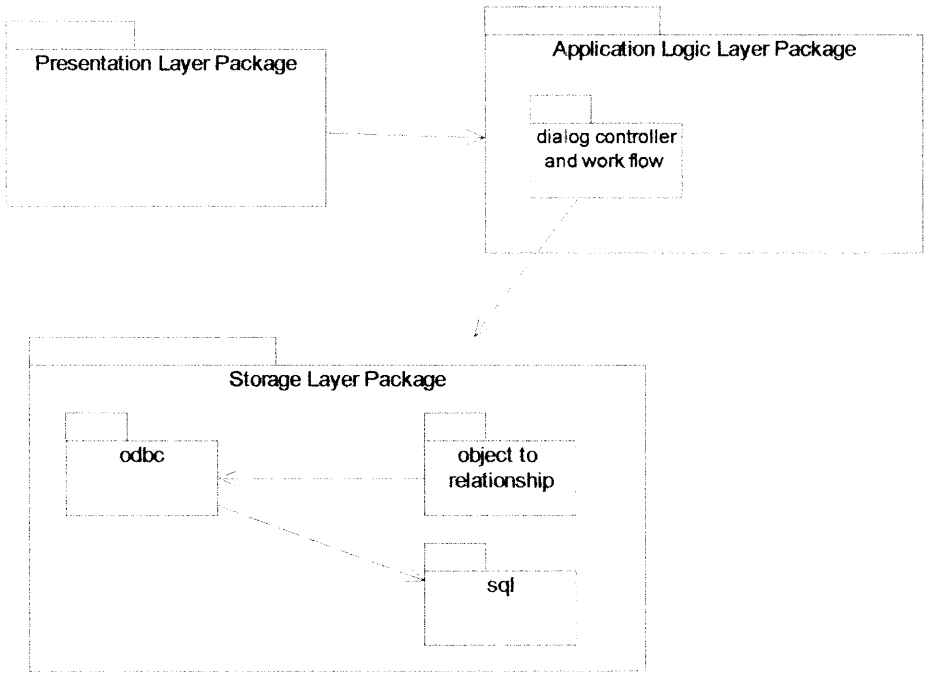


Figure 3.8: System Packages Diagram

Package	Description	Classes
1. Presentation Layer Package	Interface that user interact with the system.	a. System Panel b. SystemAdmin Panel c. StudentEnrollment Panel d. ClassAdmin Panel e SearchStudent Panel e. Print Panel
2. Application Logic Layer Package	Control the activities between the interface and entity.	a. SystemAdmin Manager b. StudentEnrollment Manager c. ClassAdmin Manager d SearchStudent Manager e. Print Manager
3. Storage Layer Package	Manage the record in the system while it is executing.	a. STUDENT b. STAFF c. CLASS d. CLASSTD

Table 3: Roles of SES Packages

There are dependency relationships between the three packages. The presentation layer package depends on the application logic layer package to send the message to storage layer package while

executing a function. On the other hand, the application logic layer package depends on the storage layer package to get the executed results and send as feedback to user.

3.1.2 System States and Modes

Not Applicable.

3.1.3 Memory and Processing Time Allocation

Not Applicable.

3.2 System Design Description

The following are the internal classes in each of the packages and their visibility to external packages.

3.2.1 Presentation Layer (SDD_ SES_1000)

The presentation layer package groups the elements that will administer system and manage staff activities. The classes that can be found in this package are SystemPanel, SystemAdminLogin Panel, StaffLogin Panel, StudentLogin Panel, SystemAdmin Panel, StudentEnrollment Panel, and ClassAdmin Panel. The classes for SystemAdminLogin Panel, StaffLogin Panel, StudentLogin Panel, SystemAdmin Panel, StudentEnrollment Panel, ClassAdmin Panel are inheriting System Panel. (Refer Class Diagram) The presentation layer package communicates with the application layer package, as shown below.

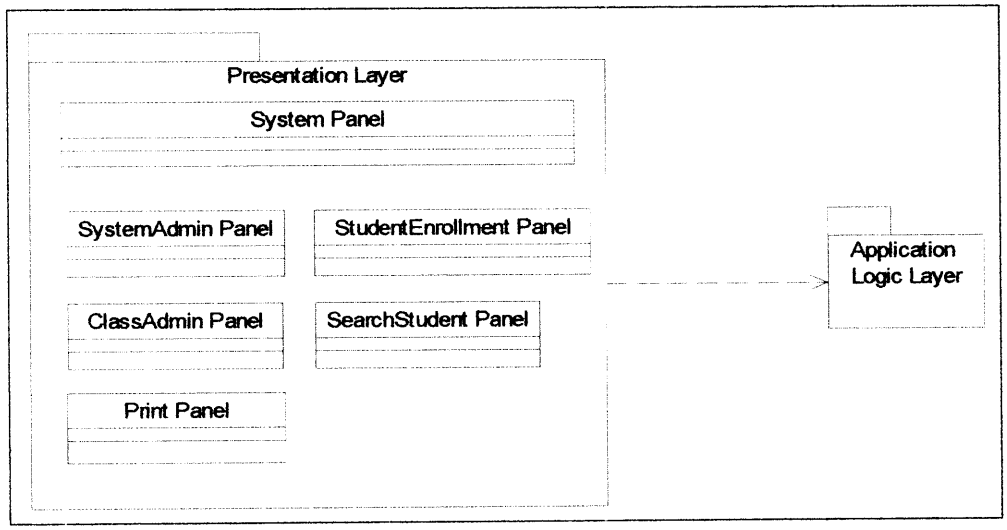


Figure 3.9 Visibility of Presentation Layer With Other Package

3.2.2 Application Logic Layer (SDD_ SES_ 2000)

The application logic layer package groups the elements that will be managing and controlling the system. The classes that can be found in this package are SystemAdmin Manager, StudentEnrollment Manager, and ClassAdmin Manager in Dialog Controller package and Error Handling in Workflow Controller package. Communication with others package like boundary and entity is shown below.

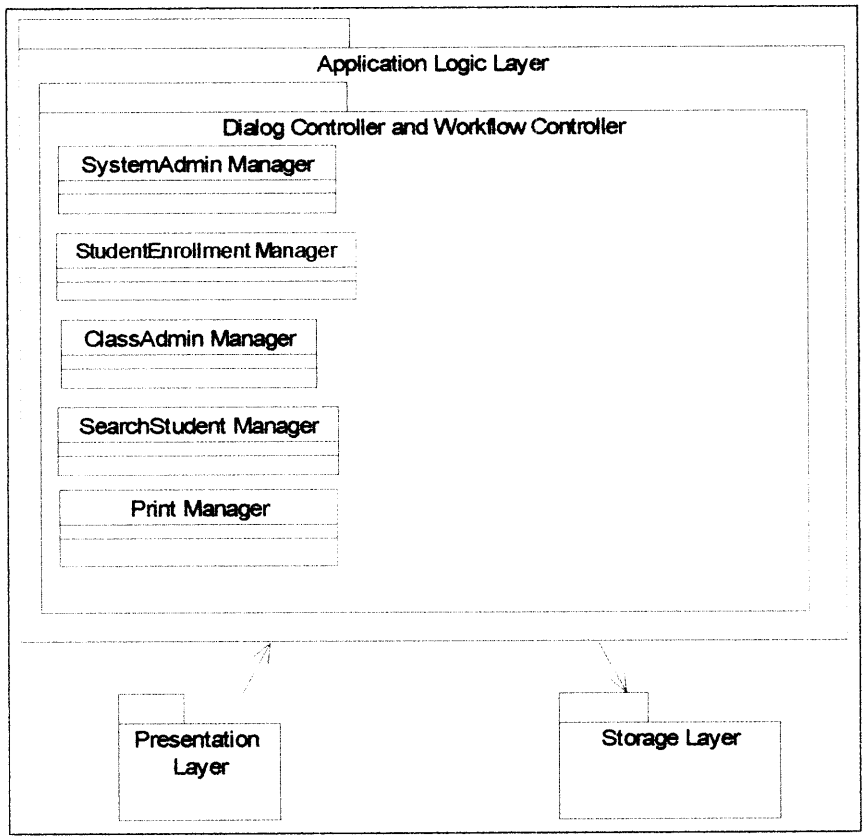


Figure 3.10: Visibility of Application Logic Layer With Other Packages

3.2.3 Storage Layer (SDD_SES_3000)

The storage layer package groups the elements that manage record of the system. The classes that can be found in this package are STUDENT, STAFF, CLASS, and CLASSTD. The SES entity communicates with the application logic layer package as shown below.

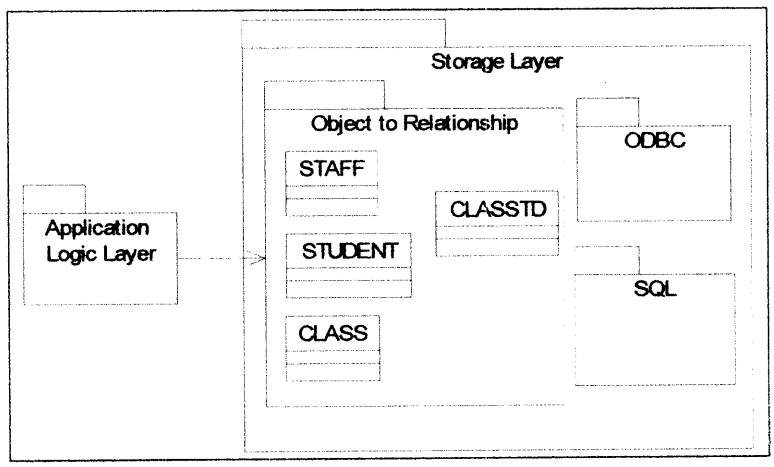


Figure 3.11: Visibility of Storage Layer With Other Packages

4.0 DETAILED DESIGN

Detailed design is the design of operations involves determining the best algorithm to perform the required function. An operation is a specification for some aspect of the behavior of a class. They are actions that can be carried out by, or on, an object. Each operation also has to specify in terms of the parameters that it passes and returns.

4.1 Presentation Layer Package

4.1.1 SystemAdmin Panel Class

Class Type	Boundary class
Responsibility	To communicate with user of the system, which provide interface that interact with user.
Attributes	None
Operation/Method	**DisplaySystemAdminLoginForm()**
Responsibility	To display system administrator login form on SystemAdmin Panel
Input Parameter	None
Output Parameter	None
Precondition	User has click Administrator Login on main page.
Post Condition	System Administrator Login form is displayed.
Algorithm	Begin Send Administrator Login Form to System Admin Panel Class. End
Operation/Method	**inputStaffIdPassword()**
Responsibility	To receive System Administrator's user ID and password.
Input Parameter	STAFFlogin, STAFFpwd
Output Parameter	None
Precondition	User must have STAFFlogin and STAFFpwd.
Post Condition	User has input STAFFlogin and STAFFpwd.
Algorithm	Begin Send STAFFlogin and STAFFpwd entered to Admin Panel class. End
Operation/Method	**displaySystemAdminMenu()**
Responsibility	To display system administrator menu on SystemAdmin Panel
Input Parameter	None
Output Parameter	None
Precondition	User must have entered valid user ID and Password
Post Condition	System Administrator Menu being displayed.
Algorithm	Begin Send system administrator menu to Admin Panel class and display. End
Operation/Method	**displayInvalidPasswordMessage()**
Responsibility	To inform the system administrator that wrong user ID or/and Password have been entered
Input Parameter	STAFFlogin, STAFFpwd
Output Parameter	None
Precondition	STAFF class has validated the user ID and Password.
Post Condition	Login failed. Administrator Login Form being redisplay
Algorithm	Begin Display "Login or Password is incorrect". End

Operation/Method	**displaySchoolAdminLoginForm()**
Responsibility	To display school administrator login form on SystemAdmin Panel
Input Parameter	None
Output Parameter	None
Precondition	User has click Staff Login on main page.
Post Condition	School Administrator Login form is displayed
Algorithm	Begin Send School Administrator Login Form to SystemAdmin Panel Class. End
Operation/Method	**inputStaffIdPassword()**
Responsibility	To receive school administrator's user ID and password.
Input Parameter	STAFFlogin, STAFFpwd
Output Parameter	None
Precondition	User must have STAFFlogin and STAFFpwd.
Post Condition	User has input STAFFlogin and STAFFpwd.
Algorithm	Begin Send STAFFlogin and STAFFpwd entered to SystemAdmin Panel class. End
Operation/Method	**displaySchoolAdminMenu()**
Responsibility	To display school administrator menu on SystemAdmin Panel
Input Parameter	None
Output Parameter	None
Precondition	User must have entered valid user ID and Password
Post Condition	School Administrator Menu being displayed.
Algorithm	Begin Send school administrator menu to SystemAdmin Panel class and display. End
Operation/Method	**displayInvalidPasswordMessage()**
Responsibility	To inform the school administrator that wrong user ID or/and Password have been entered
Input Parameter	STAFFlogin, STAFFpwd
Output Parameter	None
Precondition	STAFF class has validated the user ID and Password.
Post Condition	Login failed. School Administrator Login Form being redisplay
Algorithm	Begin Display "Login or Password is incorrect". End
Operation/Method	**displayStudentLoginForm()**
Responsibility	To display Student Login Form on SystemAdmin Panel
Input Parameter	None
Output Parameter	None
Precondition	User has click Student Login on main page.
Post Condition	Student Login form is displayed
Algorithm	Begin Send Student Login Form to SystemAdmin Panel Class. End

Operation/Method	**inputNRICPassword()**
Responsibility	To receive student's NRID and password.
Input Parameter	STDid, STDpwd
Output Parameter	None
Precondition	User must have correct STDid and STDpwd.
Post Condition	User has input STDid and STDpwd.
Algorithm	Begin Send STDid and STDpwd entered to SystemAdmin Panel class. End
Operation/Method	**displayStudentRecord()**
Responsibility	To display record of student that has logged in.
Input Parameter	STDid, STDpwd
Output Parameter	None
Precondition	User has entered valid STDid and STDpwd
Post Condition	Student's personal record being displayed.
Algorithm	Begin Send student record to SystemAdmin Panel.
Operation/Method	**displayInvalidPasswordMessage()**
Responsibility	To inform the student that wrong NRID or Password have been entered
Input Parameter	STDid, STDpwd
Output Parameter	None
Precondition	STUDENT class has validated the user ID and Password.
Post Condition	Login failed. Student Login Form being redisplay
Algorithm	Begin Display "Login or Password is incorrect". End
Operation/Method	**displayEnrollmentForm()**
Responsibility	To display blank student enrollment form on SystemAdmin Panel
Input Parameter	STDid, STDpwd
Output Parameter	None
Precondition	STUDENT class has validated the user ID and Password and student particulars not filled.
Post Condition	Student enrollment form being displayed.
Algorithm	Begin Send Student Enrollment Form to SystemAdmin Panel and display. End
Operation/Method	**pressLogoutButton()**
Responsibility	To let user log out of the system.
Input Parameter	None
Output Parameter	None
Precondition	User has login.
Post Condition	User has logged out and message is sent to SystemAdmin Manager.
Algorithm	Begin Send logout message to SystemAdmin Manager. End
Operation/Method	**displayLogoutStatus()**
Responsibility	To inform the user they have logged out of the system.
Input Parameter	None
Output Parameter	None
Precondition	User has pressed the logout button.
Post Condition	User has logged out of the system.
Algorithm	Begin Display "Thank you for using Student Enrollment System.". End

Operation/Method	**pressChangePasswordButton()**
Responsibility	Let user change password.
Input Parameter	None
Output Parameter	None
Precondition	User has logged in.
Post Condition	Change Password Form is displayed
Algorithm	Begin Display Change Password Form. End
Operation/Method	**displayChangePasswordForm()**
Responsibility	To display Change Password Form on SystemAdminPanel.
Input Parameter	None
Output Parameter	None
Precondition	User has pressed change password button.
Post Condition	Change Password Form is being displayed.
Algorithm	Begin Display Change Password Form in the SystemAdmin Panel. End
Operation/Method	**editPassword()**
Responsibility	To let user replace old password with new password.
Input Parameter	new Password
Output Parameter	None
Precondition	Change Password Form is displayed
Post Condition	User password has changed
Algorithm	Begin Get new password from interface. End
Operation/Method	**displayActivateStaffForm()**
Responsibility	To display Activate Staff Form on SystemAdmin Panel.
Input Parameter	None
Output Parameter	None
Precondition	System Administrator has selected System User Administration button after logged in.
Post Condition	Activate Staff Form is displayed.
Algorithm	Begin Display Activate Staff Form on SystemAdmin Panel. End
Operation/Method	**displayActivateStaffForm()**
Responsibility	To display Activate Staff Form on SystemAdmin Panel.
Input Parameter	None
Output Parameter	None
Precondition	System Administrator has selected System User Administration button after logged in.
Post Condition	Activate Staff Form is displayed.
Algorithm	Begin Display Activate Staff Form on SystemAdmin Panel. End

Operation/Method	**inputActivationDetails()**
Responsibility	To send system user details to SystemAdmin Panel.
Input Parameter	new STAFF details
Output Parameter	None
Precondition	Activate Staff Form is displayed.
Post Condition	New user added and the STAFF record filled.
Algorithm	Begin Send STAFF details to SystemAdmin Panel. End
Operation/Method	**displayStaffList()**
Responsibility	To display a list Staff System User on SystemAdmin Panel.
Input Parameter	None
Output Parameter	None
Precondition	System Administrator has added a new system user or has clicked System User Administration button.
Post Condition	A Staff System User List is displayed.
Algorithm	Begin Display Staff System User List. End
Operation/Method	**displayActivationErrorMessage().**
Responsibility	To display message that system user already exist.
Input Parameter	STAFF
Output Parameter	None
Precondition	System Administrator has added a new system user.
Post Condition	An error message is displayed
Algorithm	Begin Display "System user already activated". End
Operation/Method	**displaySystemUserAdminForm()**
Responsibility	To display SystemUserAdmin Form on SystemAdmin Panel.
Input Parameter	None
Output Parameter	None
Precondition	System Administrator has selected system user administration button after login.
Post Condition	SystemUserAdmin Form is displayed on SystemAdmin Panel.
Algorithm	Begin Display SystemUserAdmin Form on SystemAdmin Panel. End
Operation/Method	**selectStaff()**
Responsibility	To select a system user from a list display on SystemAdmin Panel.
Input Parameter	None
Output Parameter	None
Precondition	SystemUserAdmin Form has been displayed.
Post Condition	Select Staff message is sent to SystemAdmin Panel.
Algorithm	Begin Send selection to SystemAdmin Panel. End

Operation/Method	**displayStaffDetails()**
Responsibility	To display Staff Records of a selected staff.on SystemAdmin Panel.
Input Parameter	STAFF
Output Parameter	None
Precondition	System Administrator has selected the system user by clicking on the STAFFid in the staff list.
Post Condition	Staff record is displayed
Algorithm	Begin Display Staff record End
Operation/Method	**uncheckStatusBox()**
Responsibility	To allow user to remove rights granted to system user
Input Parameter	STAFFsysadmin, STAFFadmin,STAFFprint
Output Parameter	None
Precondition	SystemAdmin Panel has displayed the staff record.
Post Condition	An edited staff record is displayed.
Algorithm	Begin Display edited staff record. End
Operation/Method	**pressSubmitButton()**
Responsibility	To send deactivation message to SystemAdmin Panel.
Input Parameter	STAFF
Output Parameter	None
Precondition	System Administrator has unchecked status box.
Post Condition	Deactivate message is sent to SystemAdmin Panel.
Algorithm	Begin Sent deactivate message to SystemAdminPanel. End
Operation/Method	**displayStudentAccountAdminForm()**
Responsibility	To display StudentAccountAdminForm on SystemAdmin Panel.
Input Parameter	None
Output Parameter	None
Precondition	System Administrator has choosen student account admin after login.
Post Condition	StudentAccountAdminForm is displayed on SystemAdmin Panel
Algorithm	Begin Display StudentAccountAdminForm. End
Operation/Method	**inputStudNRID()**
Responsibility	To receive student NRID to SystemAdmin Panel.
Input Parameter	STDid
Output Parameter	None
Precondition	StudentAccountAdminForm is displayed.
Post Condition	Student NRID is received by SystemAdmin Panel
Algorithm	Begin Send STDid to SystemAdmin Panel. End

Operation/Method	**displayStudentAccount()**
Responsibility	To display student account on SystemAdmin Panel.
Input Parameter	STDid
Output Parameter	None
Precondition	User has input student NRID.
Post Condition	Student account is displayed by SystemAdmin Panel
Algorithm	Begin Display student account on SystemAdmin Panel. End
Operation/Method	**inputStudActivationDetails()**
Responsibility	To send student activation details to SystemAdmin Panel.
Input Parameter	STDid
Output Parameter	None
Precondition	StudentAccountAdmin Form is displayed.
Post Condition	New Student user account form is filled.
Algorithm	Begin Send student activation details to SystemAdmin Panel. End

4.1.2 StudentEnrollment Panel Class

Class Type	Boundary class
Responsibility	To communicate with user of the system, which provide interface that interact with user.
Attributes	None
Operation/Method	**displayEnrollmentForm()**
Responsibility	To display blank student enrollment form on StudentEnrollment Panel
Input Parameter	STDid, STDpwd
Output Parameter	None
Precondition	Student has login using student account and student particulars not filled.
Post Condition	Student enrollment form being displayed.
Algorithm	Begin Send Student Enrollment Form to StudentEnrollment Panel and display. End
Operation/Method	**inputStudentParticulars()**
Responsibility	To send student particulars input to StudentEnrollment Panel
Input Parameter	STUDENT
Output Parameter	None
Precondition	Student enrollment form has been displayed.
Post Condition	Student particulars are filled.
Algorithm	Begin Send input student particulars to StudentEnrollment Panel. End
Operation/Method	**showSavedParticulars()**
Responsibility	To display record of student that has just been submitted.
Input Parameter	STUDENT
Output Parameter	STUDENT
Precondition	User has submitted personal particulars via enrollment form.
Post Condition	Student's personal record being displayed.
Algorithm	Begin Send student record to StudentEnrollment Panel.

Operation/Method	**displayStudentRecord()**
Responsibility	To display student record search by school administrator on the StudentEnrollment Panel
Input Parameter	Student NRID
Output Parameter	None
Precondition	Student record is filled.
Post Condition	Student record is displayed.
Algorithm	Begin Display student record on StudentEnrollment Panel. End
Operation/Method	**pressDeleteButton()**
Responsibility	To send delete student record message to StudentEnrollment Panel.
Input Parameter	None
Output Parameter	None
Precondition	Student record is displayed
Post Condition	Send delete student record message to StudentEnrollment Panel.
Algorithm	Begin Display student record on StudentEnrollment Panel. End
Operation/Method	**displayConfirmationPrompt()**
Responsibility	To display on StudentEnrollment Panel the prompt to confirm the deletion of student record.
Input Parameter	None
Output Parameter	None
Precondition	User has pressed delete button..
Post Condition	Confirmation prompt is displayed.
Algorithm	Begin Display confirmation prompt on StudentEnrollment Panel. End
Operation/Method	**pressOkButton()**
Responsibility	Let user confirm deletion of satudent record to the StudentEnrollment Panel
Input Parameter	None
Output Parameter	None
Precondition	Confirmation prompt to deletion is displayed.
Post Condition	Delete student record.message is sent to StudentEnrollment Panel
Algorithm	Begin Send deletion confirmation to StudentEnrollment Panel. End
Operation/Method	**showEmptyStudentRecord()**
Responsibility	To display empty student record after user has deleted the record.
Input Parameter	None
Output Parameter	None
Precondition	Deletion of student record has been confirmed by user..
Post Condition	Empty student record is displayed.
Algorithm	Begin Display empty student record on StudentEnrollment Panel. End

Operation/Method	**editStudentRecord()**
Responsibility	To send edited student record to the StudentEnrollment Panel
Input Parameter	Edited student record.
Output Parameter	None
Precondition	Student record is displayed.
Post Condition	Edited student record is send to StudentEnrollment Panel.
Algorithm	Begin Send edited student record to StudentEnrollment Panel. End
Operation/Method	**showEditedStudentRecord()**
Responsibility	To display edited student record after user has submitted the edited record.
Input Parameter	Edited student record
Output Parameter	None
Precondition	Student record has been edited and submitted by user..
Post Condition	Edited student record is displayed.
Algorithm	Begin Display edited student record on StudentEnrollment Panel. End

4.1.3 ClassAdmin Panel Class

Class Type	Boundary class
Responsibility	To communicate with user of the system, which provide interface that interact with user.
Attributes	None
Operation/Method	**displayClassAdminForm()**
Responsibility	To display ClassAdmin form on ClassAdmin Panel
Input Parameter	None
Output Parameter	None
Precondition	School administrator has login using valid user ID and Password and has choosen class admin button.
Post Condition	ClassAdmin form being displayed.
Algorithm	Begin Send ClassAdmin Form to ClassAdmin Panel and display. End
Operation/Method	**inputClassParticulars()**
Responsibility	To send class particulars input to ClassAdmin Panel
Input Parameter	CLASS
Output Parameter	None
Precondition	Generate class form has been displayed.
Post Condition	Class particulars are filled.
Algorithm	Begin Send input class particulars to ClassAdmin Panel. End
Operation/Method	**displayClassList()**
Responsibility	To display a list of class including one that has just been submitted.
Input Parameter	CLASS
Output Parameter	None
Precondition	User has input class particulars via add class form.
Post Condition	A list of classes being displayed.
Algorithm	Begin Send Class record to ClassAdmin Panel. End

Operation/Method	**inputStudentNRIC()**
Responsibility	To send student ID to ClassAdmin Panel
Input Parameter	Student NRID
Output Parameter	None
Precondition	Student record is filled.
Post Condition	Student NRIC is sent to the ClassAdmin Panel.
Algorithm	Begin Send student NRIC to ClassAdmin Panel. End
Operation/Method	**displayStudentList()**
Responsibility	To display student in the class including the new submission on ClassAdmin Panel.
Input Parameter	Updated student list in class 1
Output Parameter	None
Precondition	Student has been added to class.
Post Condition	A list of student in class is displayed on the ClassAdmin Panel.
Algorithm	Begin Display student list on the ClassAdmin Panel. End
Operation/Method	**showErrorMessage()**
Responsibility	To display error message that student record not found in the add class member process.
Input Parameter	Student NRID
Output Parameter	None
Precondition	School administrator has pressed add class member button.
Post Condition	An error message is displayed.
Algorithm	Begin Display error message on the ClassAdmin Panel. End

4.1.4 SearchStudent Panel Class

Class Type	Boundary class
Responsibility	To communicate with user of the system, which provide interface that interact with user.
Attributes	None
Operation/Method	**displaySearchStudentForm()**
Responsibility	To display SearchStudent form on SearchStudent Panel
Input Parameter	None
Output Parameter	None
Precondition	School administrator has choosen ManageStudentRecord button.
Post Condition	SearchStudent form being displayed.
Algorithm	Begin Send SearchStudent Form to SearchStudent panel and display. End

Operation/Method	**inputSearchStudentDetails()**
Responsibility	To send search student particulars input to SearchStudent Panel
Input Parameter	STDid, STDname, STDadmissionNum, STDstampdate
Output Parameter	None
Precondition	Student enrollment form has been displayed.
Post Condition	Student particulars are filled.
Algorithm	Begin Send input search student particulars to SearchStudent Panel. End
Operation/Method	**displayStudentRecord()**
Responsibility	To display student record search by school administrator on the SearchStudent Panel
Input Parameter	STUDENT
Output Parameter	None
Precondition	User has submitted search student details.
Post Condition	Student record is displayed.
Algorithm	Begin Display student record on SearchStudent Panel. End
Operation/Method	**displayErrorMessage()**
Responsibility	To display error message on the SearchStudent Panel if student record not found.
Input Parameter	None
Output Parameter	None
Precondition	User has submitted search student details.
Post Condition	Error message is displayed.
Algorithm	Begin Display "No record found". End

4.1.5 Print Panel Class

Class Type	Boundary class
Responsibility	To communicate with user of the system, which provide interface that interact with user.
Attributes	None
Operation/Method	**displayIndvStudReport()**
Responsibility	To display IndvStudReport on Print Panel
Input Parameter	None
Output Parameter	None
Precondition	School administrator has enter search for individual student.
Post Condition	IndvStudentReport is displayed.
Algorithm	Begin Display IndvStudReport on Print Panel. End

Operation/Method	**pressPrintButton()**
Responsibility	To send print report message to the Print Panel
Input Parameter	None
Output Parameter	None
Precondition	Student Report has been displayed.
Post Condition	Print report message is sent to Print Panel.
Algorithm	Begin Send print report message to print Panel. End
Operation/Method	**printIndvStudReport()**
Responsibility	To print individual student report
Input Parameter	IndvStudReport
Output Parameter	None
Precondition	User has pressed the print button on Print Panel.
Post Condition	Individual student report printed.
Algorithm	Begin Print individual student report. End
Operation/Method	**pressCloseButton()**
Responsibility	To let user send message to the Print Panel so as to exit the print module.
Input Parameter	None
Output Parameter	None
Precondition	Student report has been displayed.
Post Condition	Close print module message is sent to Print Panel.
Algorithm	Begin Send close print module message. End
Operation/Method	**displayPrintStudentForm()**
Responsibility	To display PrintStudent form on Print Panel
Input Parameter	None
Output Parameter	None
Precondition	School administrator has log in to Print Module.
Post Condition	PrintStudent form being displayed.
Algorithm	Begin Send PrintStudent Form to Print panel and display. End
Operation/Method	**selectClass()**
Responsibility	To let user select class so as to access its student report.
Input Parameter	None
Output Parameter	None
Precondition	Class list has been displayed on Print Panel.
Post Condition	Selection is sent to Print Panel.
Algorithm	Begin Send selection to Print Panel. End
Operation/Method	**displayClassStudentReport()**
Responsibility	To display student report of a whole class on Print Panel.
Input Parameter	None
Output Parameter	None
Precondition	User has selected class of which students are to be displayed.
Post Condition	ClassStudentReport is displayed on Print Panel
Algorithm	Begin Display ClassStudentReport on Print Panel End

Operation/Method	**printClassStudReport()**
Responsibility	To print student report of the whole class selected
Input Parameter	ClassStudReport
Output Parameter	None
Precondition	User has pressed the print button on Print Panel.
Post Condition	Student report for the whole class printed.
Algorithm	Begin Print ClassStudReport. End
Operation/Method	**displayPrintStudentByYearForm()**
Responsibility	To display PrintStudentByYear form on Print Panel.
Input Parameter	None
Output Parameter	None
Precondition	School administrator has log in to Print Module and choose to print by student intake year.
Post Condition	PrintStudentByYear form being displayed.
Algorithm	Begin Send PrintStudentByYear Form to Print Panel and display. End
Operation/Method	**inputPrintByIntakeYearDetails()**
Responsibility	To send PrintByIntakeYear particulars input to Print Panel.
Input Parameter	IntakeYear
Output Parameter	None
Precondition	PrintStudentByIntakeYear form has been displayed.
Post Condition	PrintByIntakeYear particulars are filled.
Algorithm	Begin Send input PrintByIntakeYear partrticulars to Print Panel. End
Operation/Method	**displayYearStudentReport()**
Responsibility	To display student report of a whole intake year on the Print Panel
Input Parameter	IntakeYear
Output Parameter	None
Precondition	User has entered intake year of which students records are to be displayed.
Post Condition	YearStudentReport is displayed on Print Panel.
Algorithm	Begin Display YearStudentReport on Print Panel. End
Operation/Method	**printYearStudReport()**
Responsibility	To print student reports of the whole selected intake year.
Input Parameter	YearStudReport
Output Parameter	None
Precondition	User has pressed the print button on Print Panel.
Post Condition	Student report for the whole intake year is printed.
Algorithm	Begin Print YearStudReport. End

Operation/Method	**inputClassDetail()**
Responsibility	To send class particulars input to Print Panel.
Input Parameter	CLASSid, CLASSname, CLASSteacher, CLASSyear
Output Parameter	None
Precondition	PrintStudent form has been displayed.
Post Condition	Class search particulars are filled.
Algorithm	Begin Send input search class partrticulars to Print Panel. End
Operation/Method	**displayClassList()**
Responsibility	To display class list search by school administrator on the Print Panel.
Input Parameter	None
Output Parameter	None
Precondition	User has submitted search class details.
Post Condition	Class list is displayed.
Algorithm	Begin Display class list on Print Panel. End
Operation/Method	**printClassList()**
Responsibility	To print student name list of the selected class.
Input Parameter	ClassList
Output Parameter	None
Precondition	User has pressed the print button on Print Panel.
Post Condition	Class list for the selected class is printed.
Algorithm	Begin Print ClassList. End

4.2 Application Layer Package

4.2.1 SystemAdmin Manager Class

Class Type	Control class
Responsibility	To communicate between one entity and another entity
Attributes	None
Operation/Method	**sendStaffIdPassword()**
Responsibility	To send school administrator's user ID and Password to SystemAdmin Manager.
Input Parameter	STAFFlogin, STAFFpwd
Output Parameter	None
Precondition	User has pressed Login button.
Post Condition	User. ID and Password is sent to SystemAdmin Manager
Algorithm	Begin Send STAFFlogin and STAFFpwd entered to SystemAdmin Manager class. End

Operation/Method	**sendValidatedMessage()**
Responsibility	To send user ID and Password validated message to SystemAdmin Manager.
Input Parameter	None
Output Parameter	None
Precondition	The STAFF class has validated user ID and Password
Post Condition	User ID and Password validated message is sent to SystemAdmin Manager.
Algorithm	Begin Send user ID and Password validated message to SystemAdmin Manager End
Operation/Method	**sendInvalidPasswordMessage()**
Responsibility	To inform the school administrator that wrong user ID or Password have been entered
Input Parameter	STAFFlogin, STAFFpwd
Output Parameter	None
Precondition	STAFF class has failed to validated the user ID and Password.
Post Condition	Invalid user ID or Password message is sent to SystemAdmin Manager.
Algorithm	Begin Send message"Login or Password is incorrect". End
Operation/Method	**sendNRICPassword()**
Responsibility	To send students user's NRID and Password to SystemAdmin Manager.
Input Parameter	STUDlid, STDpwd
Output Parameter	None
Precondition	User has pressed Login button.
Post Condition	User has pressed Login button.
Algorithm	Begin Send STUDlid, STDpwd entered to SystemAdmin Manager class. End
Operation/Method	**sendValidatedMessage()**
Responsibility	To send Student NRID and Password validated message to SystemAdmin Manager.
Input Parameter	None
Output Parameter	None
Precondition	The STUDENT class has validated Student NRID and Password
Post Condition	Student NRID and Password validated message is sent to SystemAdmin Manager.
Algorithm	Begin Send Student NRID and Password validated message to SystemAdmin Manager End
Operation/Method	**sendInvalidPasswordMessage()**
Responsibility	To inform the student that wrong NRID or Password have been entered
Input Parameter	STUDid, STUDpwd
Output Parameter	None
Precondition	STUDENT class has failed to validated the user ID and Password.
Post Condition	Invalid user ID or Password message is sent to SystemAdmin Manager.
Algorithm	Begin Send message"Login or Password is incorrect" to SystemAdmin Manager. End

Operation/Method	**sendLogoutMessage()**
Responsibility	To send log out message to System Admin Manager.
Input Parameter	LogOutMessage
Output Parameter	None
Precondition	User has pressed Logout Button.
Post Condition	Logout message is sent to SystemAdmin Manager.
Algorithm	Begin Send logout message"Tkhank You for using Student Enrollment System" to SystemAdmin Manager. End
Operation/Method	**sendPasswordChangedMessage()**
Responsibility	Send password has changed message to SystemAdmin Manager.
Input Parameter	None
Output Parameter	None
Precondition	Password has been validated.
Post Condition	Password changed message is sent to SystemAdmin Manager.
Algorithm	Begin Send password changed message to SystemAdmin Manager. End
Operation/Method	**sendActivationDetails()**
Responsibility	To send system user details to SystemAdmin Manager.
Input Parameter	new STAFF details
Output Parameter	None
Precondition	New user added and the STAFF record filled and submitted.
Post Condition	Activation details are sent to SystemAdmin Manager.
Algorithm	Begin Send STAFF details to SystemAdmin Manager. End
Operation/Method	[illegible]
Responsibility	To send staff added message to SystemAdmin Manager.
Input Parameter	StaffList
Output Parameter	None
Precondition	New Staff added to the STAFF class.
Post Condition	Staff added message is sent to SystemAdmin Manager
Algorithm	Begin Send staff added message to SystemAdmin Manager. End
Operation/Method	**sendErrorMessage().**
Responsibility	To send error message that system user already exist to SystemAdmin Manager
Input Parameter	None
Output Parameter	None
Precondition	System Administrator has submitted a new system user.
Post Condition	An error message is displayed
Algorithm	Begin Send message "System user already activated" to SystemAdmin Manager. End

Operation/Method	**sendStaffID()**
Responsibility	To send Staff ID to SystemAdmin Manager.
Input Parameter	STAFFid
Output Parameter	None
Precondition	System Administrator has selected system user listed in SystemAdmin panel.
Post Condition	STAFFid is sent to SystemAdmin Manager.
Algorithm	Begin StaffID is sent to SystemAdmin Manager. End
Operation/Method	**sendStaffDetails()**
Responsibility	To send Staff Records of a selected staff to SystemAdmin Manager.
Input Parameter	STAFF
Output Parameter	None
Precondition	The STAFF class has got the Staff record selected.
Post Condition	Staff record is sent to SystemAdmin Manager.
Algorithm	Begin Send staff record to SystemAdmin Manager. End
Operation/Method	**sendDeactStaffDetails()**
Responsibility	To send new user status to SystemAdmin Manager.
Input Parameter	editedStaffDetails
Output Parameter	None
Precondition	SystemAdministrator has made and submitted changes to user rights.
Post Condition	An edited staff record is sent to SystemAdmin Manager.
Algorithm	Begin Send edited staff record to SystemAdmin Manager. . End
Operation/Method	**sendStudActivationDetails()**
Responsibility	To send student activation details to SystemAdmin Manager.
Input Parameter	STDid, STDactive
Output Parameter	None
Precondition	Add Student User Form has been filled and submitted.
Post Condition	New StudActivationDetails is sent to SystemAdmin Manager.
Algorithm	Begin Send student activation details to SystemAdmin Manager. End
Operation/Method	**sendStudentAddedMessage()**
Responsibility	To send student added message to SystemAdmin Manager.
Input Parameter	New Student Account
Output Parameter	None
Precondition	New Student Account has been added by STUDENT class.
Post Condition	Student added message is sent to SystemAdmin Manager.
Algorithm	Begin Send student added message to SystemAdmin Manager. End
Operation/Method	**sendErrorMessage()**
Responsibility	To send error message in adding new student user to SystemAdmin Manager.
Input Parameter	New Student Account
Output Parameter	None
Precondition	Student Account has already existed.
Post Condition	Add student error message is sent to SystemAdmin Manager.
Algorithm	Begin Send error message to SystemAdmin Manager. End

Operation/Method	**sendSearchMessage()**
Responsibility	To send search student message to SystemAdmin Manager.
Input Parameter	STDid
Output Parameter	None
Precondition	System Administrator has enter and submitted student NRID in SystemAdmin Panel.
Post Condition	Search student message is sent to SystemAdmin Manager.
Algorithm	Begin Sent search student message to SystemAdmin Manager. End
Operation/Method	**sendStudentAccount().**
Responsibility	To send StudentAccount to SystemAdmin Manager.
Input Parameter	STDid, STDactive
Output Parameter	None
Precondition	The STUDENT class has got the student account selected by System Administrator.
Post Condition	The Student Account is sent to SystemAdmin Manager
Algorithm	Begin Send StudentAccount. End
Operation/Method	**sendStudDeactMessage().**
Responsibility	To send deactivate student message to SystemAdmin Manager.
Input Parameter	New Student Account
Output Parameter	None
Precondition	StudentAccount has been edited and submitted.
Post Condition	Deactivate student message is send to SystemAdmin Manager
Algorithm	Begin Send deactivate student message to SystemAdmin Manager. End
Operation/Method	**sendStudentAccountUpdatedMessage().**
Responsibility	To send student account to SystemAdmin Manager.
Input Parameter	Student Account
Output Parameter	None
Precondition	Syatem Administrator has submitted edited student account.
Post Condition	Student account updated message is sent to SystemAdmin Manager.
Algorithm	Begin Send student account updated message to SystemAdmin Manager. End

4.2.2 StudentEnrollment Manager Class

Class Type	Control class
Responsibility	To communicate between one entity and another entity
Attributes	None
Operation/Method	**SendAddRecordMessage()**
Responsibility	To send student particulars to StudentEnrollment Manager.
Input Parameter	STUDENT
Output Parameter	None
Precondition	Student enrollment form has been filled and submitted.
Post Condition	Student particulars is sent to StudentEnrollment Manager.
Algorithm	Begin Send Student Records to Student Enrollment Manager. End

Operation/Method	**sendSaveMessage()**
Responsibility	To send message to the StudentEnrollment Manager that student particulars entered has been saved.
Input Parameter	STUDENT
Output Parameter	None
Precondition	Student enrollment form has been filled and submitted.
Post Condition	Record saved message is sent to StudentEnrollment Manager.
Algorithm	Begin Send record saved message to StudentEnrollment Manager. End
Operation/Method	**sendDeleteMessage()**
Responsibility	To sent delete student record message to StudentEnrollment Manager.
Input Parameter	None
Output Parameter	None
Precondition	User has pressed delete record button on StudentEnrololment Panel.
Post Condition	Delete student record message is sent to the StudentEnrollment Manager.
Algorithm	Begin Send delete student record message to StudentEnrollment Manager. End
Operation/Method	**sendConfirmDeleteMessage()**
Responsibility	To send confirm delete student record message to the StudentEnrollment Manager.
Input Parameter	Student NRID
Output Parameter	None
Precondition	Student record is displayed.
Post Condition	Confirm delete message is sent to StudentEnrollment Manager.
Algorithm	Begin Send confirm delete student record message. End
Operation/Method	**sendNoDeleteMessage()**
Responsibility	To send not to delete student record message to the StudentEnrollment Manager.
Input Parameter	None
Output Parameter	None
Precondition	Student record is displayed.
Post Condition	Not to delete message is sent to StudentEnrollment Manager.
Algorithm	Begin Send not to delete student record message. End
Operation/Method	**sendUpdatedRecord()**
Responsibility	To send edited student record to the StudentEnrollment Manager
Input Parameter	Edited student record.
Output Parameter	None
Precondition	Student record has been edited and submitted.
Post Condition	Edited student record is send to StudentEnrollment Manager.
Algorithm	Begin Send edited student record to StudentEnrollment Manager. End

Operation/Method	**sendStudentUpdatedMessage()**
Responsibility	To send edited student record to StudentEnrollment Manager.
Input Parameter	Edited student record
Output Parameter	None
Precondition	Student record has been edited and submitted by user..
Post Condition	StudentRecordUpdated message is sent to StudentEnrollment Manager.
Algorithm	Begin Send student record updated message to StudentEnrollment Manager. End

4.2.3 ClassAdmin Manager Class

Class Type	Control class
Responsibility	To communicate between one entity and another entity.
Attributes	None
Operation/Method	**sendClassParticulars()**
Responsibility	To send class particulars input to ClassAdmin Manager
Input Parameter	CLASS
Output Parameter	None
Precondition	Class particulars are filled and submitted.
Post Condition	Class particulars are sent to ClassAdmin Manager
Algorithm	Begin Send input class particulars to ClassAdmin Manager. End
Operation/Method	**sendClassCreatedMessage()**
Responsibility	To send class created message to ClassAdmin Manager.
Input Parameter	ClassList
Output Parameter	None
Precondition	CLASS class has been updated.
Post Condition	Class created message is sent to ClassAdmin Manager.
Algorithm	Begin Send class created message to ClassAdmin Manager. End
Operation/Method	**sendStudentNRIC()**
Responsibility	To send student ID to ClassAdmin Manager
Input Parameter	Student NRID
Output Parameter	None
Precondition	Student NRIC is entered to the ClassAdminForm and submitted.
Post Condition	Student NRIC is sent to ClassAdmin Manager
Algorithm	Begin Send student NRIC to ClassAdmin Manager. End
Operation/Method	**sentStudentRecords()**
Responsibility	To send student records in the class including the new submission to the ClassAdmin Manager.
Input Parameter	STDid
Output Parameter	None
Precondition	Student record exist in the CLASS class.
Post Condition	STDid is sent to ClassAdmin Manager.
Algorithm	Begin Send Student ID to ClassAdmin Manager. End

Operation/Method	**sendErrorMessage()**
Responsibility	To send error message to the ClassAdmin Manager that student record not found.
Input Parameter	None
Output Parameter	None
Precondition	STUDENT class failed to find relevant record.
Post Condition	An error message is sent to ClassAdmin Manager.
Algorithm	Begin Send error message to the ClassAdmin Manager. End

4.2.4 SearchStudent Manager Class

Class Type	Control class
Responsibility	To communicate between one entity and another entity
Attributes	None
Operation/Method	**sendSearchStudentDetails()**
Responsibility	To send SearchStudent details to the SearchStudent Manager
Input Parameter	STDid, STDname, STDadmissionNum, STDstampdate
Output Parameter	None
Precondition	School administrator has enter and submitted the search criteria.
Post Condition	SearchStudent message is sent to SearchStudent Manager.
Algorithm	Begin Send SearchStudent message to SearchStudent Manager. End
Operation/Method	**sendStudentRecord()**
Responsibility	To send student record search by school administrator to the SearchStudent Manager
Input Parameter	STUDENT
Output Parameter	None
Precondition	Student record been fetched by STUDENT class.
Post Condition	Student record is sent to SearchStudent Manager.
Algorithm	Begin Send student record to the SearchStudent Manager. End
Operation/Method	**sendErrorMessage()**
Responsibility	To send error message to the SearchStudent Manager if student record not found.
Input Parameter	None
Output Parameter	None
Precondition	STUDENT class failed to find the selected student.
Post Condition	Error message is sent to SearchStudent Manager.
Algorithm	Begin Send error message to SearchStudent Manager." End

4.2.5 Print Manager Class

Class Type	Control class
Responsibility	To communicate between one entity and another entity.
Attributes	None
Operation/Method	**sendPrintIndvStudReportMessage()**
Responsibility	To send PrintIndvStudReport messageto Print Manager
Input Parameter	IndvStudReport
Output Parameter	None
Precondition	IndvStudentReport is displayed.
Post Condition	IndvStudentReport is sent to the Print Manager
Algorithm	Begin Send PrintIndvStudReport message to Print Manager. End
Operation/Method	**sendCloseButton()**
Responsibility	To send exit Print Module message to the Print Manager.
Input Parameter	None
Output Parameter	None
Precondition	Close print module message is sent to Print Panel.
Post Condition	Exit Print Module message is sent to the Print Manager.
Algorithm	Begin Send close print module message to Print Manager. End
Operation/Method	**sendClassDetails()**
Responsibility	To send class details to Print Manager.
Input Parameter	CLASS
Output Parameter	None
Precondition	Print student form has been filled and submitted.
Post Condition	Class details are sent to Print Manager.
Algorithm	Begin Send Class details to Print Manager. End
Operation/Method	**sendClassList()**
Responsibility	To send a list of available classes to the Print Manager.
Input Parameter	ClassList
Output Parameter	None
Precondition	Selected class list has been fetched from CLASS class.
Post Condition	Class list is send to Print Manager.
Algorithm	Begin Send class list to Print Manager. End
Operation/Method	**sendCLASSid()**
Responsibility	To send CLASSid of selected class to the Print Manager.
Input Parameter	CLASSid
Output Parameter	None
Precondition	User has selected the class which its report is to be printed.
Post Condition	Selected CLASSid is sent to the Print Manager.
Algorithm	Begin Send CLASSid to the Print Manager. End

Operation/Method	**sendStudentId()**
Responsibility	To send selected StudentId to the Print Manager.
Input Parameter	STDid
Output Parameter	None
Precondition	The CLASSTD class has provide the StudentId based on CLASSid entered.
Post Condition	Selected STDid is sent to the Print Manager.
Algorithm	Begin Send STDid to the Print Manager. End
Operation/Method	**sendStudentRecord()**
Responsibility	To send student records of the selected class to the Print Manager.
Input Parameter	STUDENT
Output Parameter	None
Precondition	Selected student record has been fetched from STUDENT class.
Post Condition	ClassStudentReport is sent to the Print Manager.
Algorithm	Begin Send ClassStudentReport to the Print Manager. End
Operation/Method	**sendPrintClassStudReport()**
Responsibility	To send PrintClassStudReport message to the Print Manager.
Input Parameter	None
Output Parameter	None
Precondition	User has pressed Print button on the Print Panel.
Post Condition	Print ClassStudReport message is sent to the Print Manager.
Algorithm	Begin Send print ClassStudReport message to the Print Manager. End
Operation/Method	**sendPrintByIntakeYearDetails()**
Responsibility	To send PrintByIntakeYear particulars input to Print Manager
Input Parameter	IntakeYear
Output Parameter	None
Precondition	PrintByIntakeYear particulars are filled.
Post Condition	PrintByIntakeYearDetails are sent to Print Manager.
Algorithm	Begin Send PrintByIntakeYear particulars to Print Manager. End
Operation/Method	**sendPrintYearStudentReportMessage()**
Responsibility	To send PrintYearStudentReport Message to the Print Manager.
Input Parameter	None
Output Parameter	None
Precondition	User has pressed the print button.
Post Condition	PrintYearStudentReport message is sent to the Print Manager.
Algorithm	Begin Send PrintYearStudentReport message to the Print Manager End
Operation/Method	**sendPrintClassListMessage()**
Responsibility	To send PrintClassList message to the Print Manager.
Input Parameter	None
Output Parameter	None
Precondition	User has pressed the print button on Print Panel.
Post Condition	PrintClassList message is sent to the Print Manager.
Algorithm	Begin Send PrintClassList message. End

4.3 Storage Layer Package

4.3.1 STAFF Class

Class Type	Entity class
Responsibility	To manage staff user record.
Attributes	STAFFid: String STAFFname: String STAFFlogin: String STAFFpwd: String STAFFpost: String STAFFstatus: String STAFFsysadmin: Boolean STAFFadmin: Boolean STAFFprint: Boolean STAFFtimestamp: Date
Operation/Method	**validateStaffIdPassword()**
Responsibility	To validate staff user ID and password
Input Parameter	STAFFlogin, STAFFpwd
Output Parameter	None
Precondition	Staff press login button.
Post Condition	Staff user ID and password is validated.
Algorithm	Begin Validate StaffId and Password. End
Operation/Method	**updatePassword()**
Responsibility	To allow staff to change own password.
Input Parameter	New pasasword
Output Parameter	None
Precondition	Overwrite old password.
Post Condition	Staff password has changed.
Algorithm	Begin Validate new password.. End
Operation/Method	**addStaffUsers()**
Responsibility	To add new staff record to STAFF class..
Input Parameter	CLASS
Output Parameter	None
Precondition	Staff details has been filled and submitted.
Post Condition	New Staff user added.
Algorithm	Begin Add new Staff to STAFF. End
Operation/Method	**getStaffDetails()**
Responsibility	To get the Staff record from STAFF class.
Input Parameter	STAFFid
Output Parameter	None
Precondition	User has input STAFFid or click on one.
Post Condition	Staff details are displayed.
Algorithm	Begin Receive STAFFid and get staff record. End

4.3.2 STUDENT class

Class Type	Entity class
Responsibility	To manage personal record of student.
Attributes	STDid: String STDname: String STDpwd: String STDsex: String STDreligion: String STDrace: String STDbirthcert: String STDdob: Date STDiniclass: String STDfathername: String STDfatheradd: String STDguardianname: String STDguardianadd: String STDguardiantype: String STDstampdate: Date STDadmissionNum: Numeric STDsijil: Boolean STDcertnum: String STDleavedate: Date STDleaveReason: String STDactive: Boolean
Operation/Method	**validateStudentPassword()**
Responsibility	To validate student NRID and password
Input Parameter	STDid, STDpwd
Output Parameter	None
Precondition	Student press login button.
Post Condition	Student NRID and password is validated.
Algorithm	Begin Validate student's NRID and Password. End
Operation/Method	**updatePassword()**
Responsibility	To allow student to change own password.
Input Parameter	New pasasword
Output Parameter	None
Precondition	Overwrite old password.
Post Condition	Student password has changed.
Algorithm	Begin Validate new password.. End
Operation/Method	**sendErrorMessage()**
Responsibility	To send error message if student record not found.
Input Parameter	STDid
Output Parameter	None
Precondition	Search student criteria has been entered and submitted.
Post Condition	Error message is displayed.
Algorithm	Begin Send error message. End

Operation/Method	**addStudentUser()**
Responsibility	To add new student user record to STUDENT class..
Input Parameter	STDid
Output Parameter	None
Precondition	System Administrator has invoke add student user function.
Post Condition	New Student user added.
Algorithm	Begin Add new Student to STUDENT. End
Operation/Method	**getStudentAccount()**
Responsibility	To get the Student record from STUDENT class.
Input Parameter	STDid
Output Parameter	None
Precondition	User has input STDid or click on one.
Post Condition	Student accounts are displayed.
Algorithm	Begin Receive STDid and get student account. End
Operation/Method	**updateStudentRecord()**
Responsibility	To keep the latest student record after editing.
Input Parameter	STDid
Output Parameter	None
Precondition	Student records are displayed
Post Condition	Student record updated.
Algorithm	Begin Overwrites old student record with new record. End
Operation/Method	**addStudentRecord()**
Responsibility	To fill the student record with particulars.
Input Parameter	STUDENT
Output Parameter	None
Precondition	Student has input NRID and NRID as default password. Blank enrollment form is displayed.
Post Condition	Student record is filled.
Algorithm	Begin Add student particulars to record. End
Operation/Method	**deleteStudentRecord()**
Responsibility	To delete the student record from STUDENT.
Input Parameter	STDid
Output Parameter	None
Precondition	School Administrator enter Student NRID to display student record.
Post Condition	Student record is deleted.
Algorithm	Begin Replace all nullable fields with null. End

4.3.3 CLASS class

Class Type	Entity class
Responsibility	To manage class record.
Attributes	CLASSid: String CLASSname: String CLASSteacher: String CLASSyear: String
Operation/Method	**updateCLASS()**
Responsibility	To add, edit and delete class record.
Input Parameter	CLASS
Output Parameter	None
Precondition	Class details are filled
Post Condition	CLASS updated.
Algorithm	Begin Class records added, deleted or edited. End
Operation/Method	**getClassList()**
Responsibility	To get a list of available classes.
Input Parameter	CLASSid, CLASSname,CLASSteacher, CLASSyear
Output Parameter	None
Precondition	Class search criteria has been entered and submitted.
Post Condition	Class list is displayed.
Algorithm	Begin Receive class details and get class list. End
Operation/Method	**sendErrorMessage()**
Responsibility	To send error message if class record not found.
Input Parameter	CLASSid, CLASSname,CLASSteacher, CLASSyear
Output Parameter	None
Precondition	Class search criteria has been entered and submitted.
Post Condition	Error message is displayed.
Algorithm	Begin Send error message. End

4.3.4 CLASSTD class

Class Type	Entity class
Responsibility	To manage STUDENT class and CLASS class.
Attributes	CLASSid: Numeric STDid: String Cstimestamp: Date
Operation/Method	**relayCLASSid()**
Responsibility	To receive CLASSid and furnish print manager with STDid of the class.
Input Parameter	CLASSid
Output Parameter	STDid
Precondition	User has entered the CLASSid.
Post Condition	STDid(s) are sent to Print Manager.
Algorithm	Begin Receive CLASSid and send STDid.. End

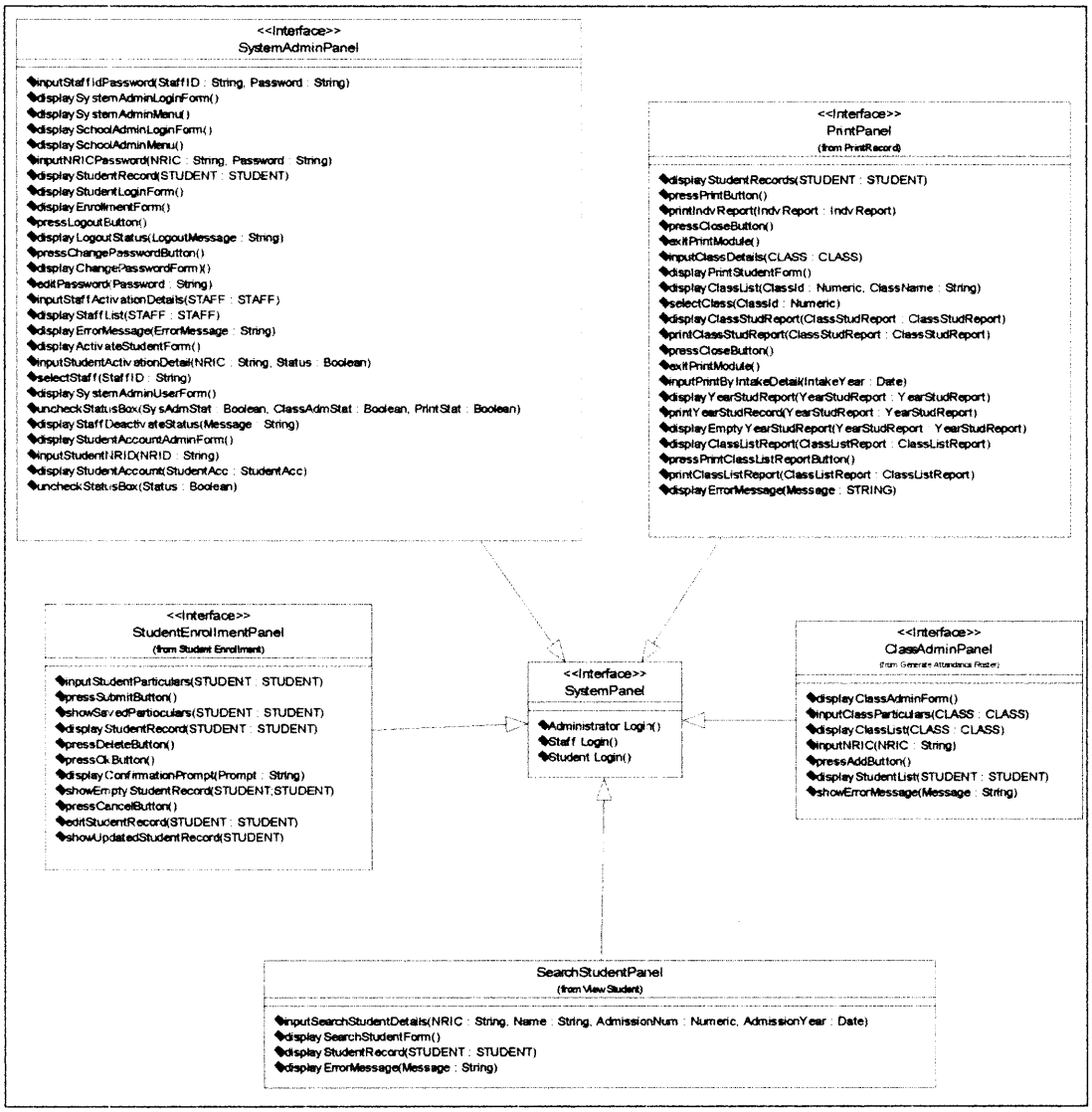
Operation/Method	**updateCLASSTD()**
Responsibility	To store STDid(s) to its associated CLASSid when student is added to the class.
Input Parameter	STDid
Output Parameter	None
Precondition	ClassAdmin Manager has received the STDid.
Post Condition	CLASSTD is updated.
Algorithm	Begin Receive STDid and update CLASSTD. End

5.0 NOTES

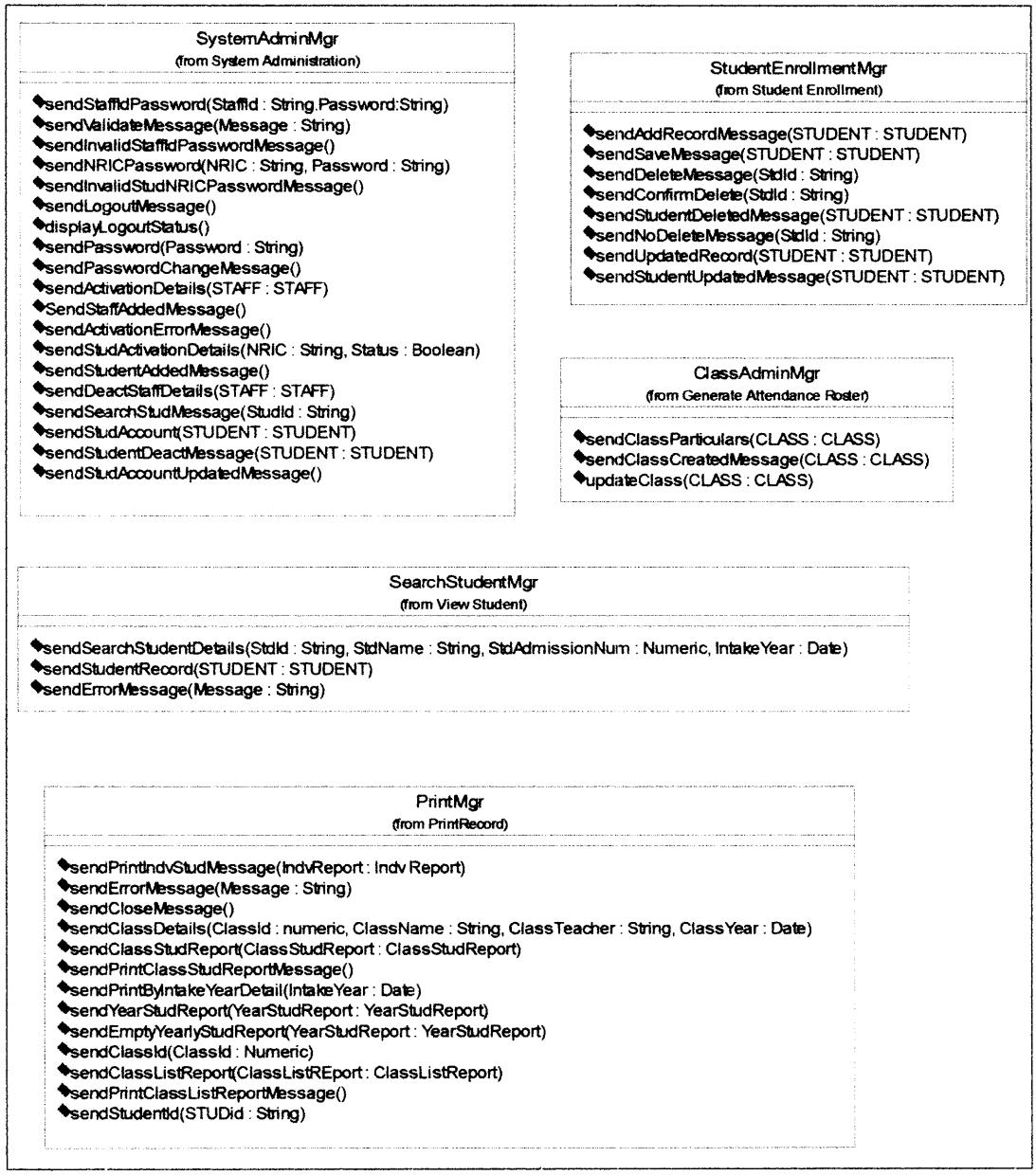
SES Student Enrollment System

APPENDIX A

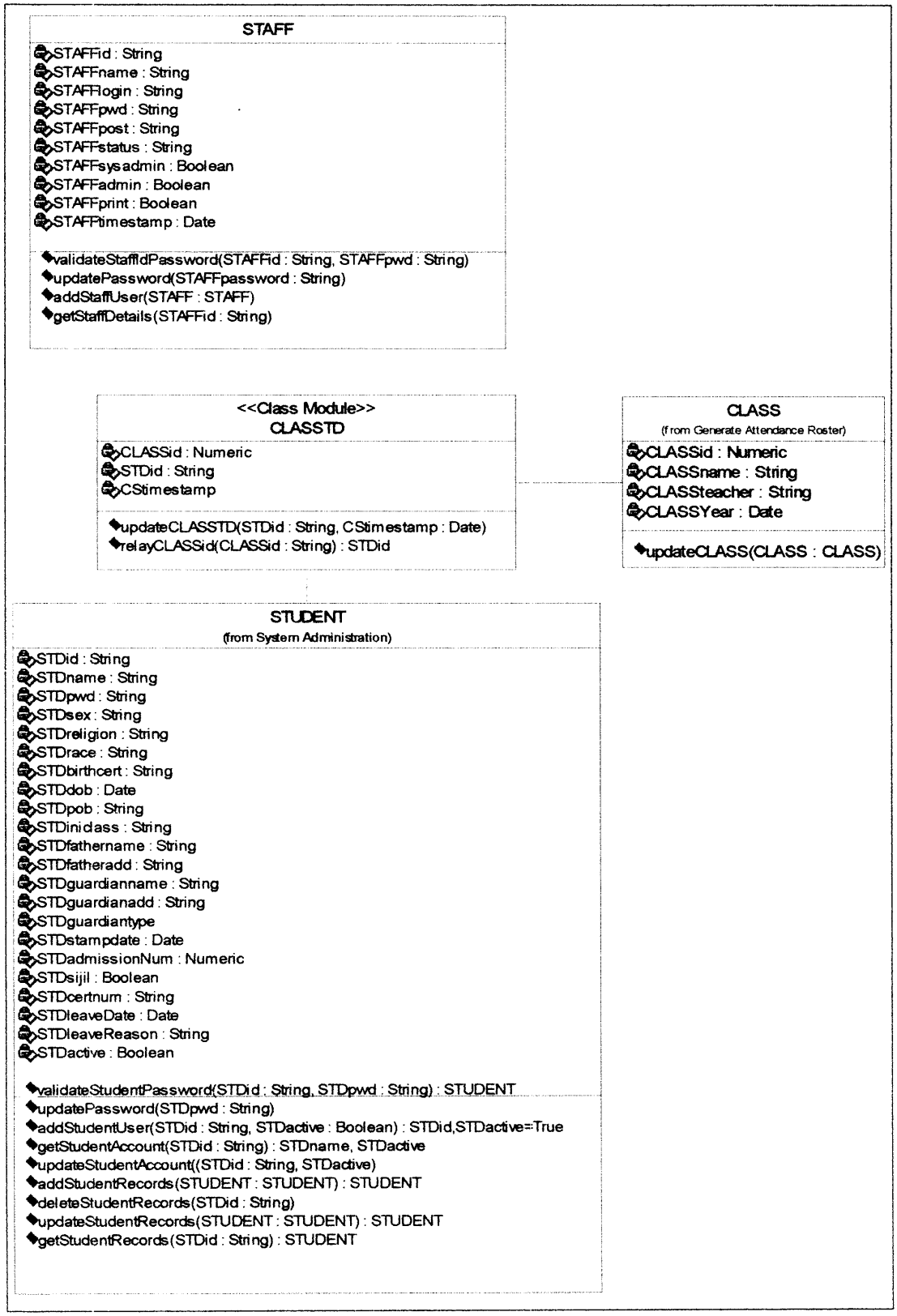
CLASS DIAGRAM



Presentation Layer Package



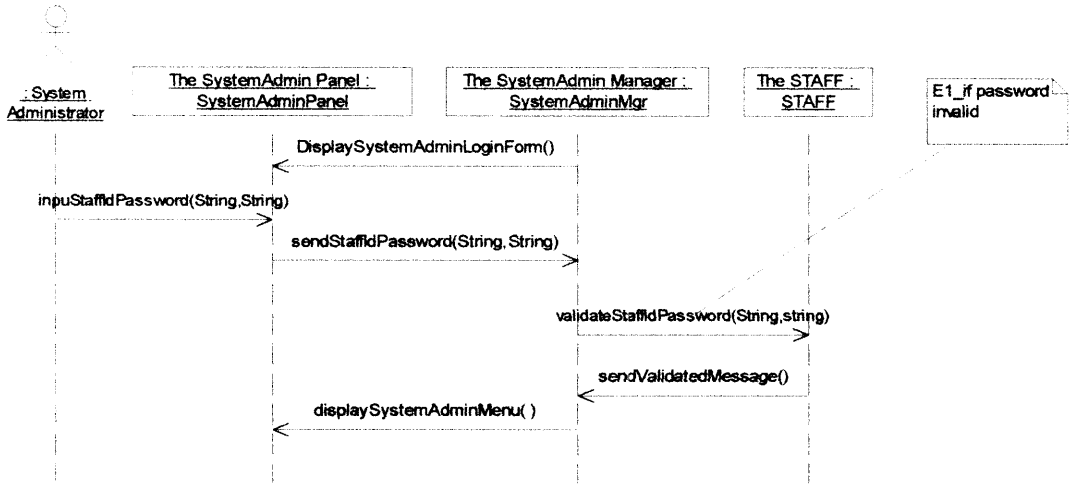
Application Logic Layer Package



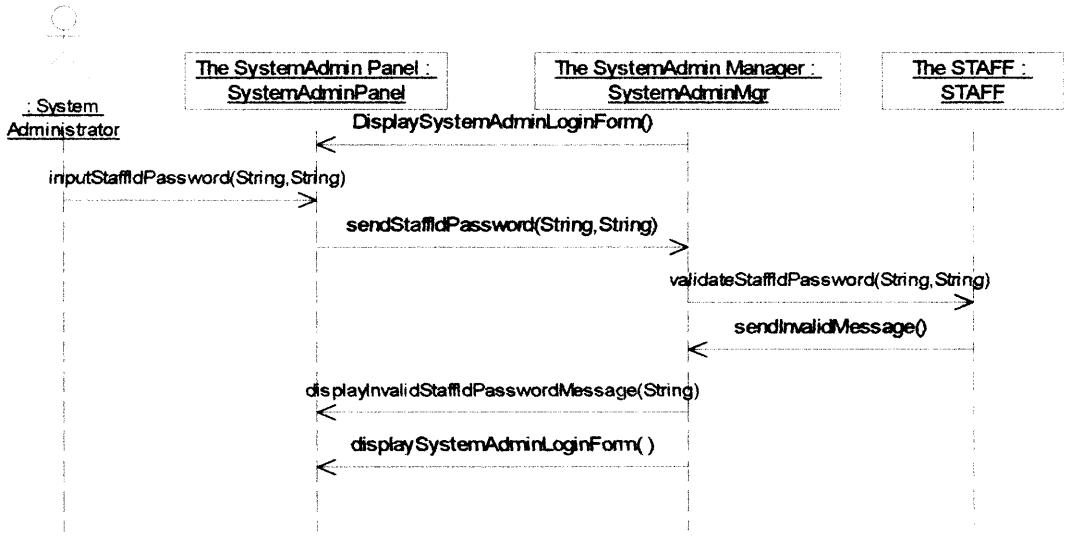
Storage Layer package

Appendix B

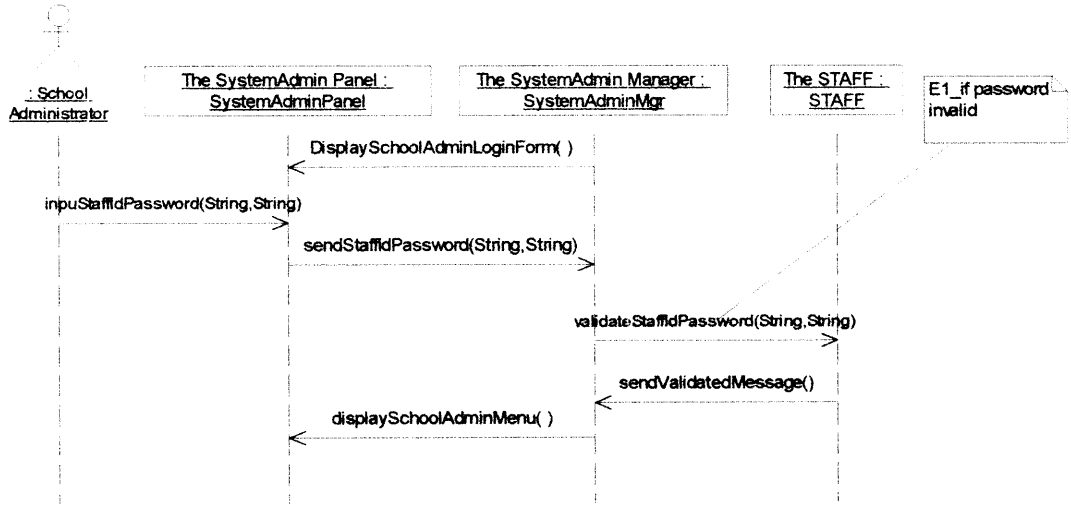
Sequence Diagrams



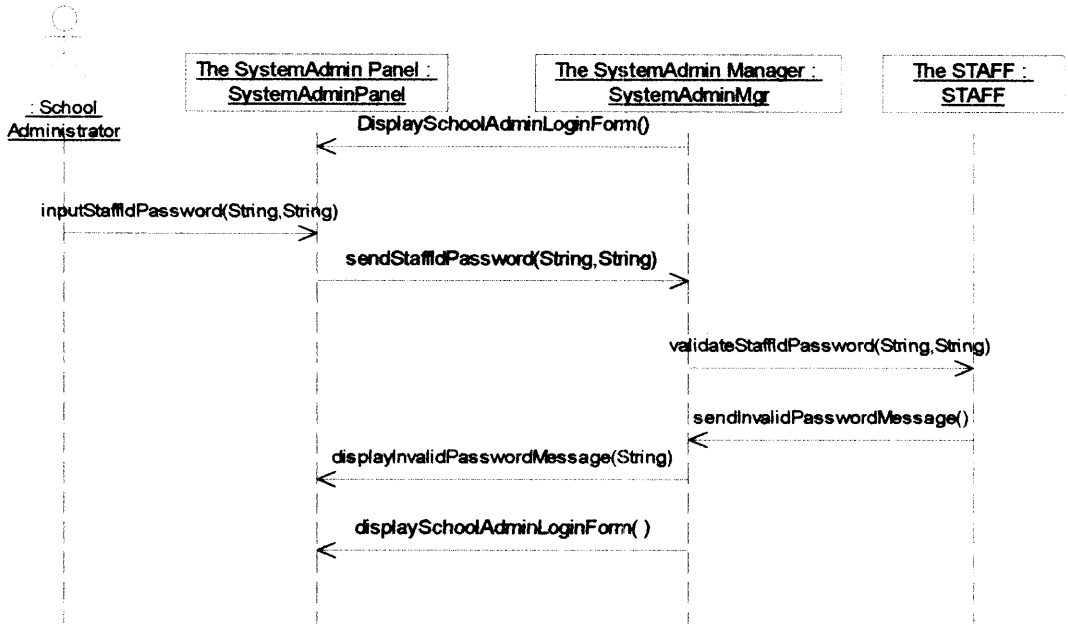
System Administrator Login



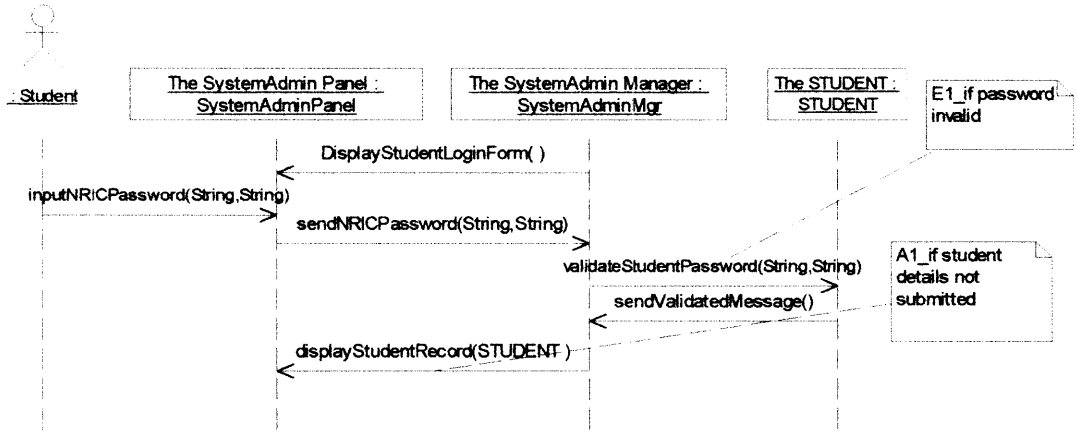
System Administrator Login/E1_if password invalid.



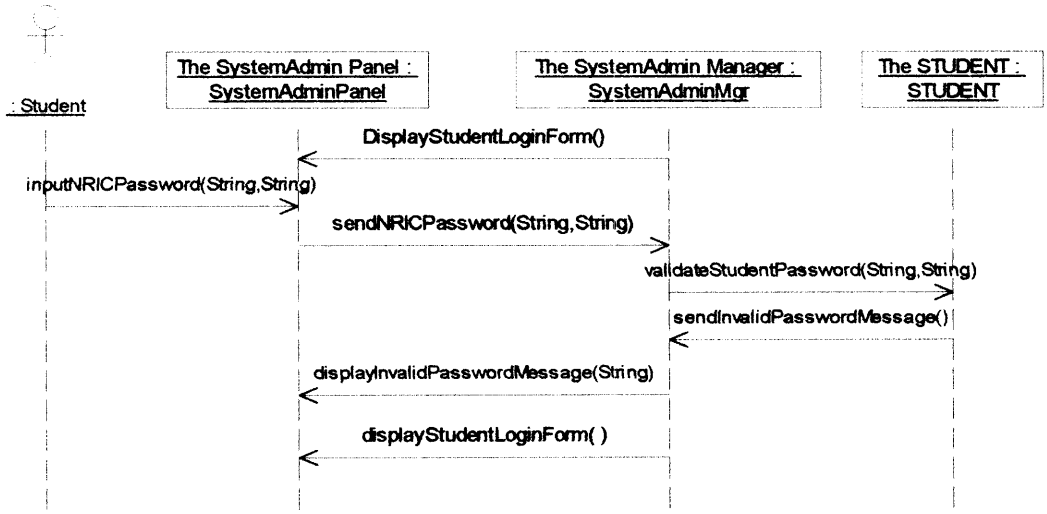
School Administrator Login



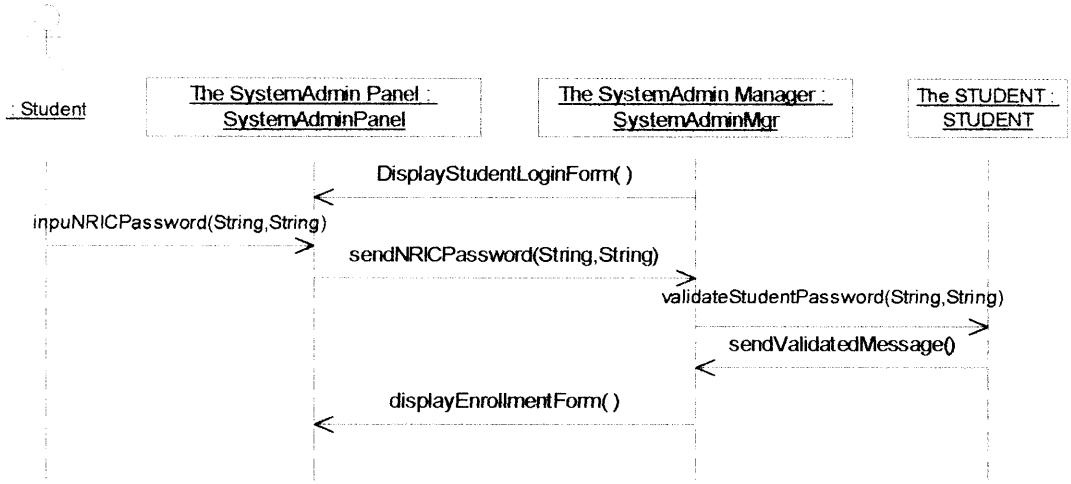
School Administrator Login/E1_if password invalid



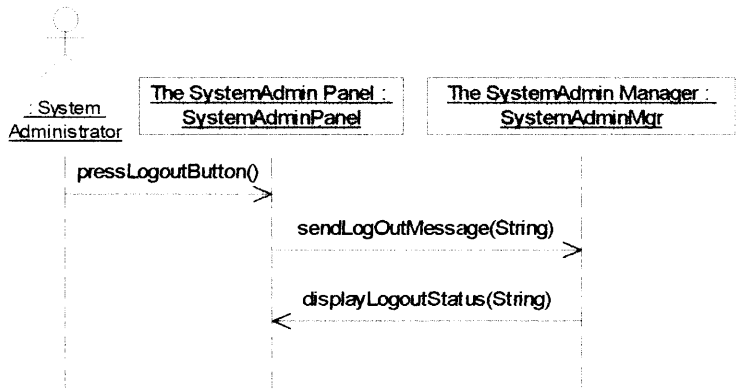
Student Login



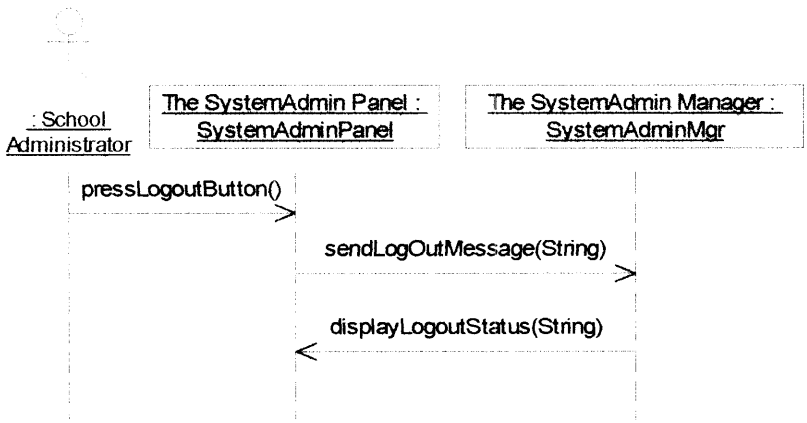
Student Login/E1_if password invalid



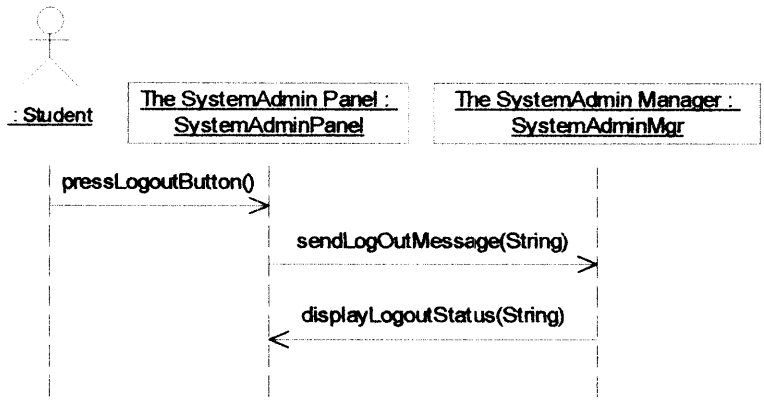
Student Login/A1_if student particulars not submitted



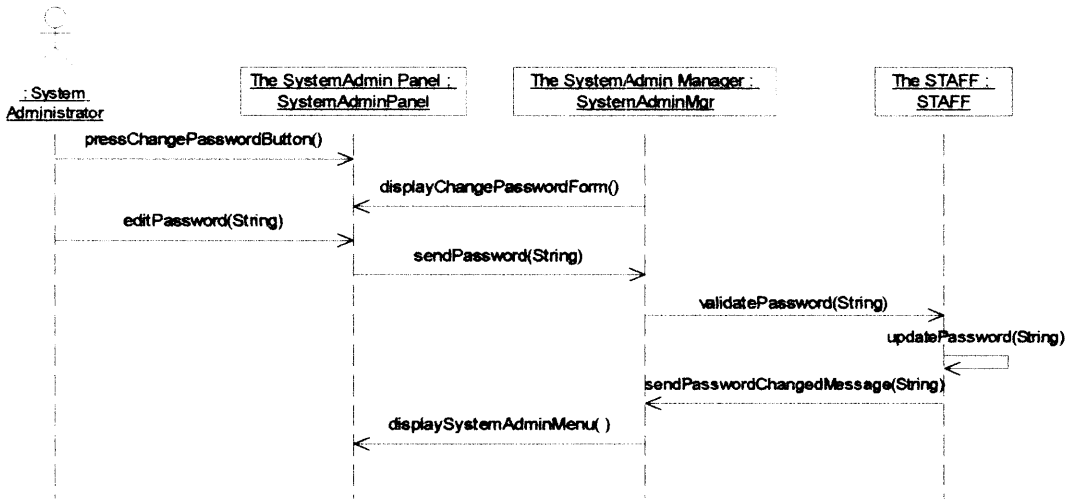
System Administrator Logout



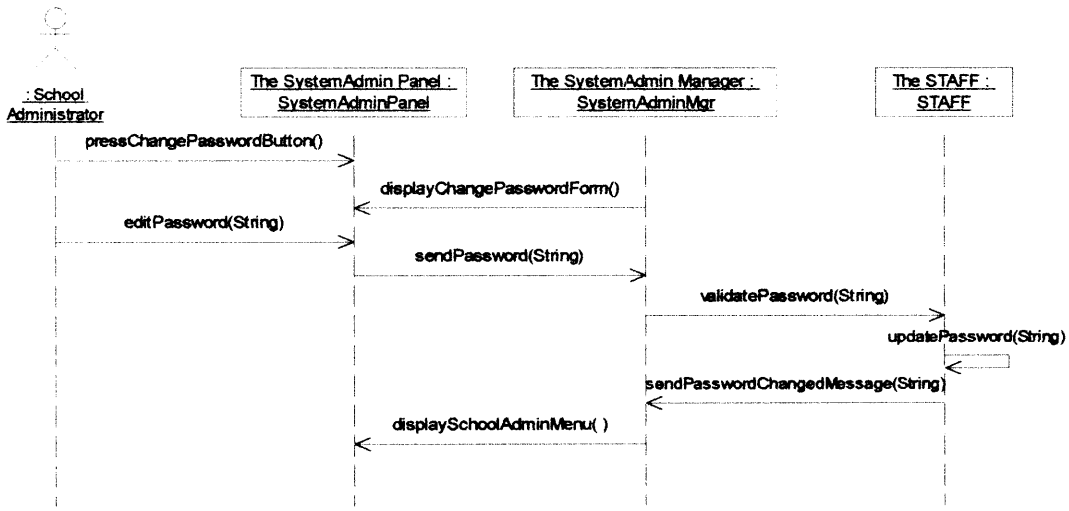
School Administrator Logout



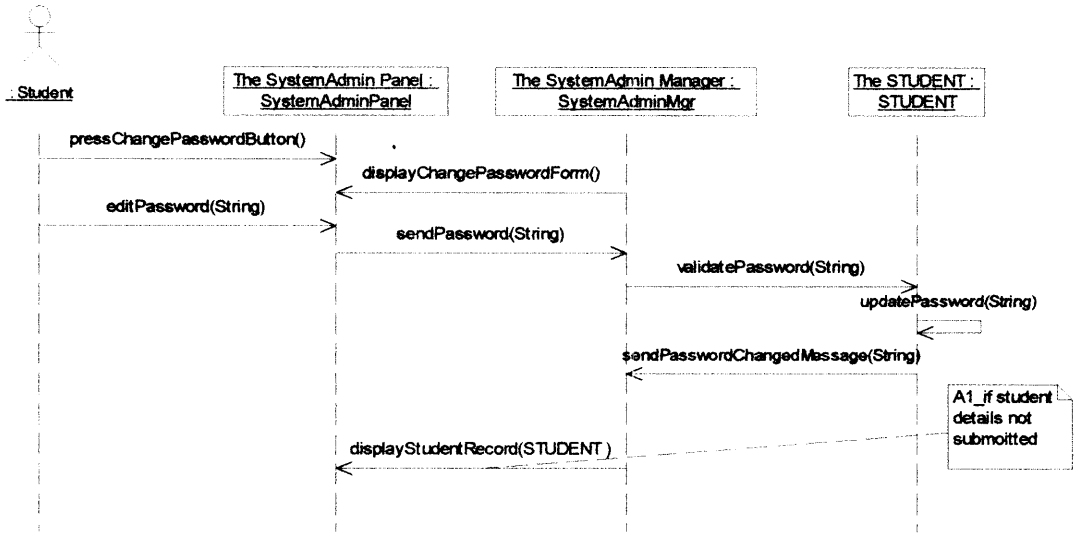
Student Logout



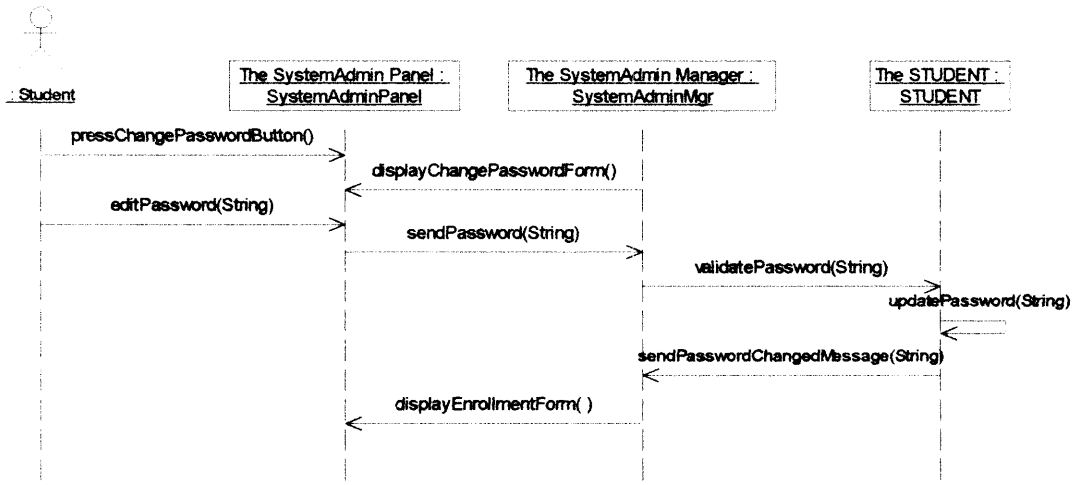
System Administrator Change Password



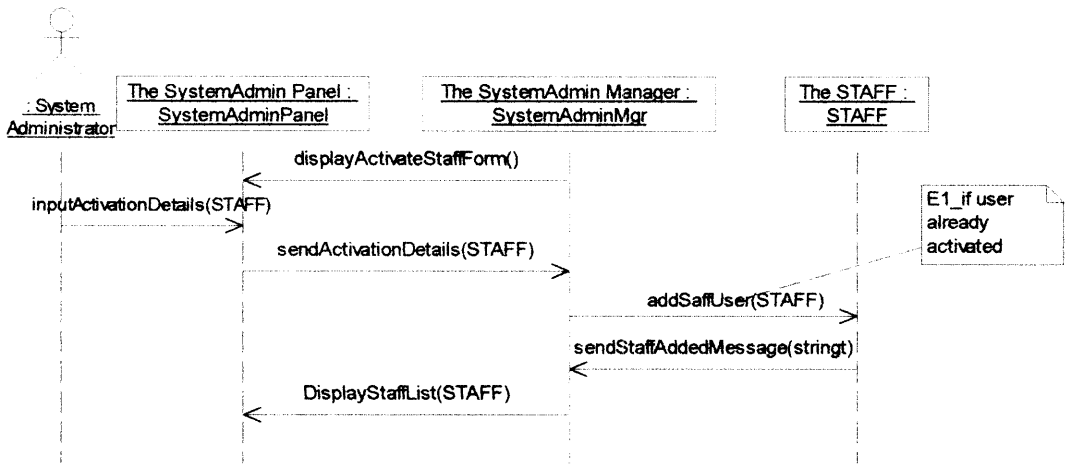
School Administrator Change Password



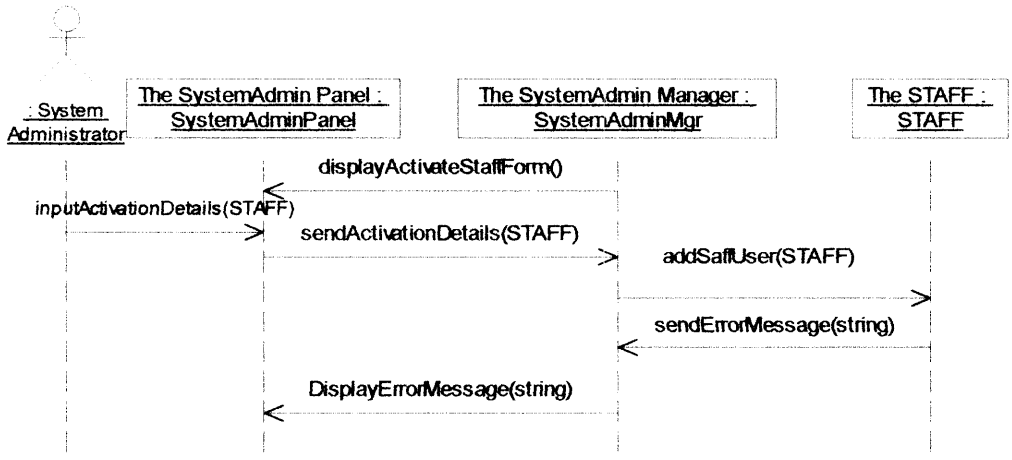
Student Change Password



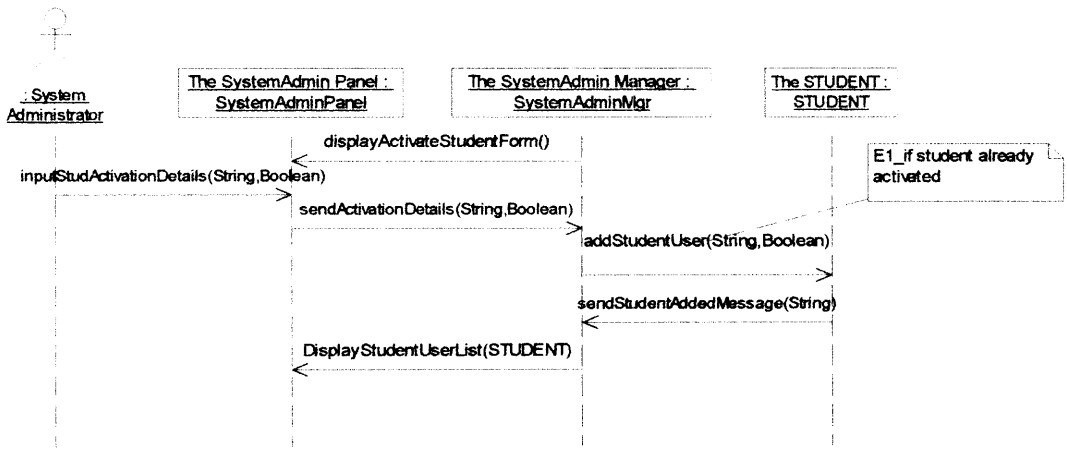
Student Change Password/A1_if student particulars not submitted



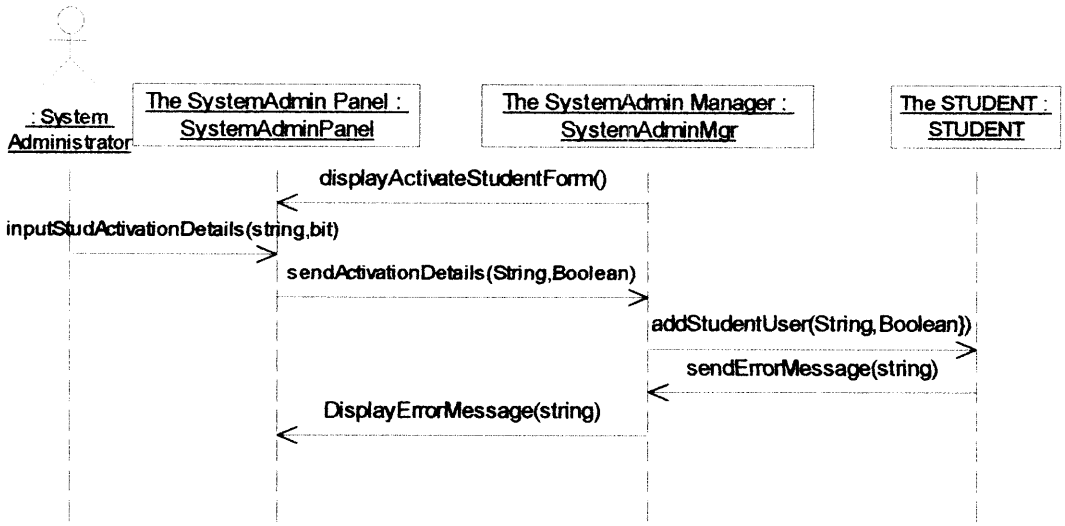
Activate Staff



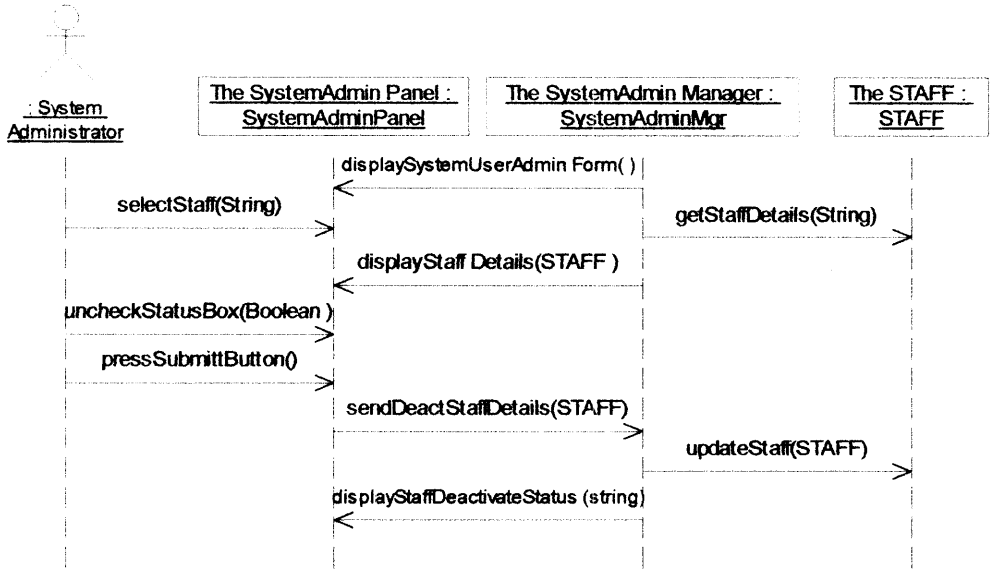
Activate Staff/E1_if user already activate



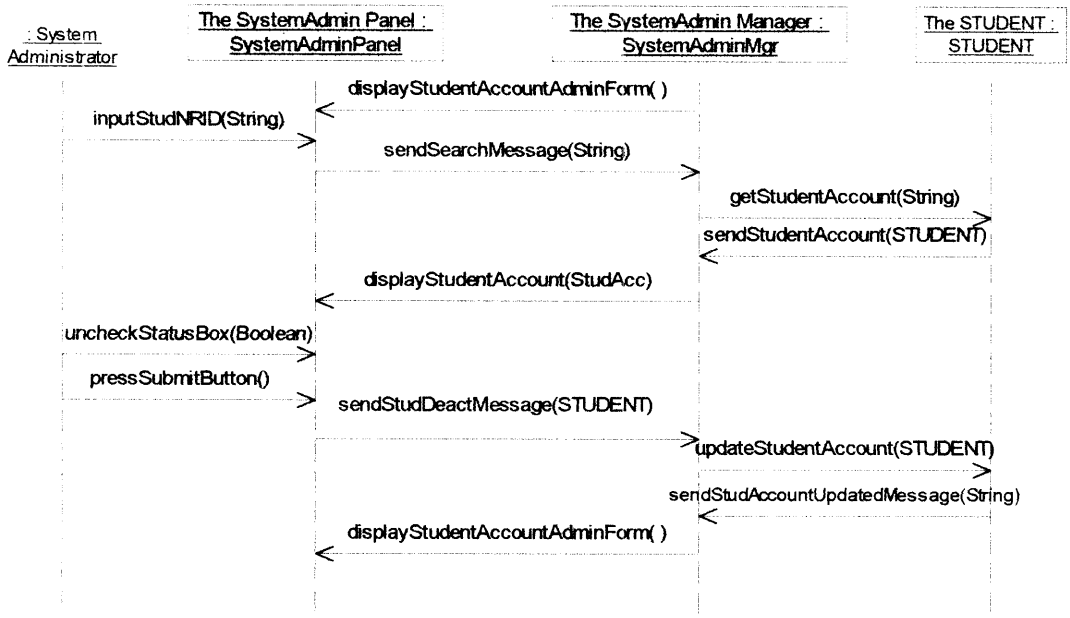
Activate Student



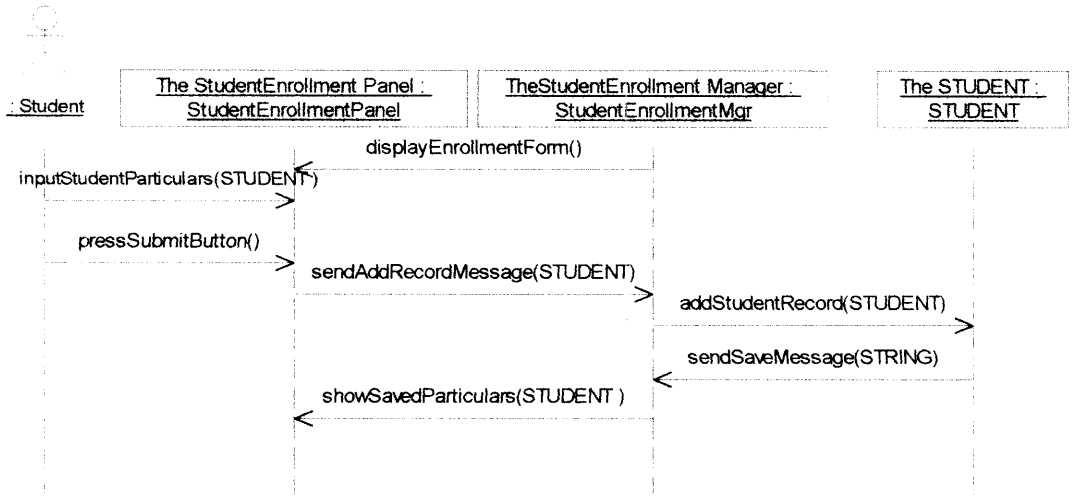
Activate Student/E1_if student already activated



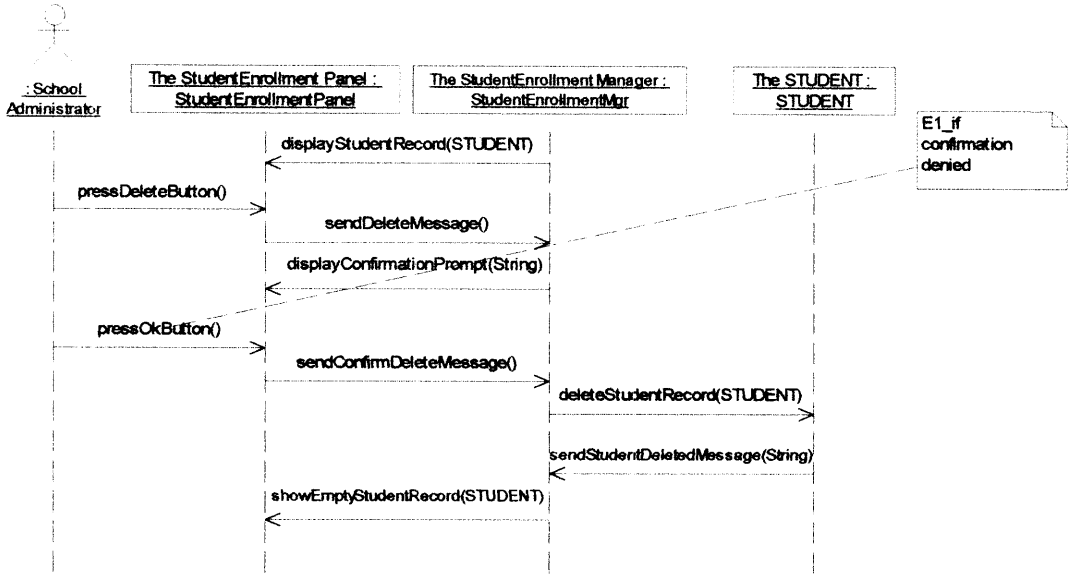
Deactivate Staff



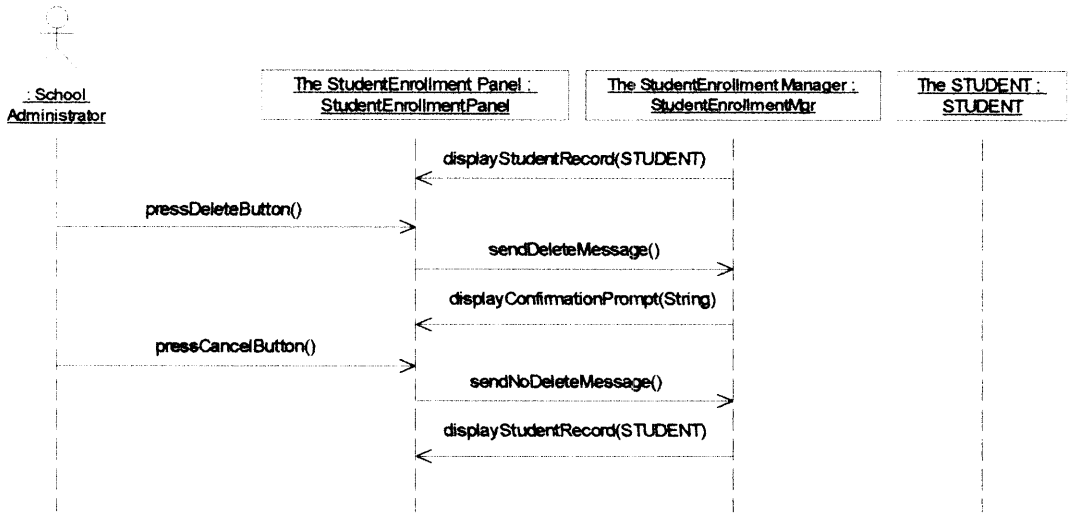
Deactivate Student



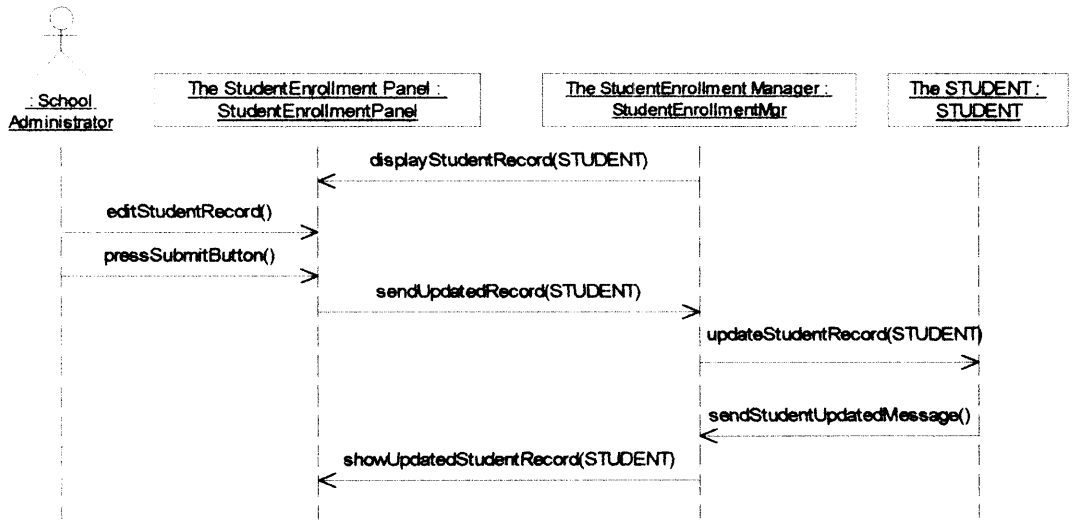
Add Student Particulars



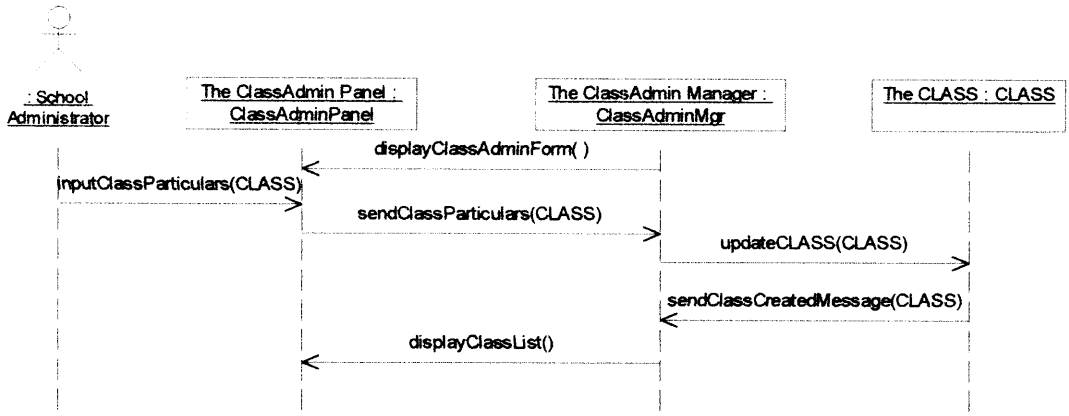
Delete Student Record



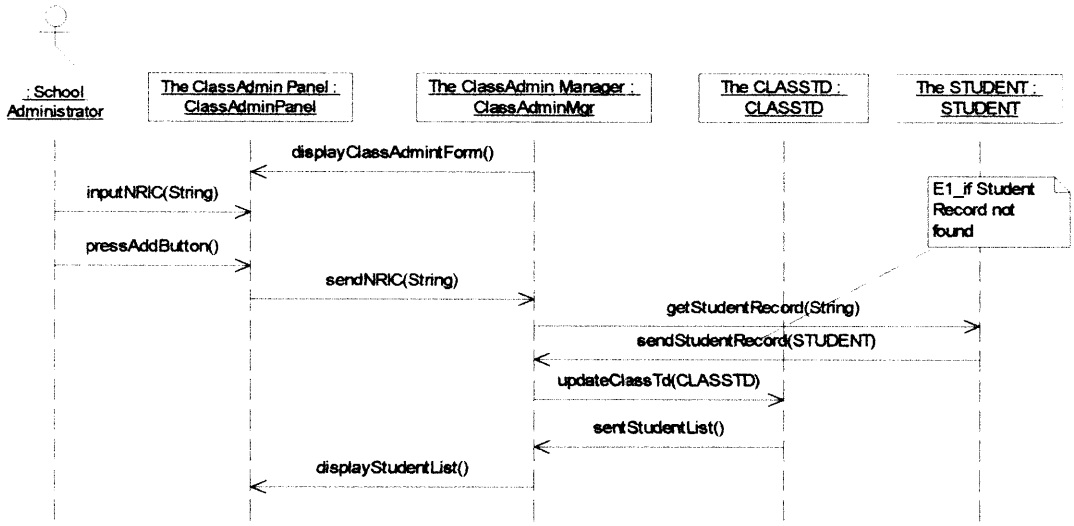
Delete Student Record/E1_if confirmation denied



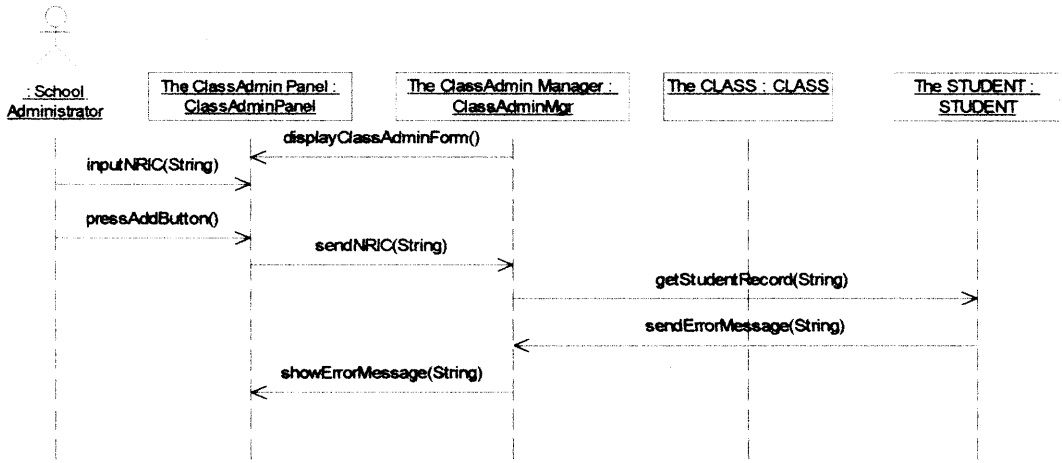
Update Student Record



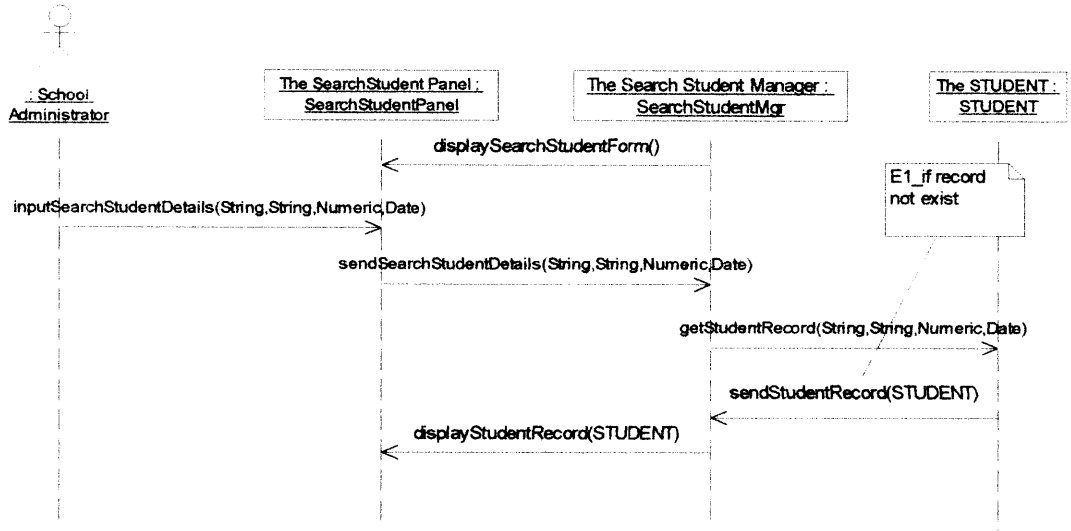
Generate Class



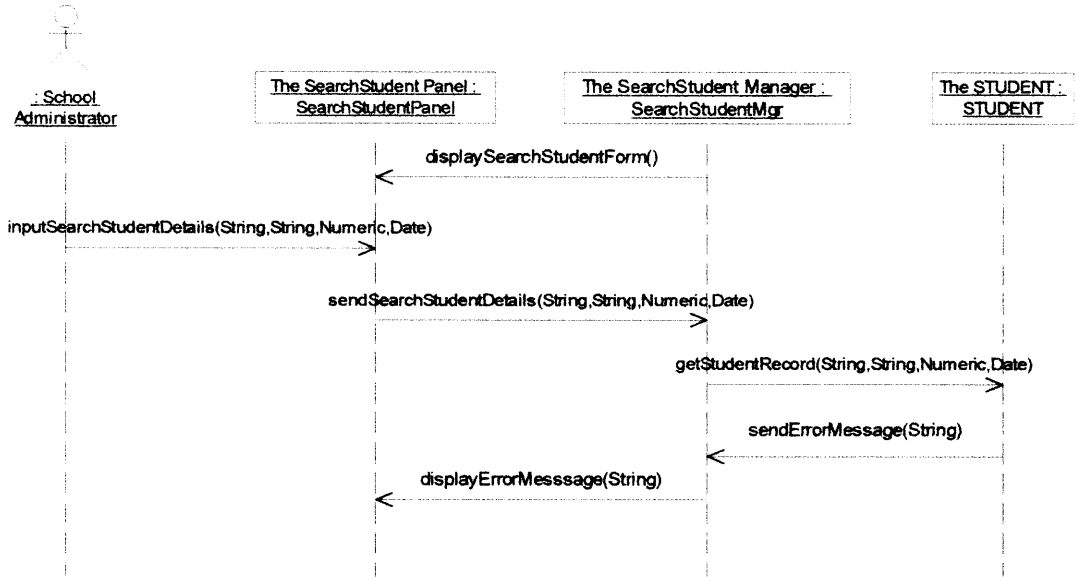
Add Class Member



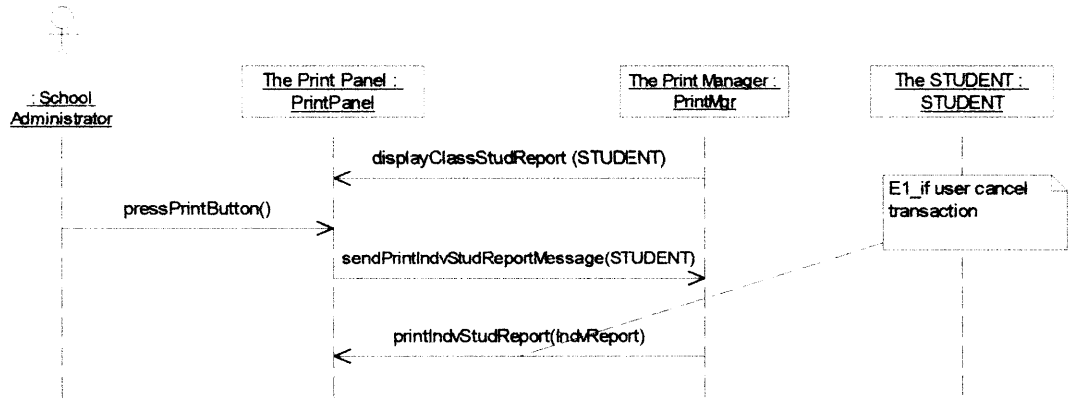
Add Class Member/E1_if student record not found



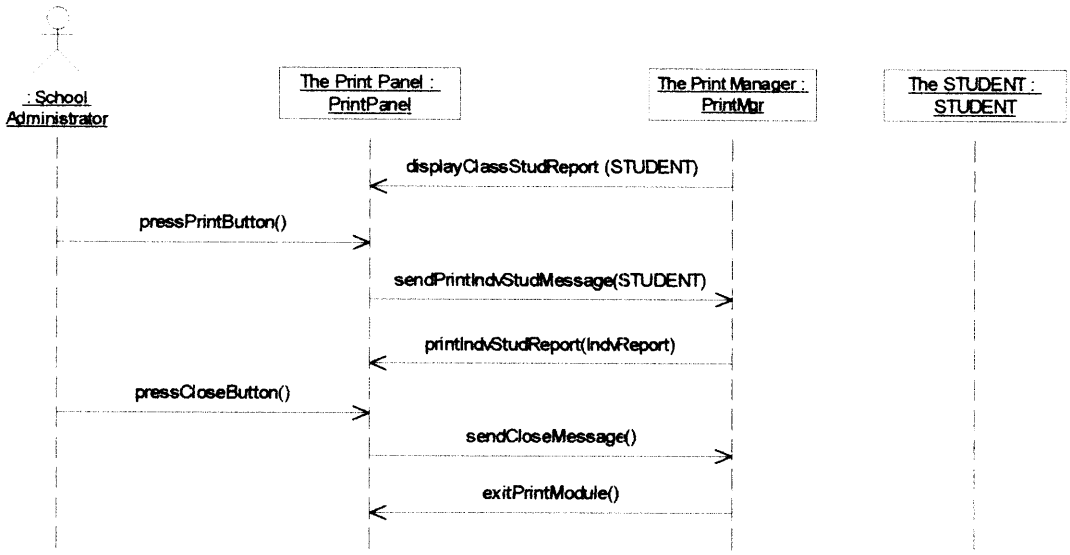
Search Student Record



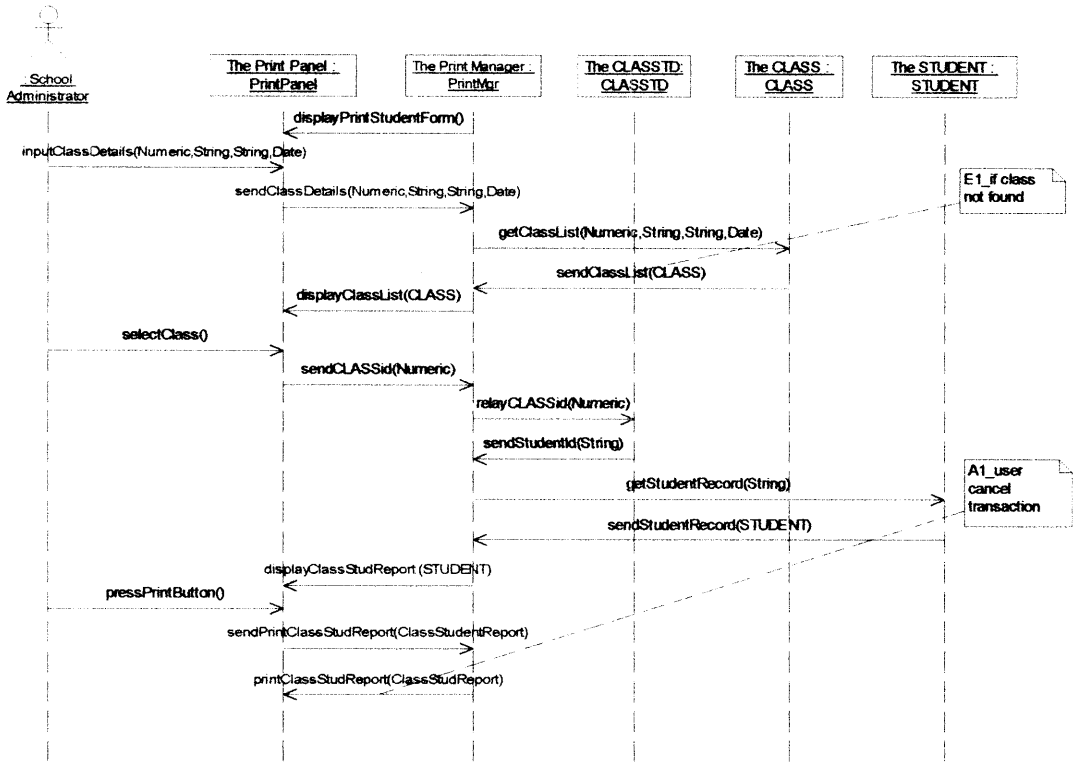
Search Student Record/E1_if student record not found.



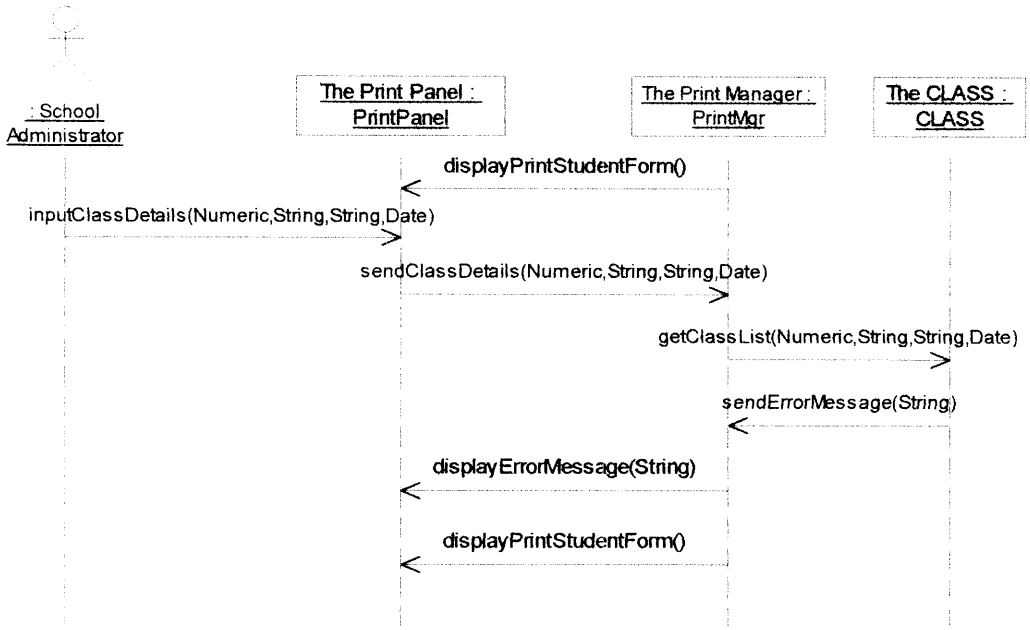
Print Individual Student Record



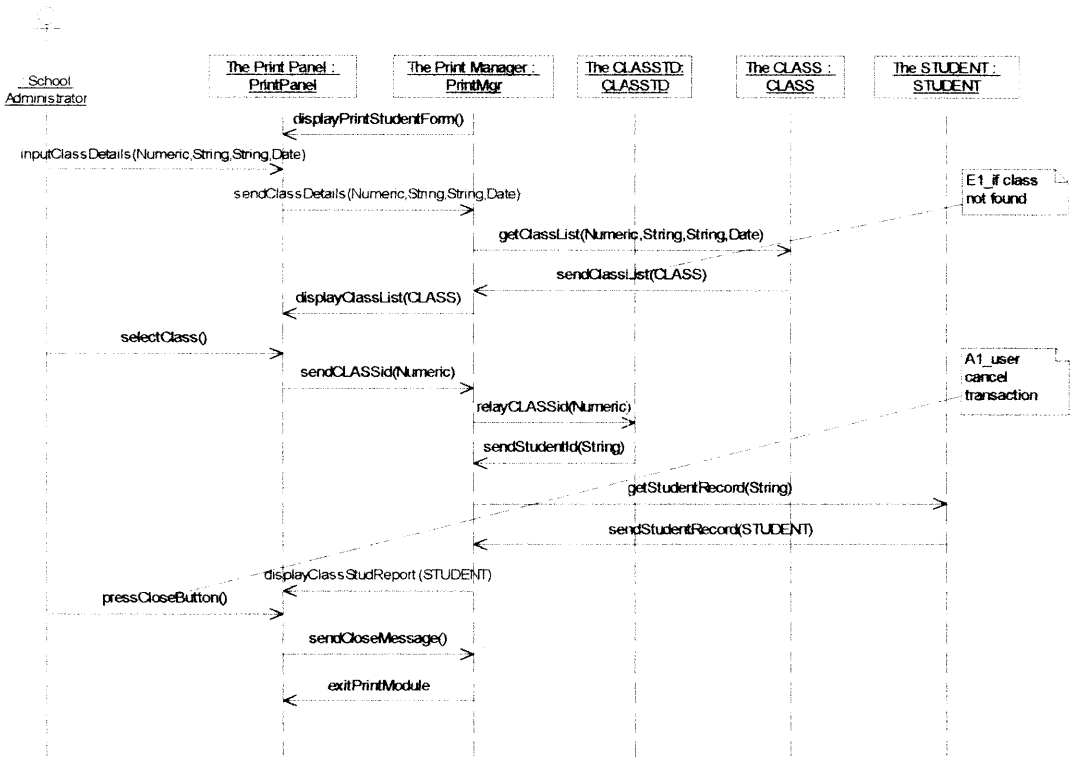
Print Individual Student Record/E1_if user cancels transacrtion



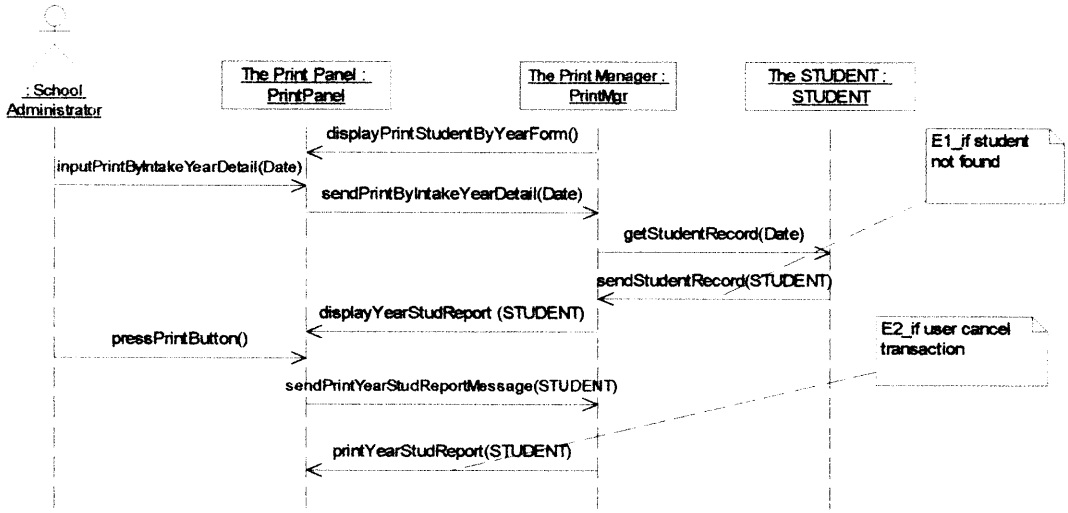
Print Student Record By Class



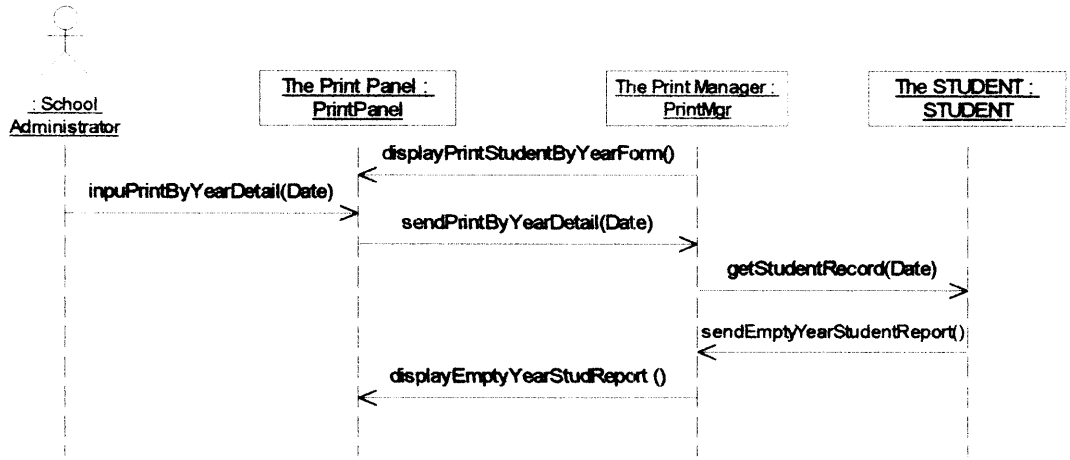
Print Student Record By Class/E1_if class not found



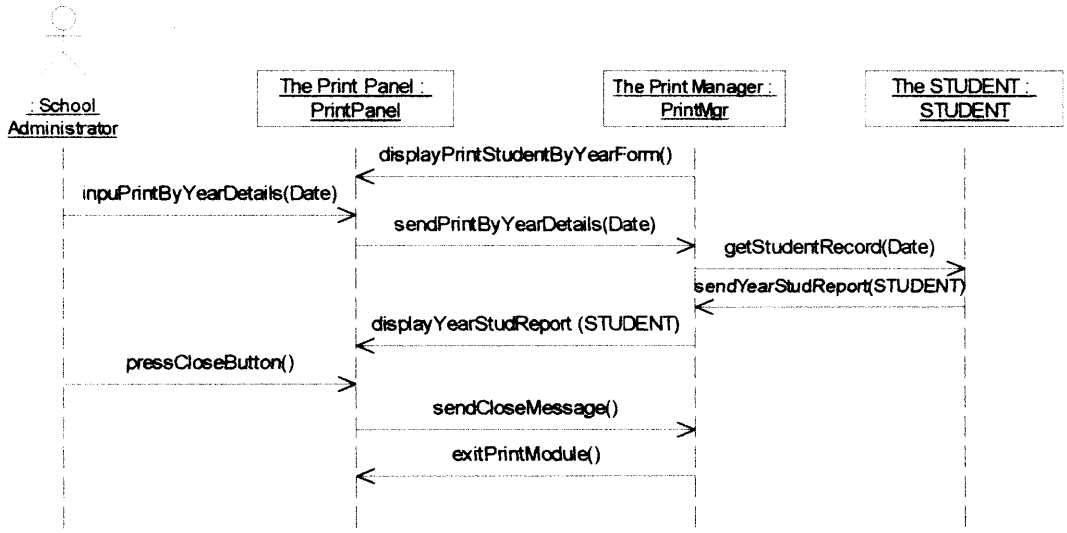
Print Student Record By Class/A1_if user cancel transaction



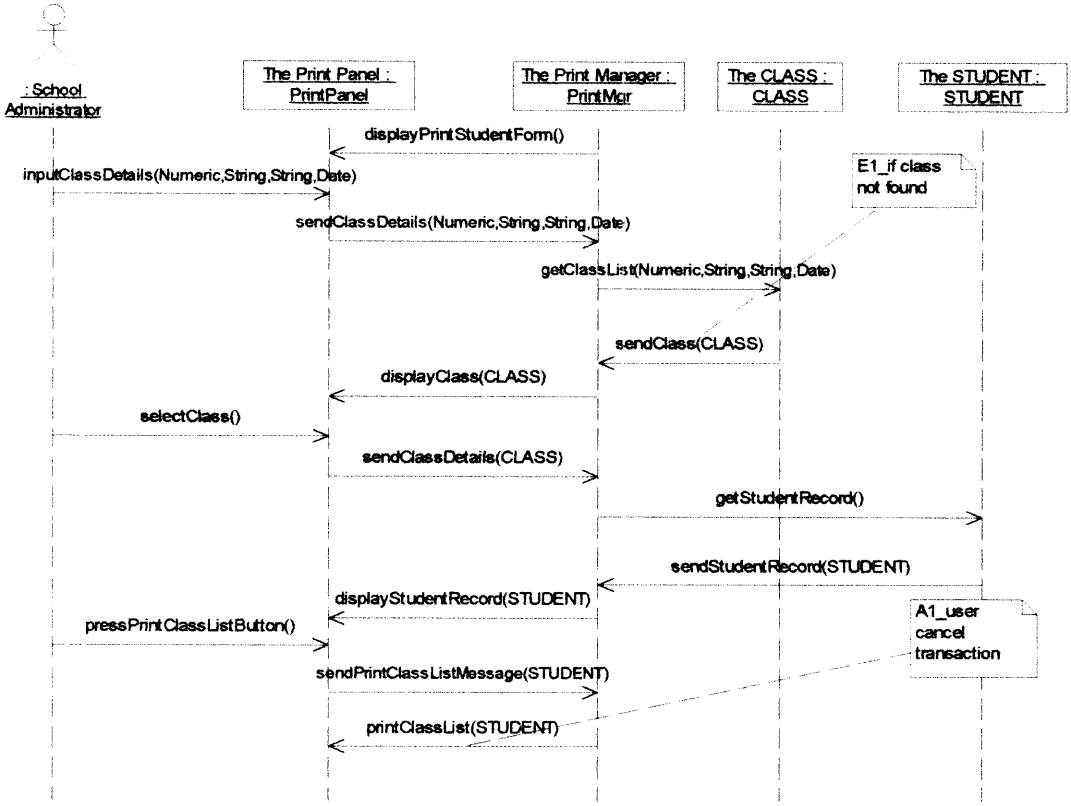
Print Student Record By Intake



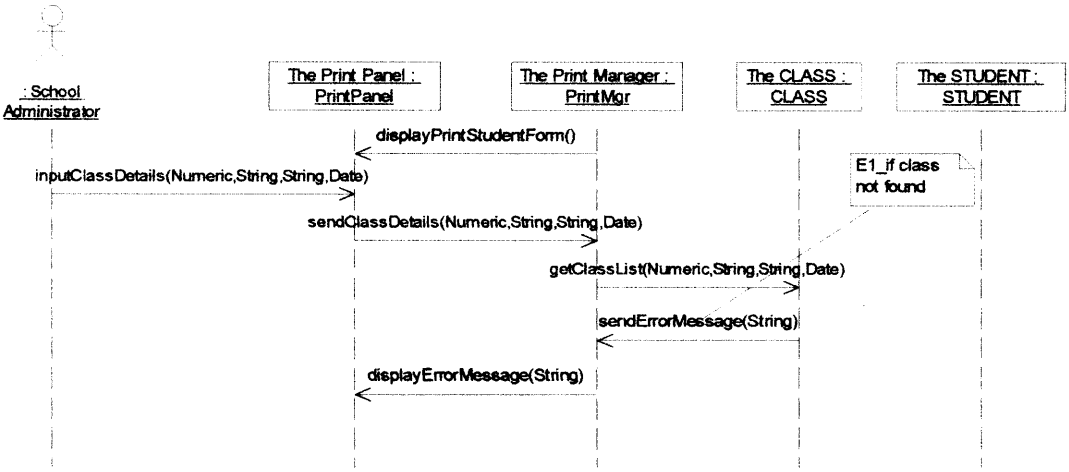
Print Student Record By Intake/E1_if record not found



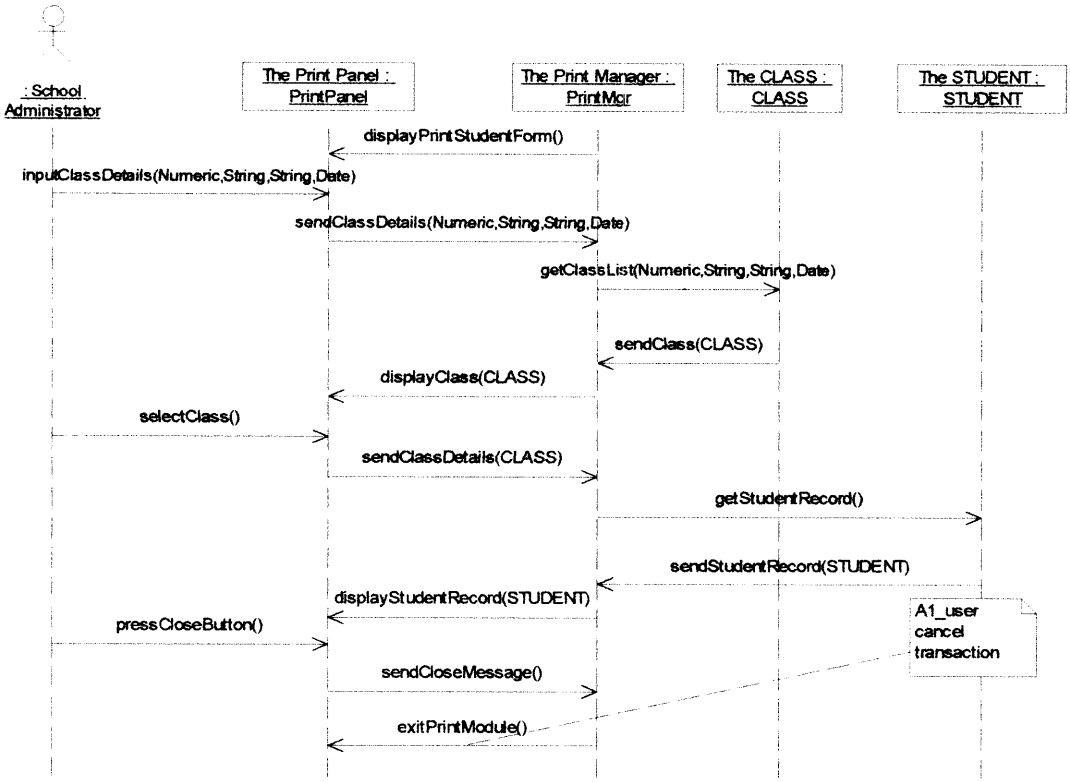
Print Student Record By Intake/A1_if user cancel transaction



Print Class List



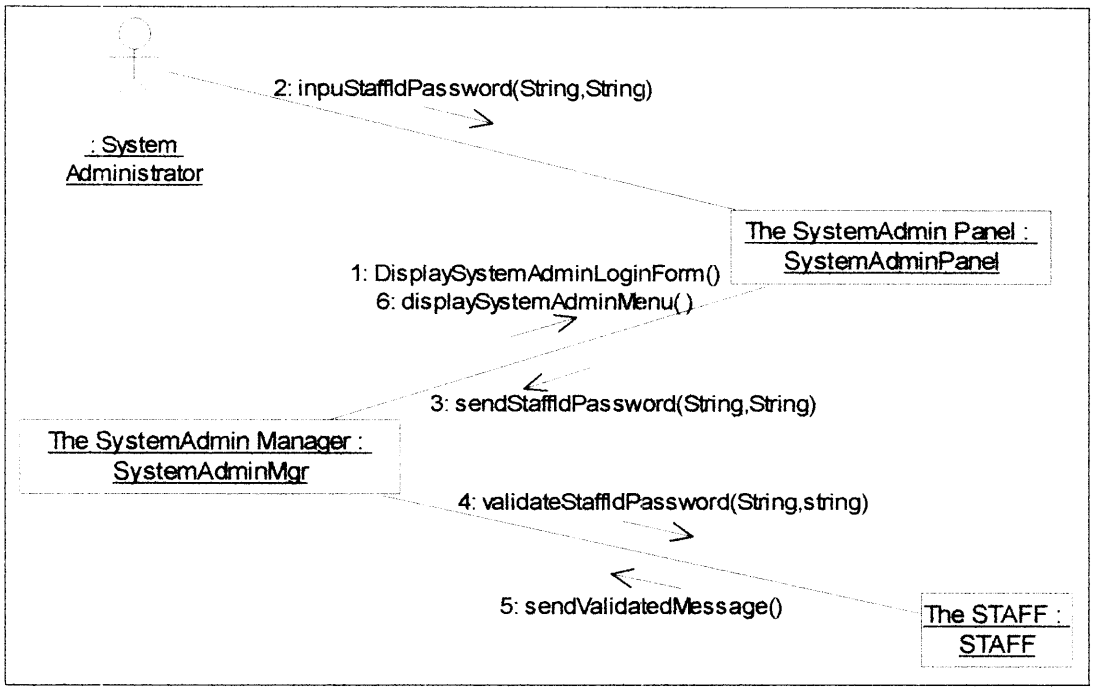
Print Class List/E1_if class not found



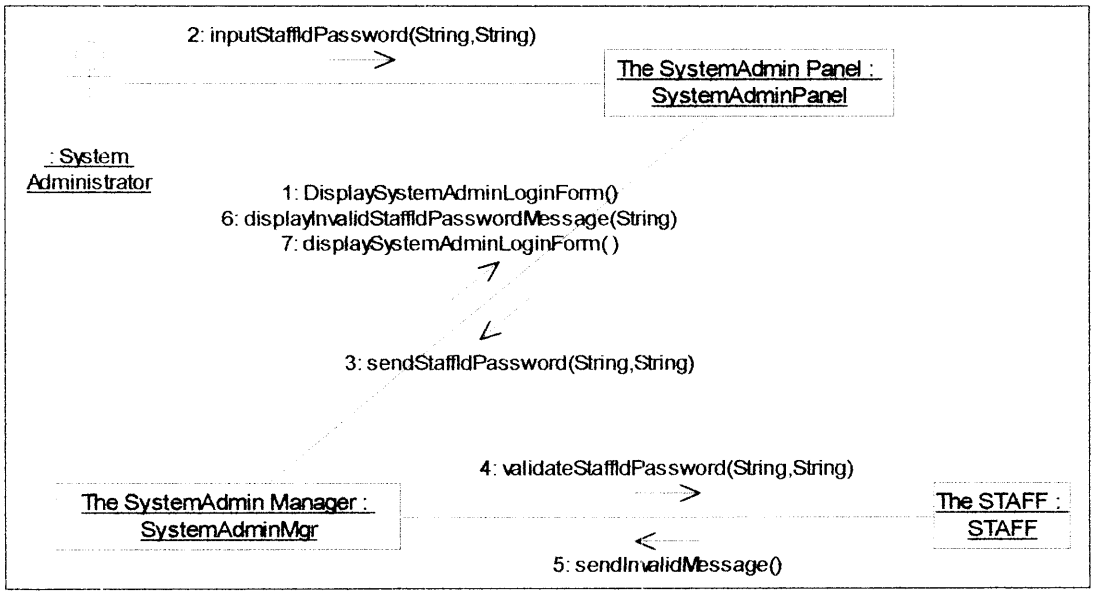
Print Class List/A1_if user cancel transaction

APPENDIX C

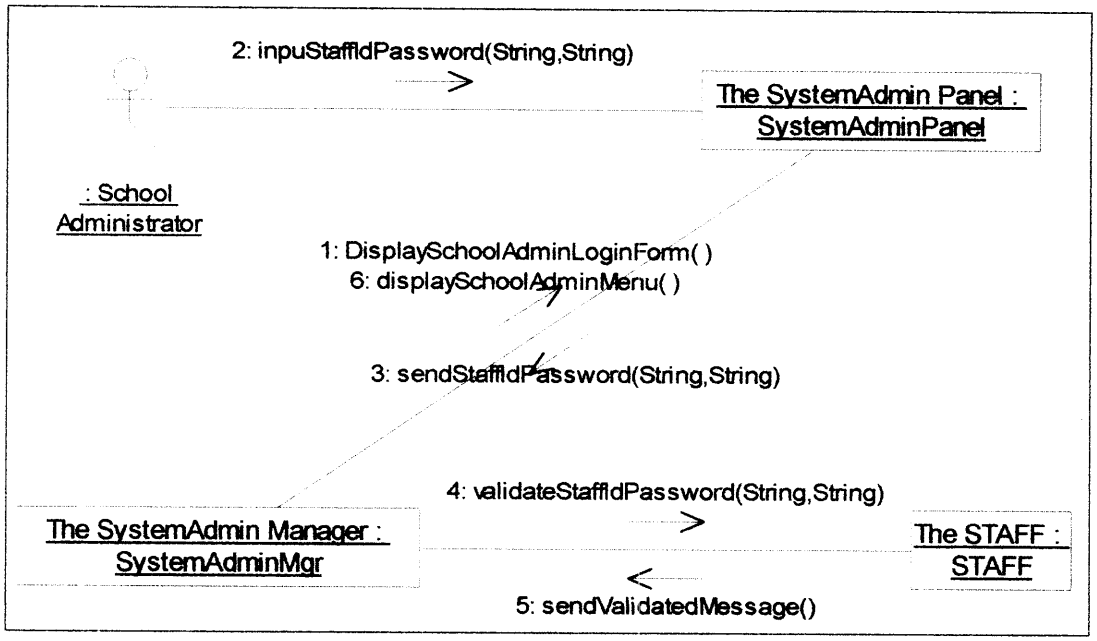
COLLABORATION DIAGRAM



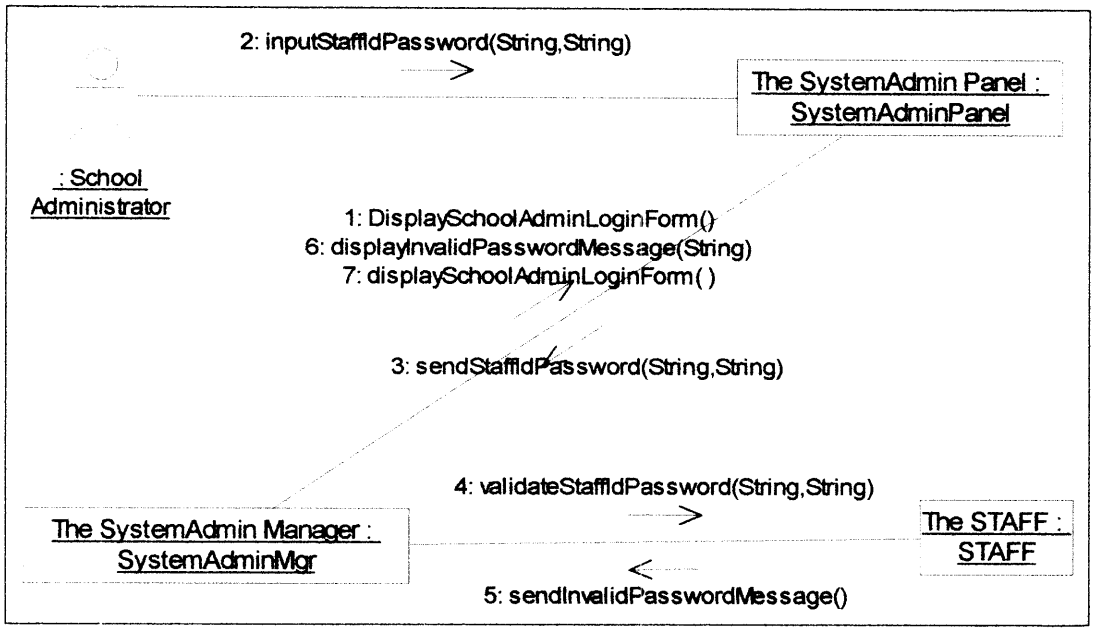
Log In_System Administrator Log In



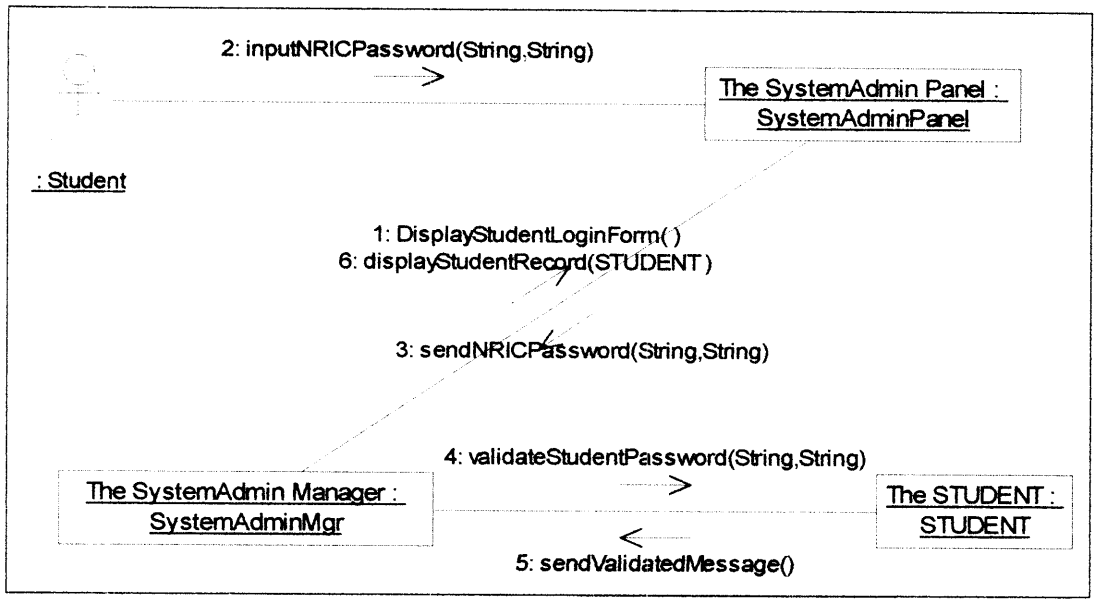
Login_System Administrator Log In/E1_if invalid password



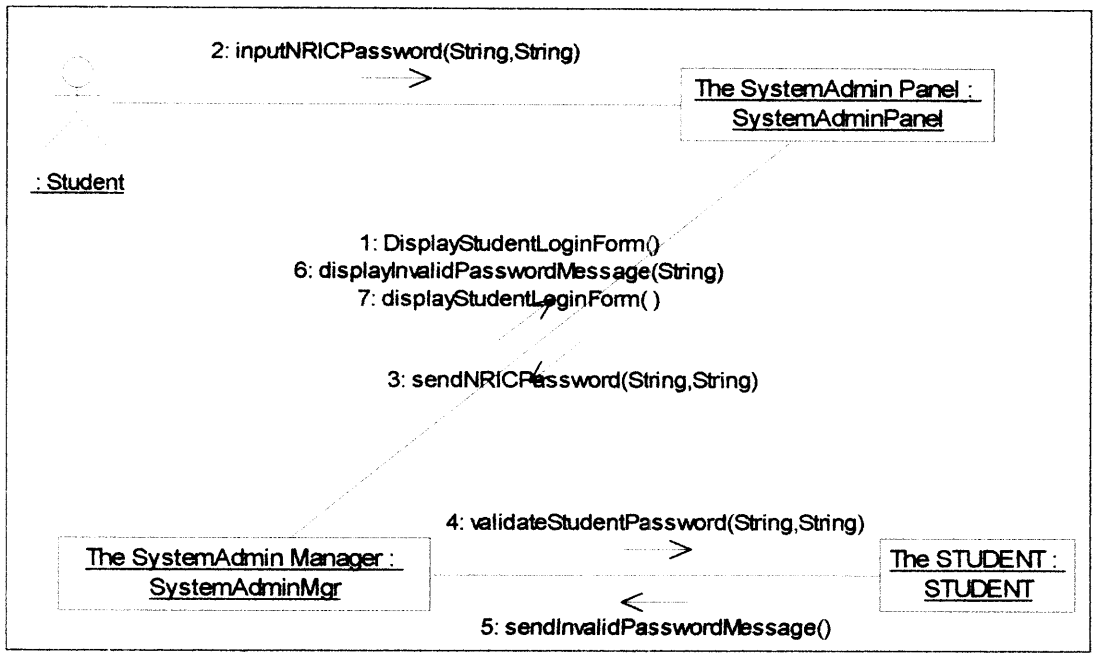
Login_School Administrator Log In



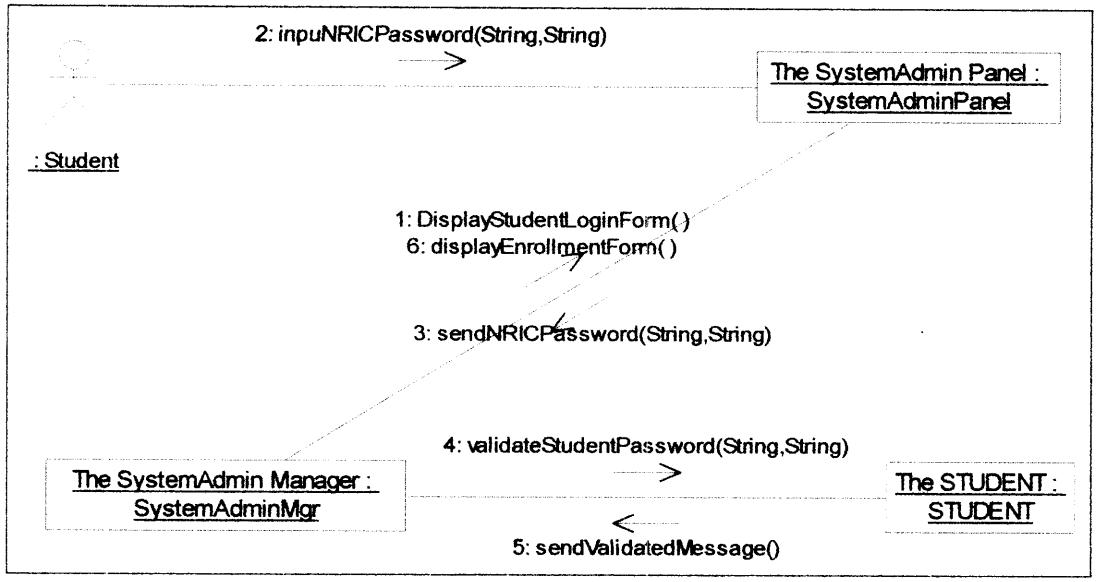
Login_School Administrator Log In/E1_if invalid password



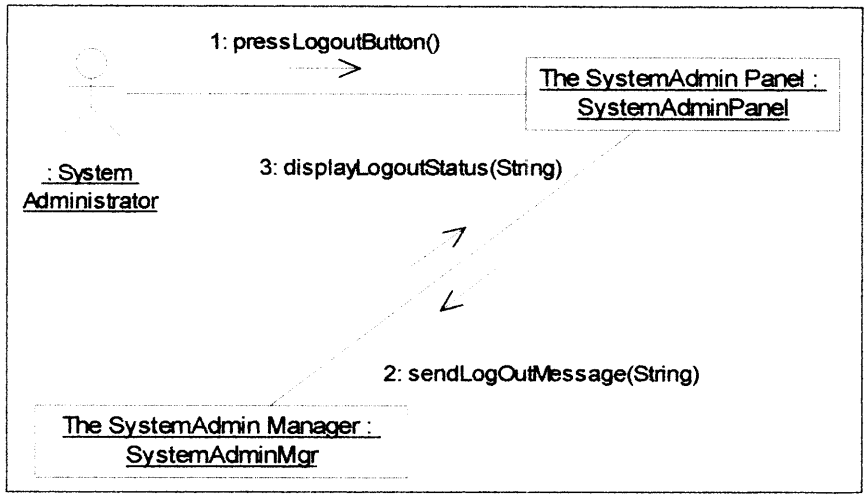
Log In_Student Log In



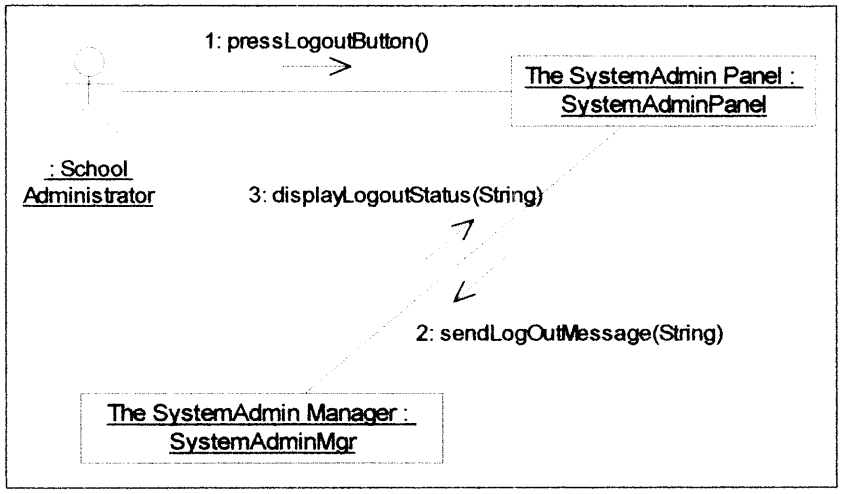
Log In_Student Log In/E1_if invalid password



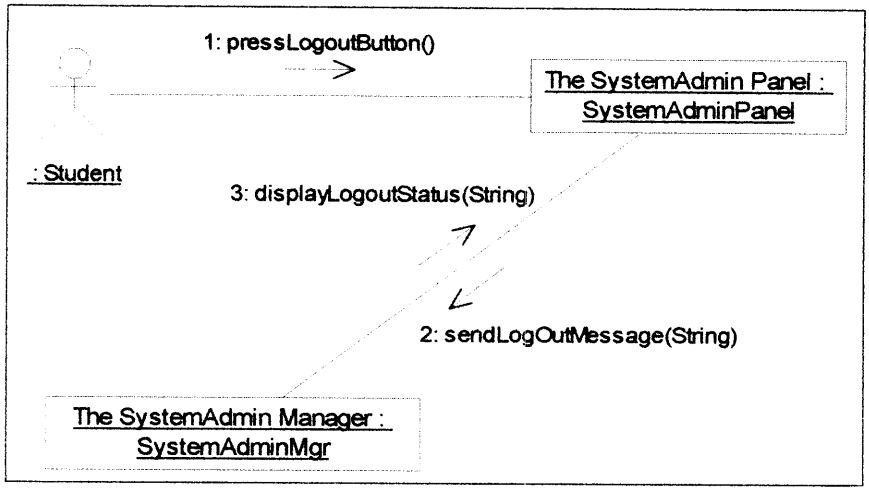
Log In_Student Log In/A1_if student details not submitted



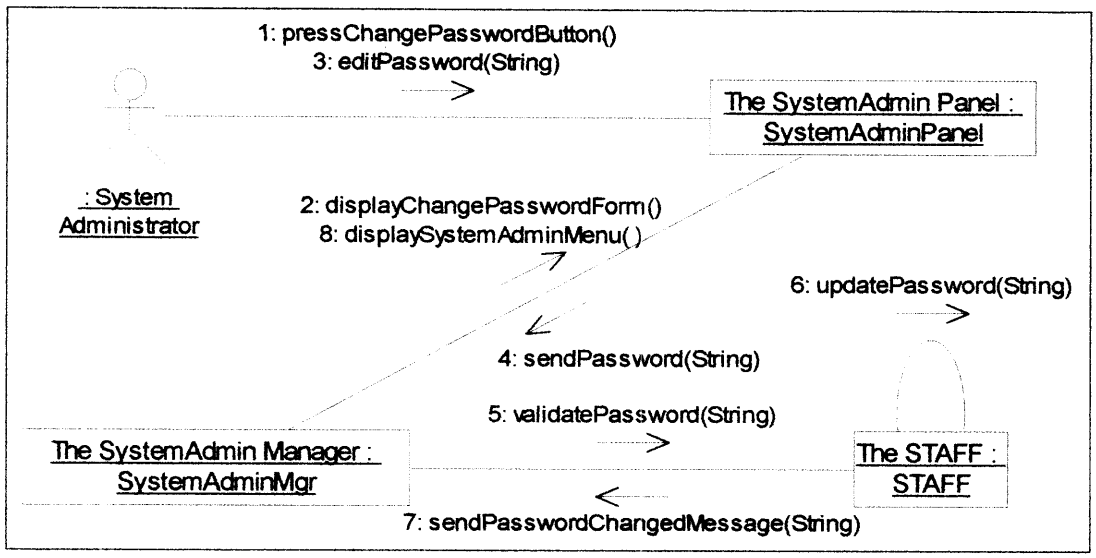
Log Out_System Administrator Logout



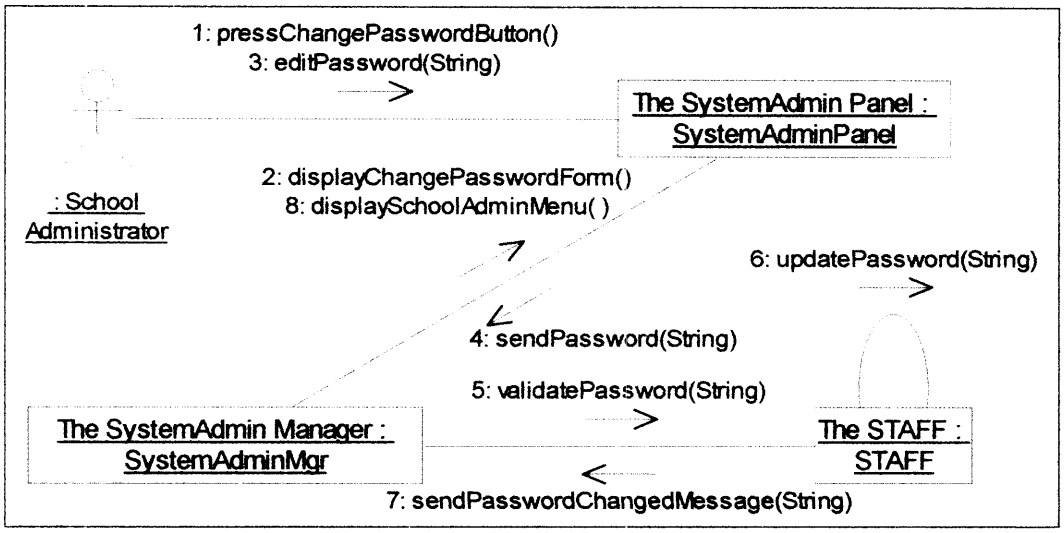
Log Out_School Administrator Logout



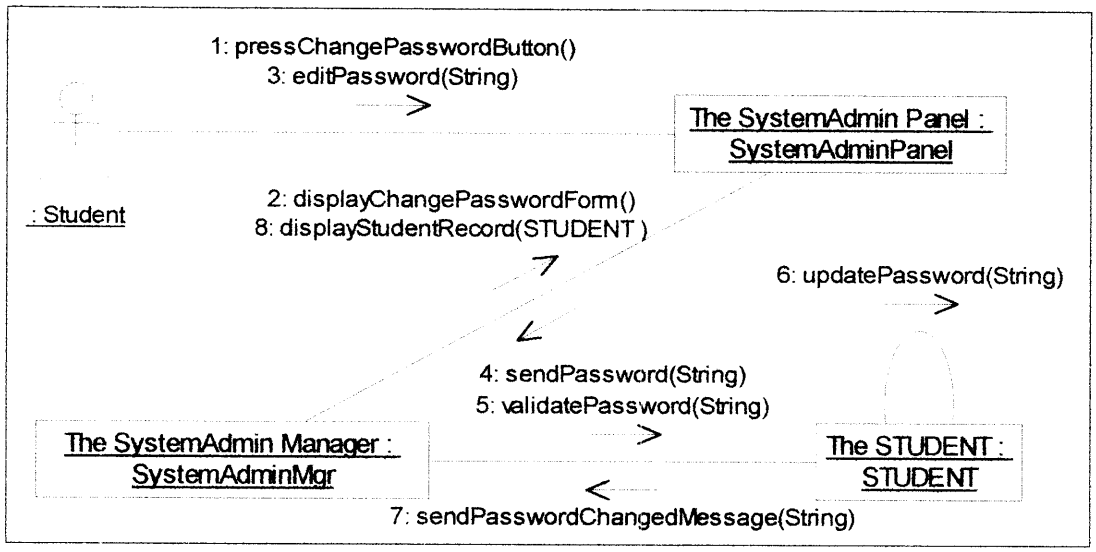
Log Out_Student Logout



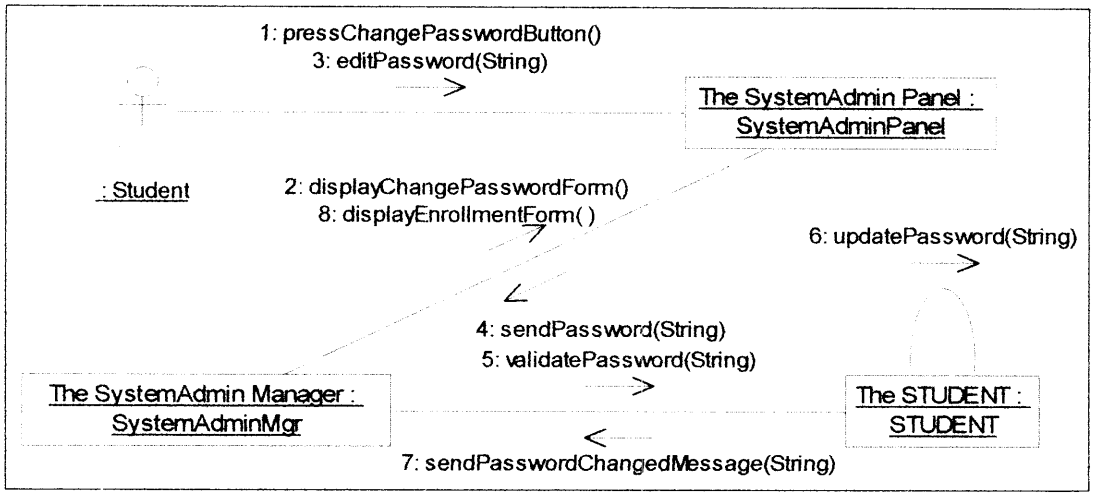
Change Password_System Administrator Change Password



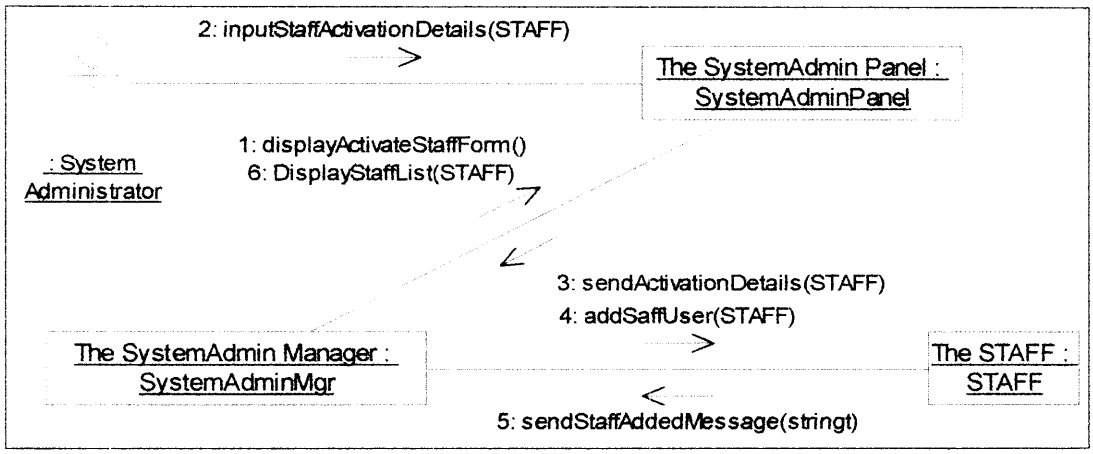
Change Password_School Administrator Change Password



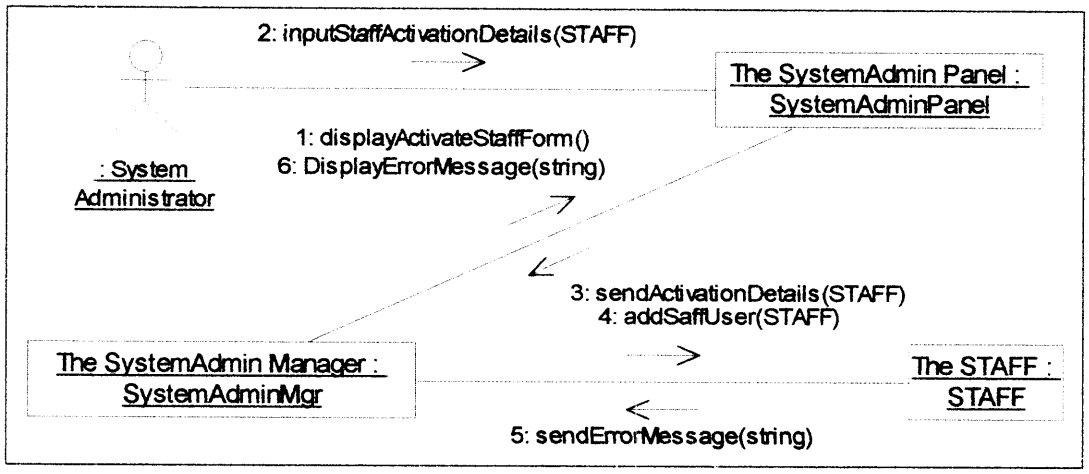
Change Password_Student Change Password



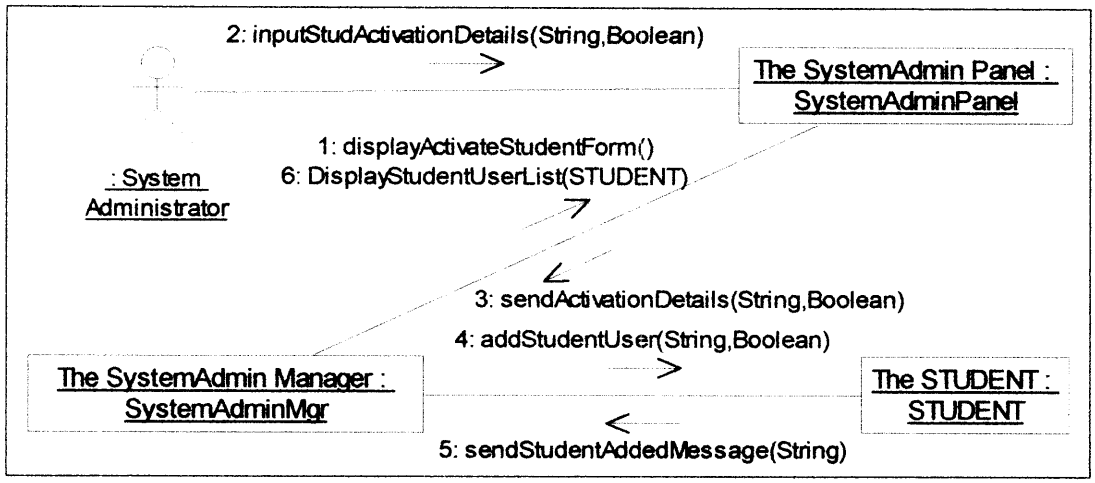
Change Password_Student Change Password/A1_if student details not submitted



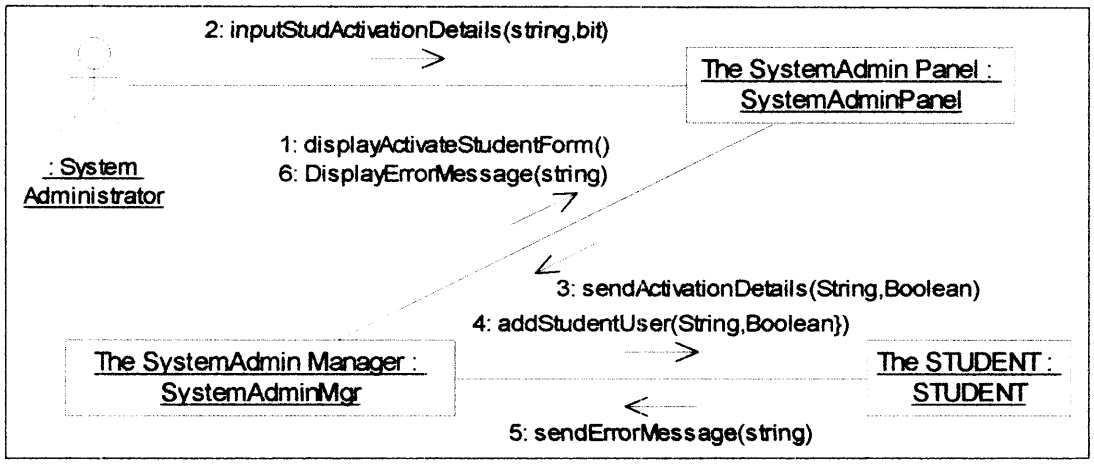
Activate User_Activate Staff



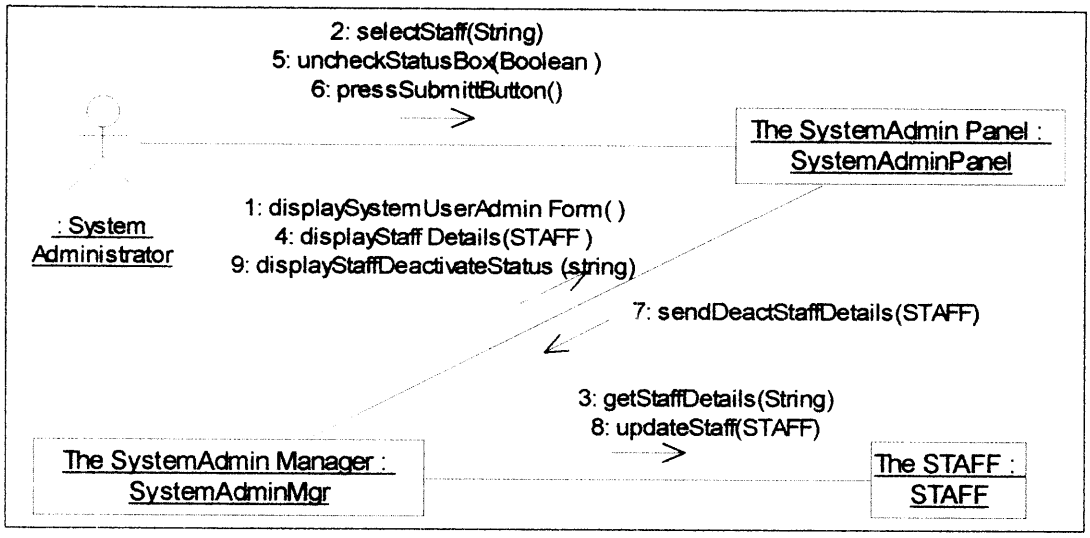
Activate User_Activate Staff/E1_if user already activated



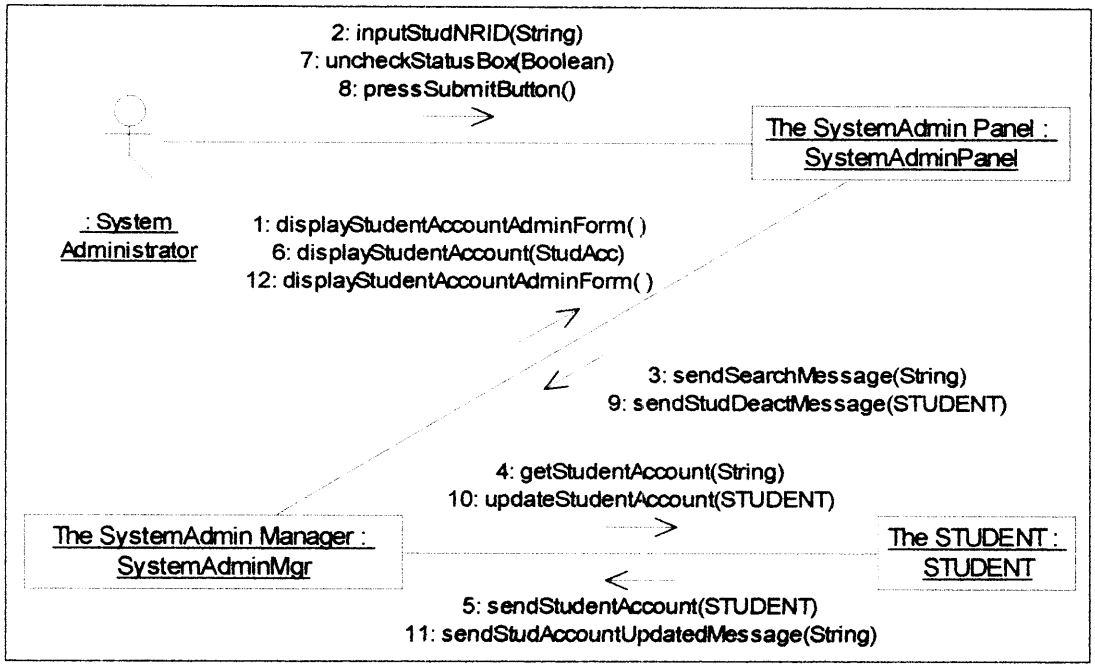
Activate User_Activate Student



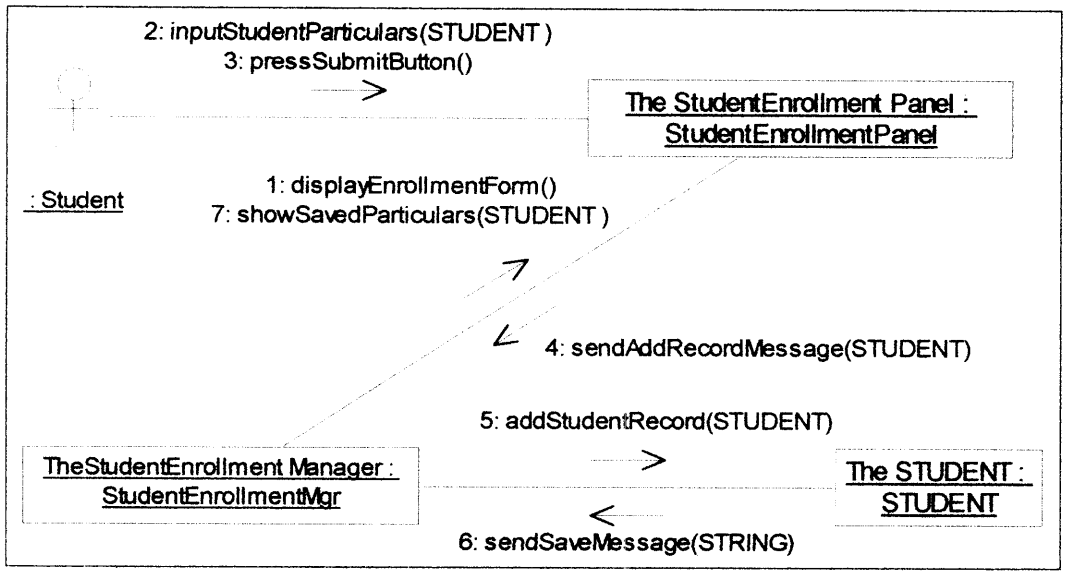
Activate User_Activate Student/ E1_if user already activated



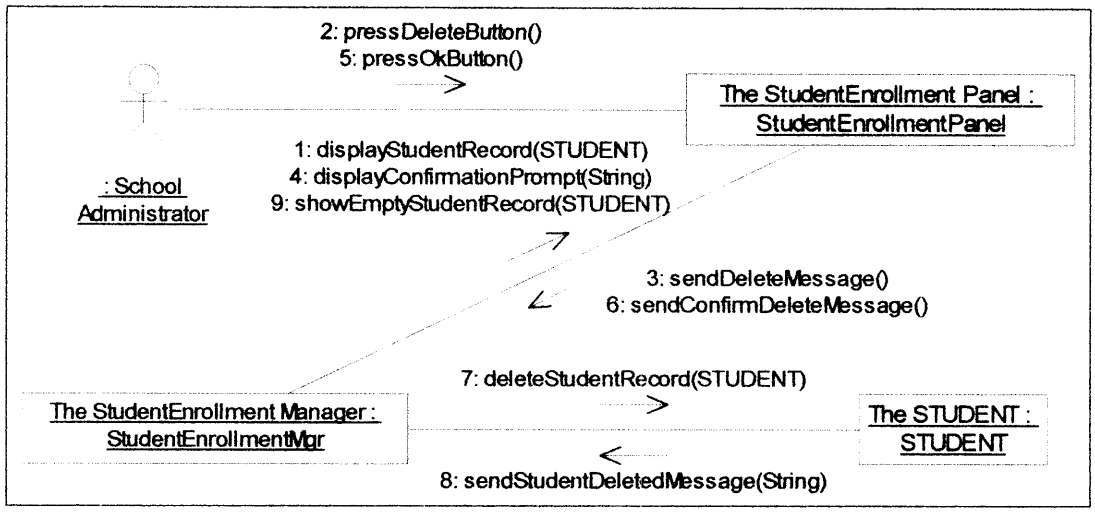
Deactivate User_Deactivate Staff



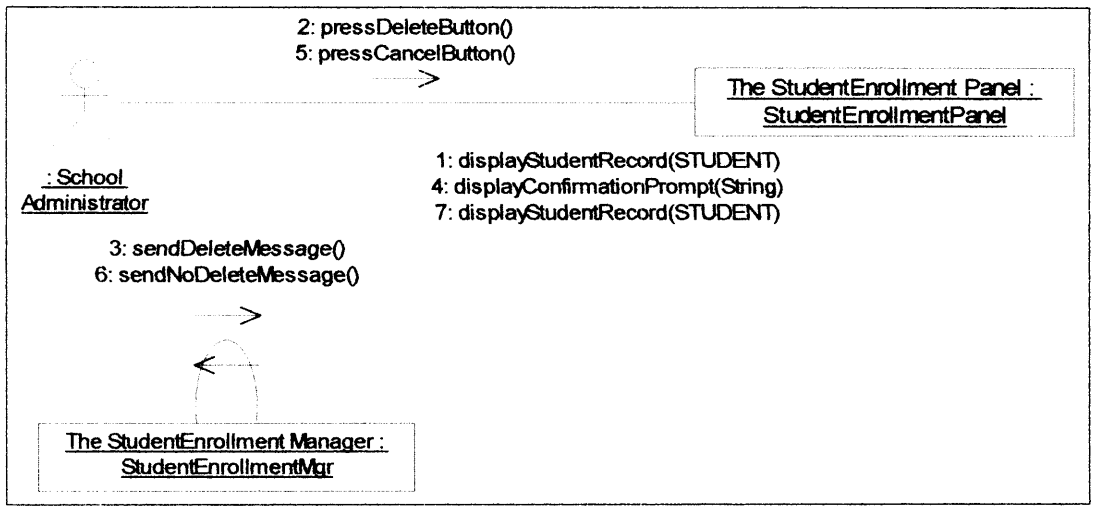
Deactivate User_Deactivate Student



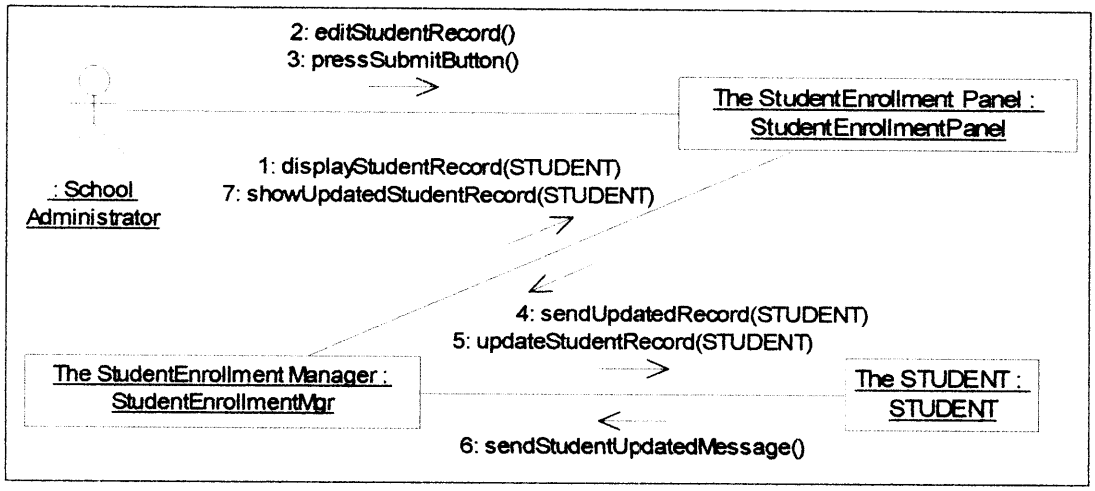
Add Student Particulars_Add Student Particulars



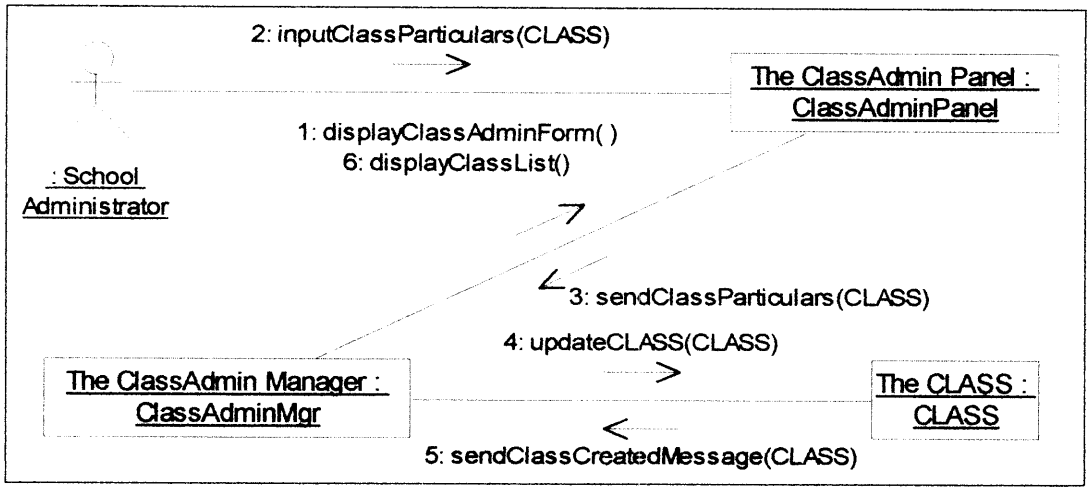
Delete Student Record_Delete Student Record



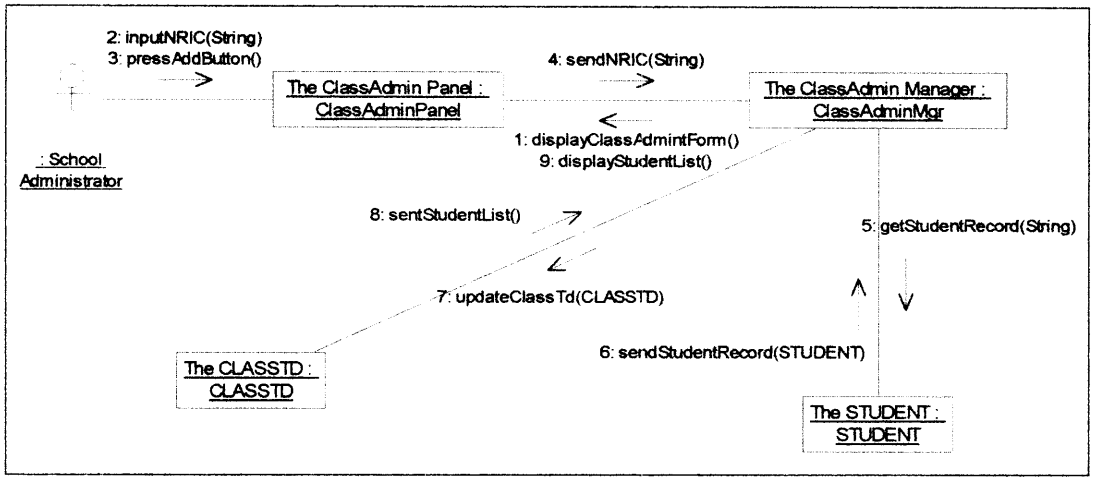
Delete Student Record/Delete Student Record/E1_if confirmation denied



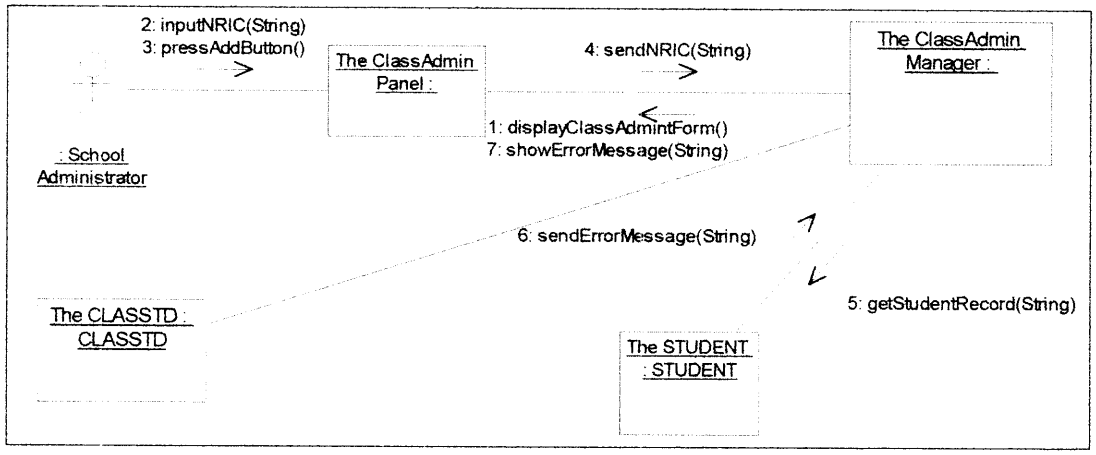
Update Student Record/Update Student Record



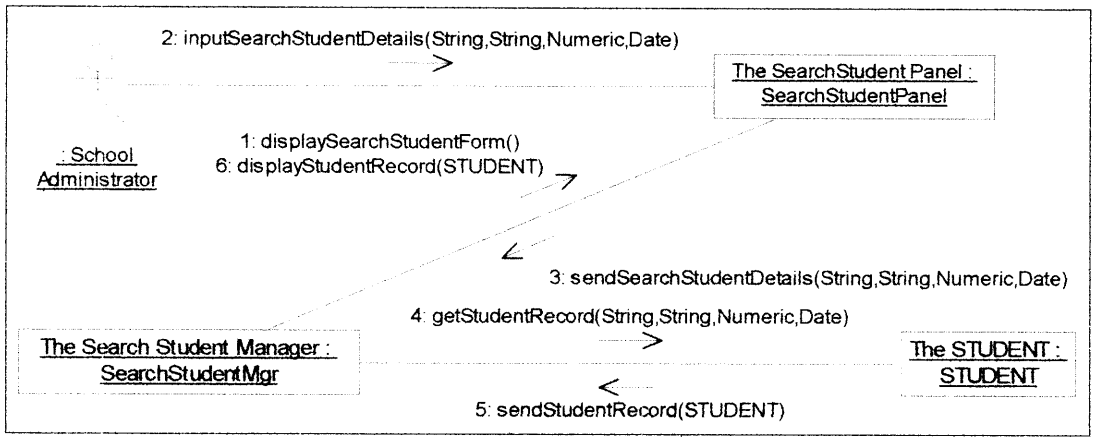
Generate Attendance Roster/Generate Class



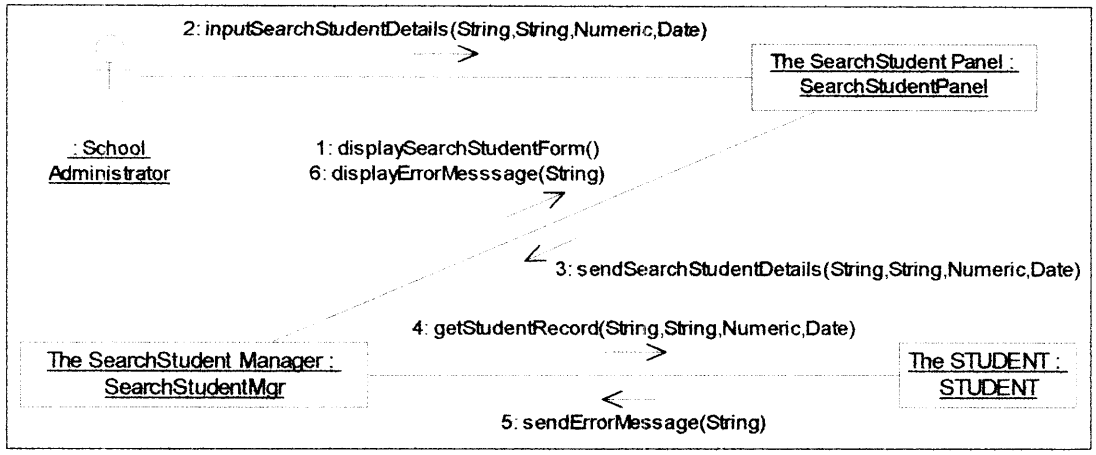
Add Class Member/Add Class Member



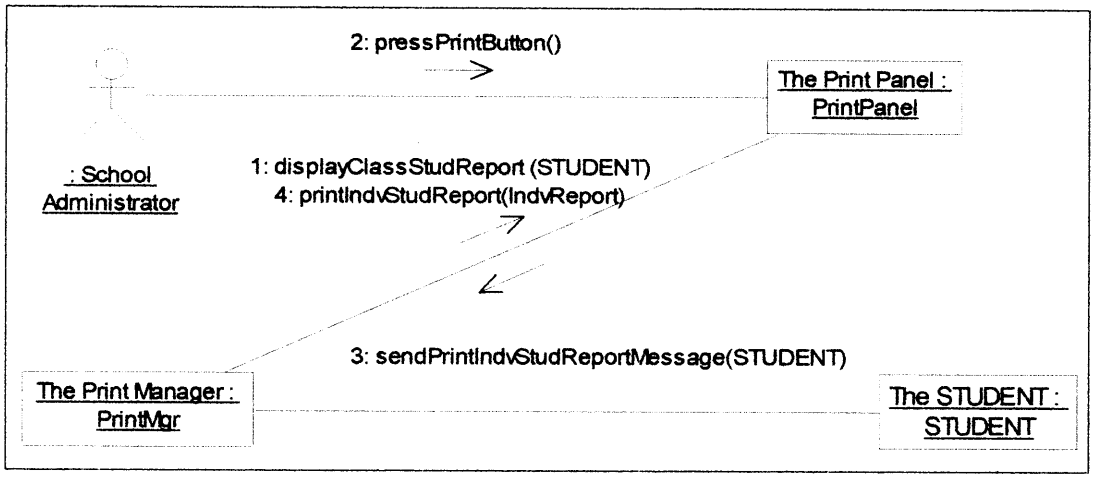
Add Class Member/E1_if record not found



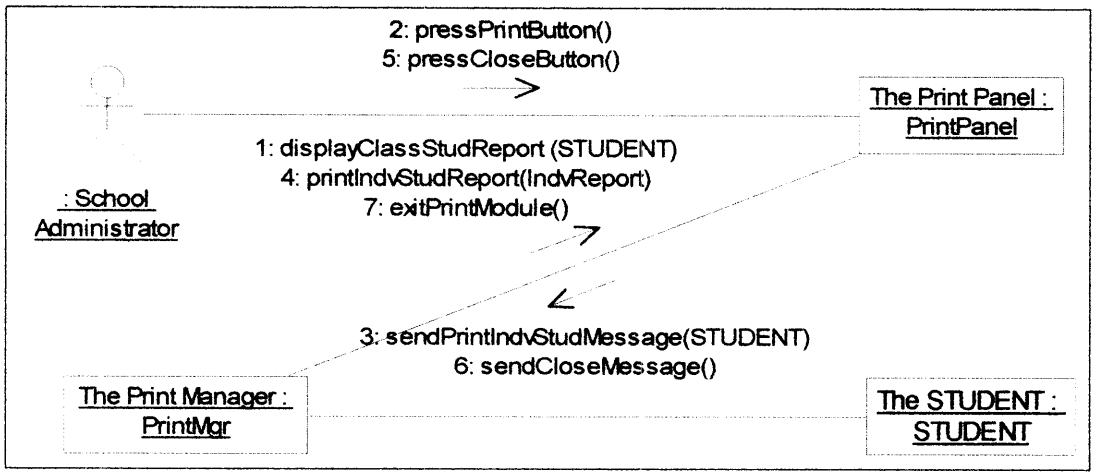
Search Student Record/Search Student Record



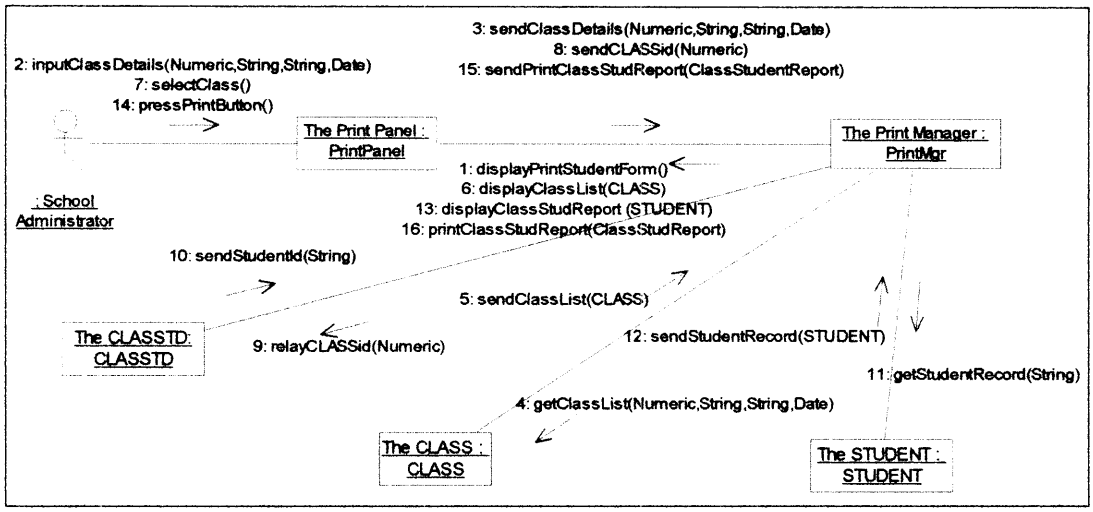
Search Student Record/E1_if student record not found



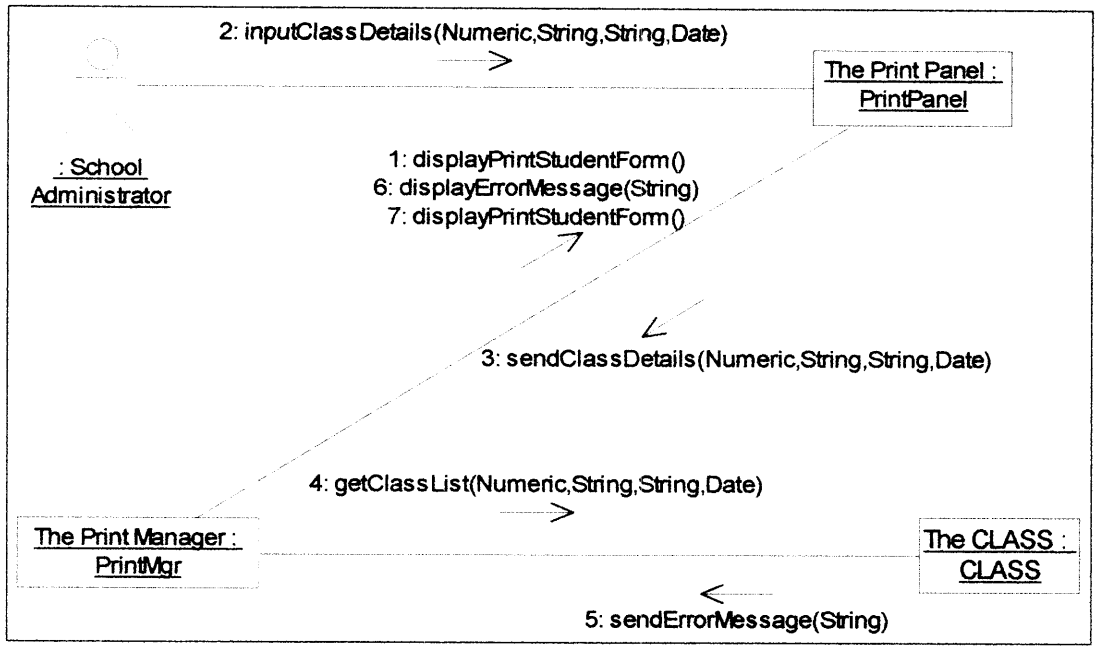
Print Individual Student Record/Print Individual Student Record



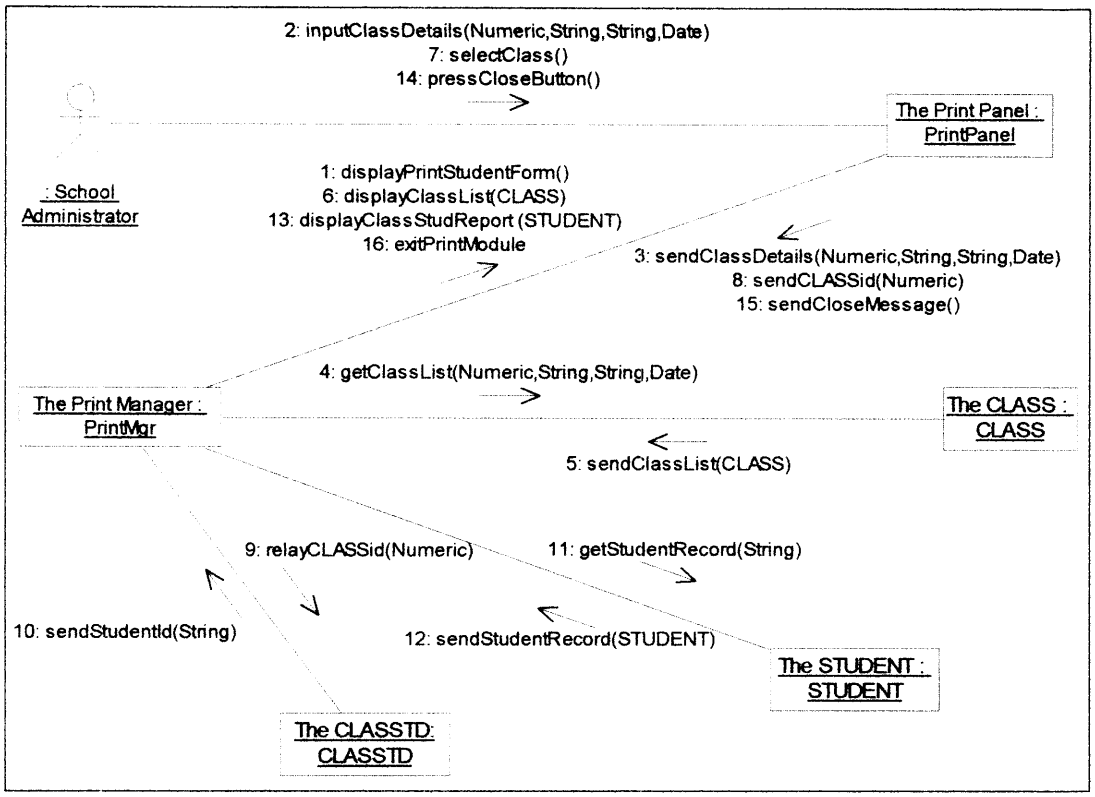
Print Individual Student Record/E1_if user cancel transaction



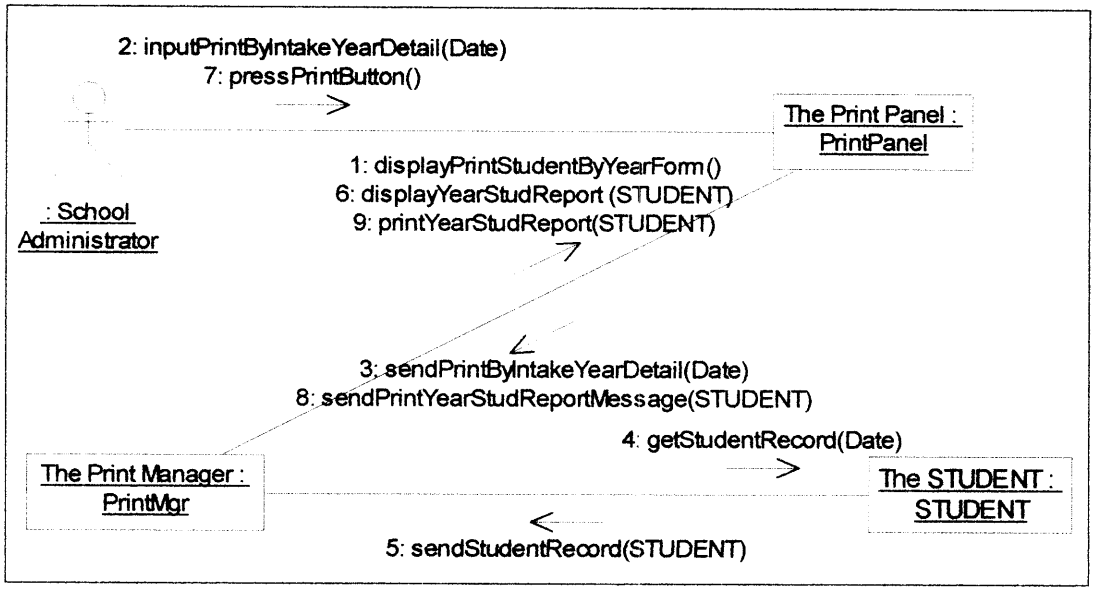
Print Student Record By Class



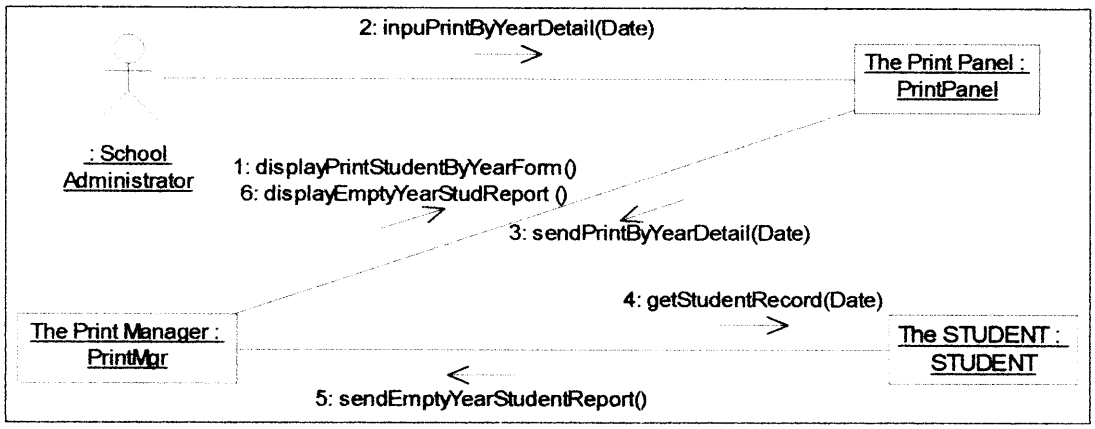
Print Student Record By Class/E1_if class not found



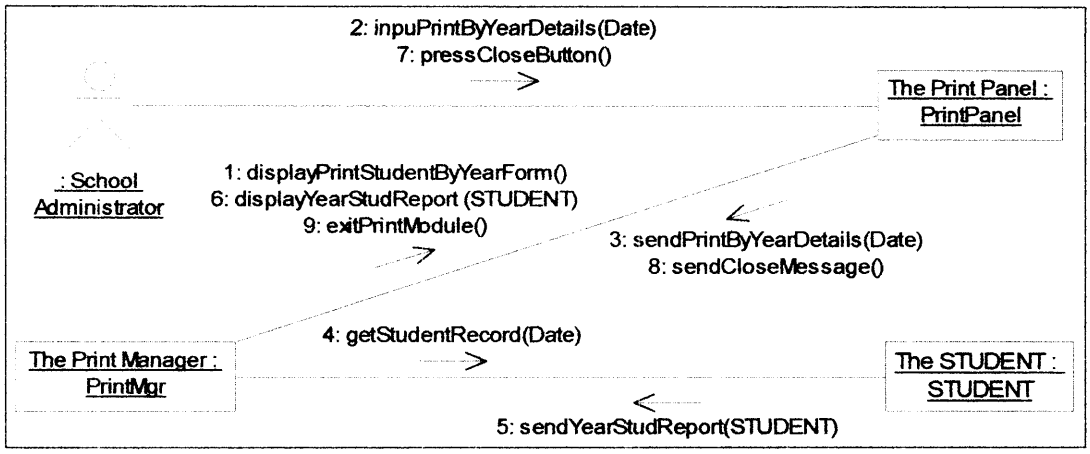
Print Student Record By Class/A1_if user cancel transaction



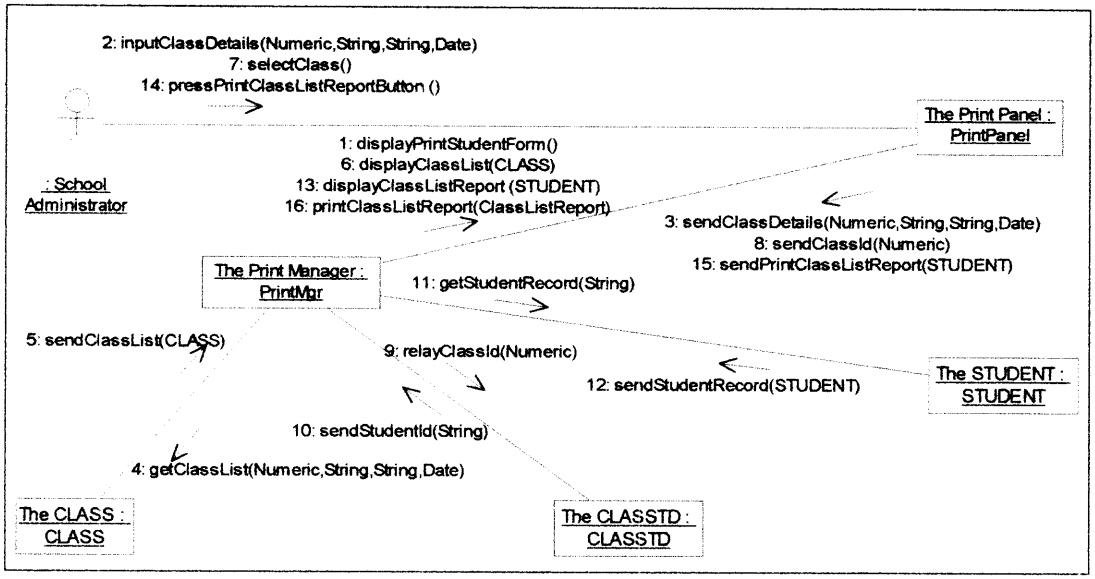
Print Student Record By Intake/ Print Student Record By Intake



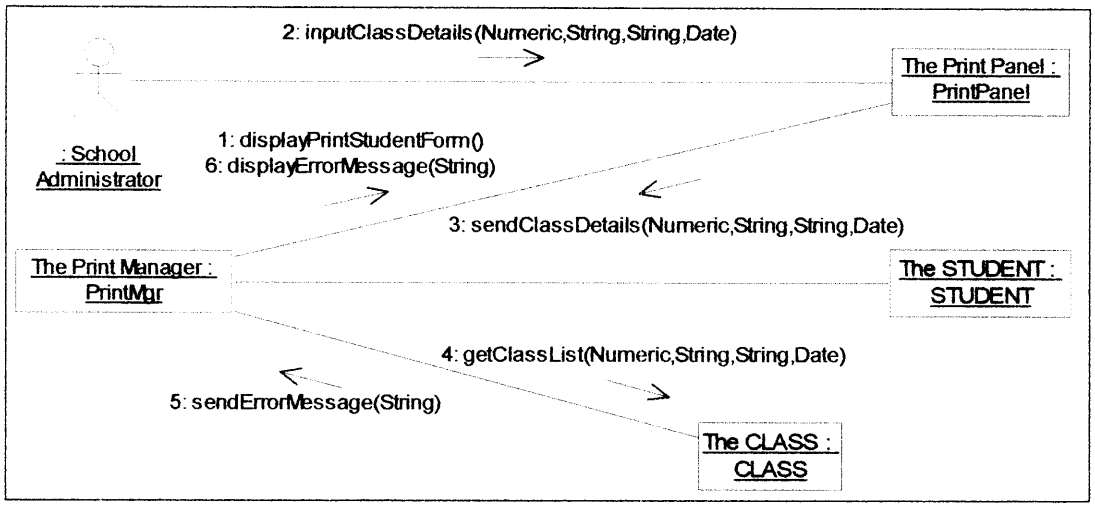
Print Student Record By Intake/E1_if student record not found



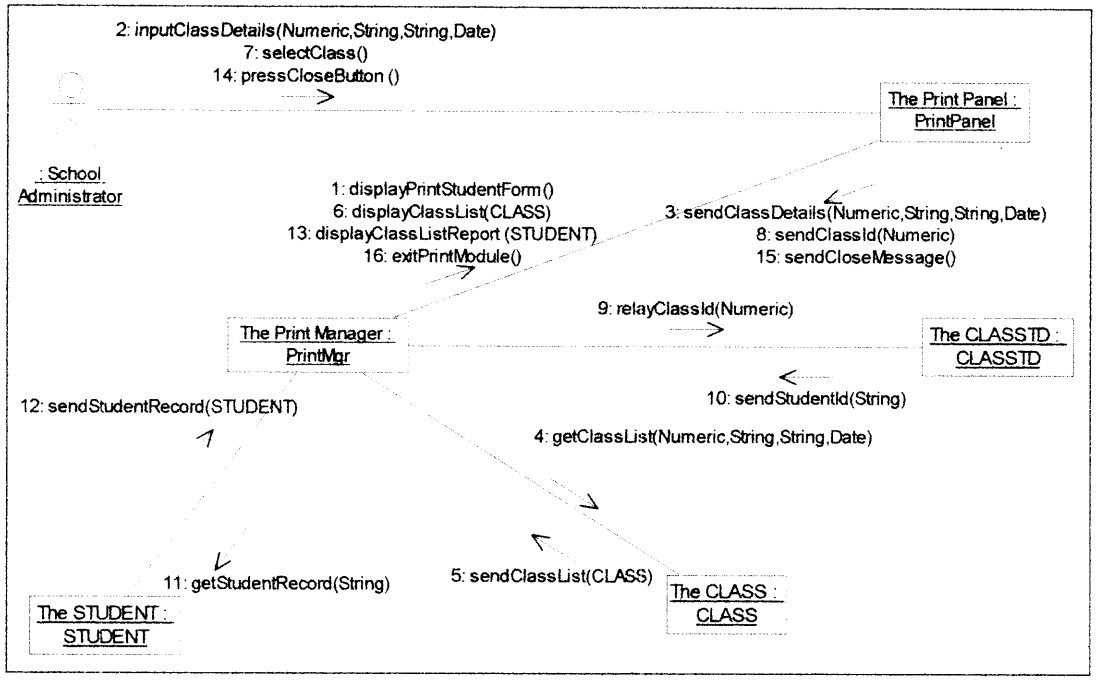
Print Student Record By Intake/A1_user cancel transaction



Print Class List/Print Class List



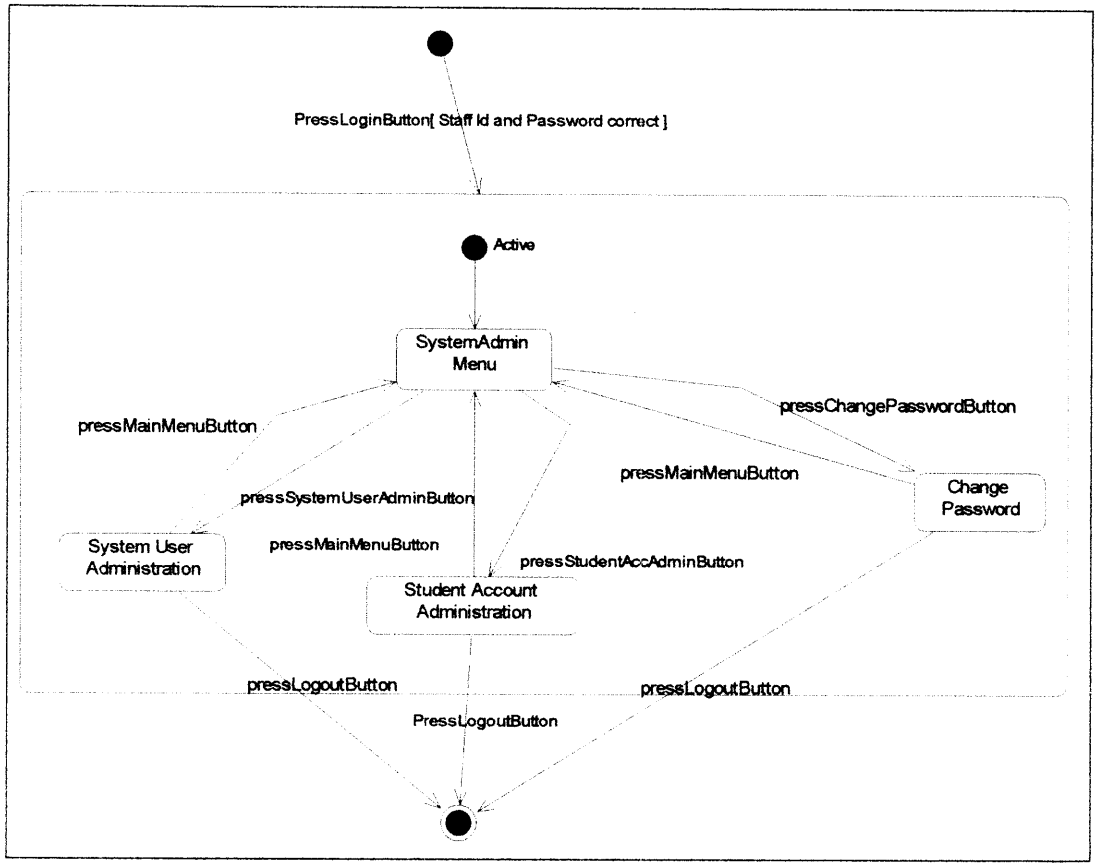
Print Class List/E1_if class not found



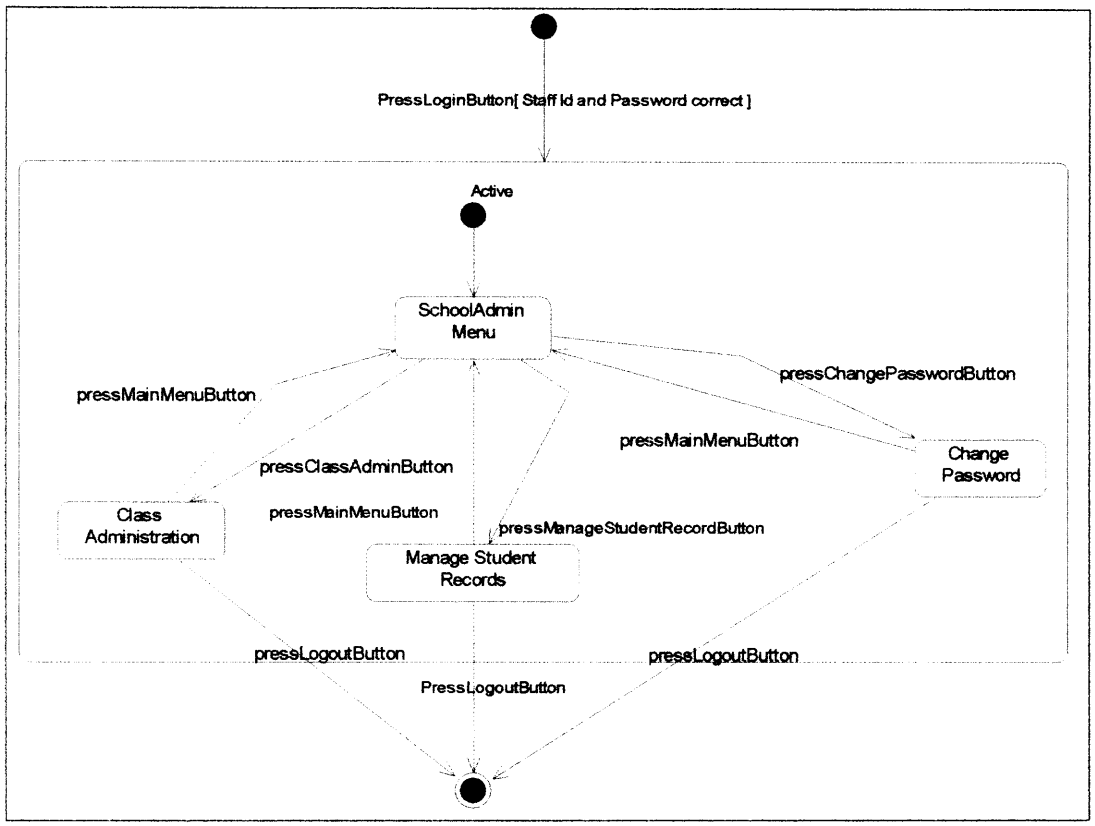
Print Class List/A1_if user cancels transaction

APPENDIX D

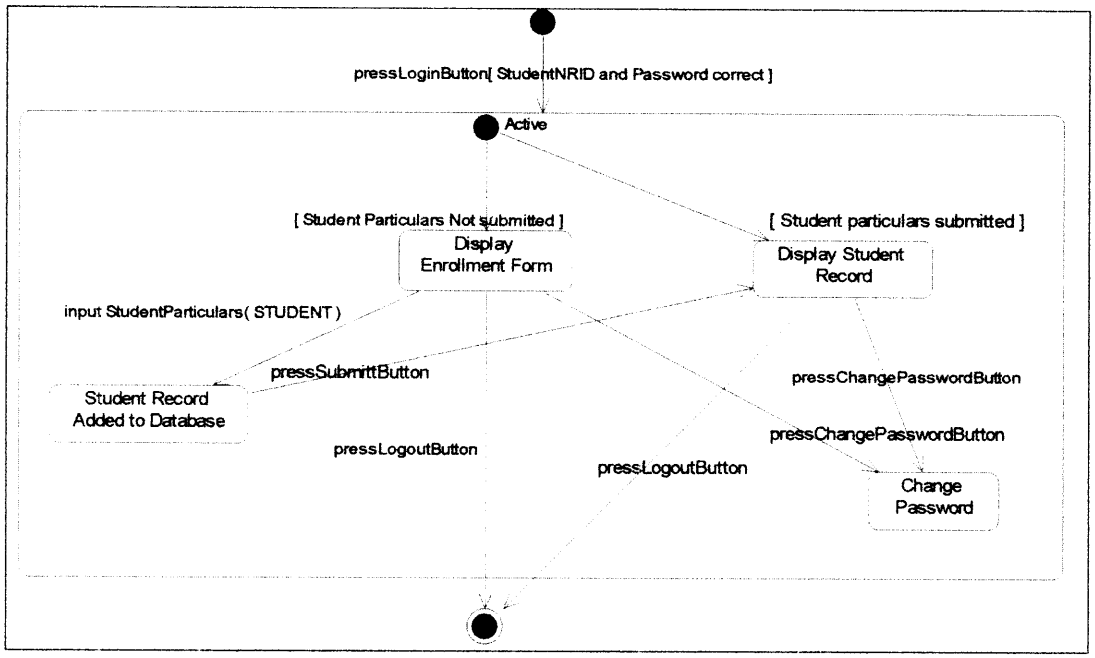
STATE DIAGRAM



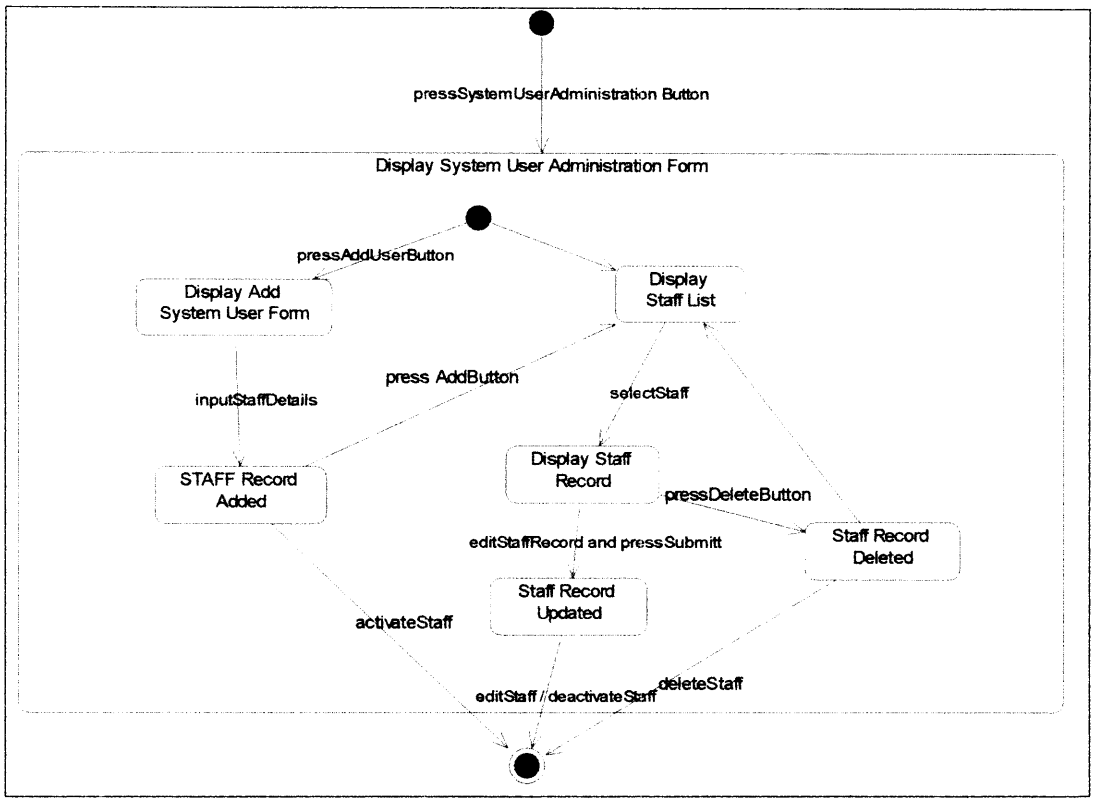
System Administration/ SystemAdministrator Login



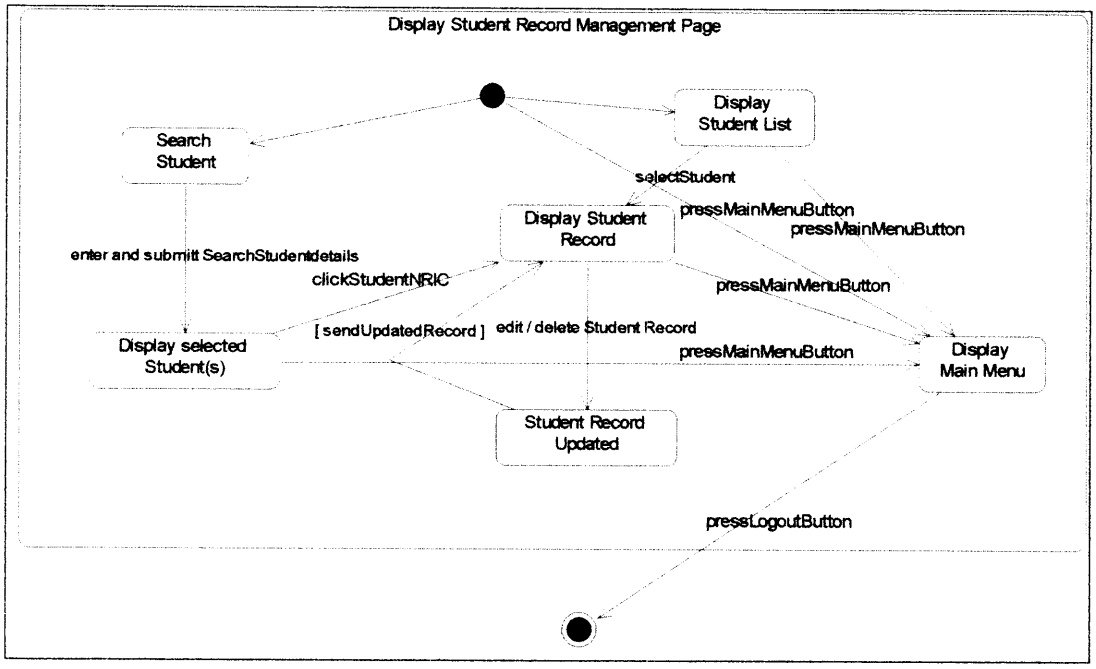
System Administration/Staff Login



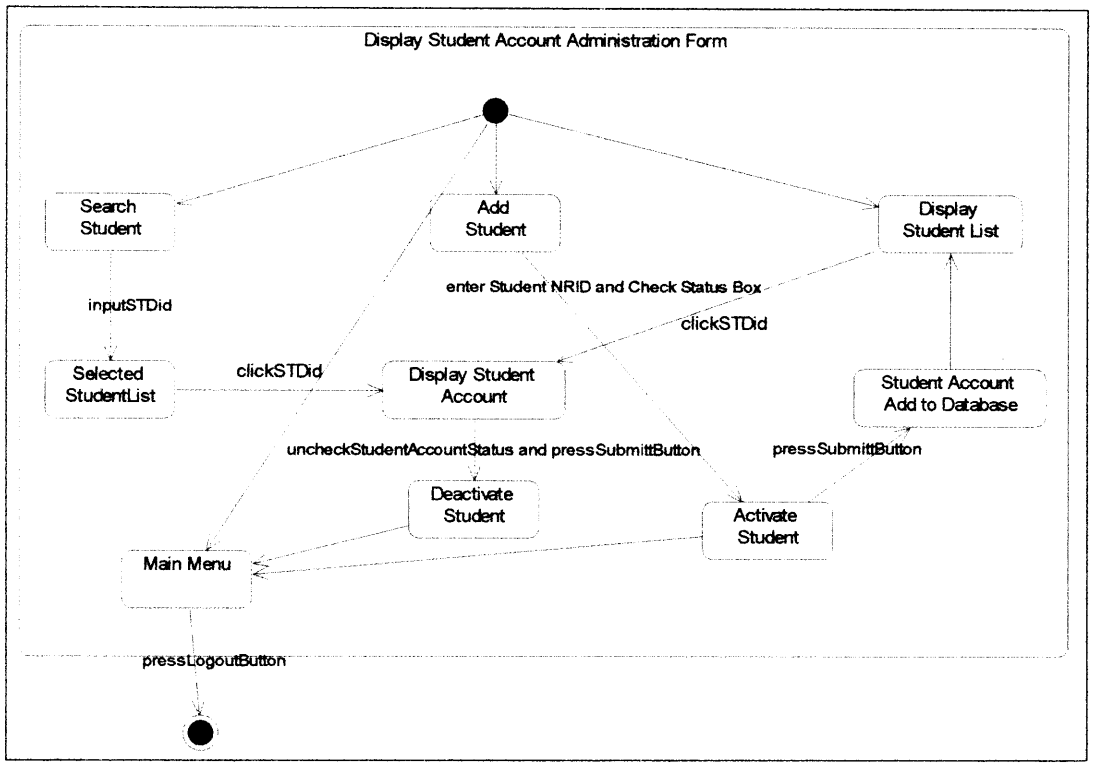
System Administration/ Student Login



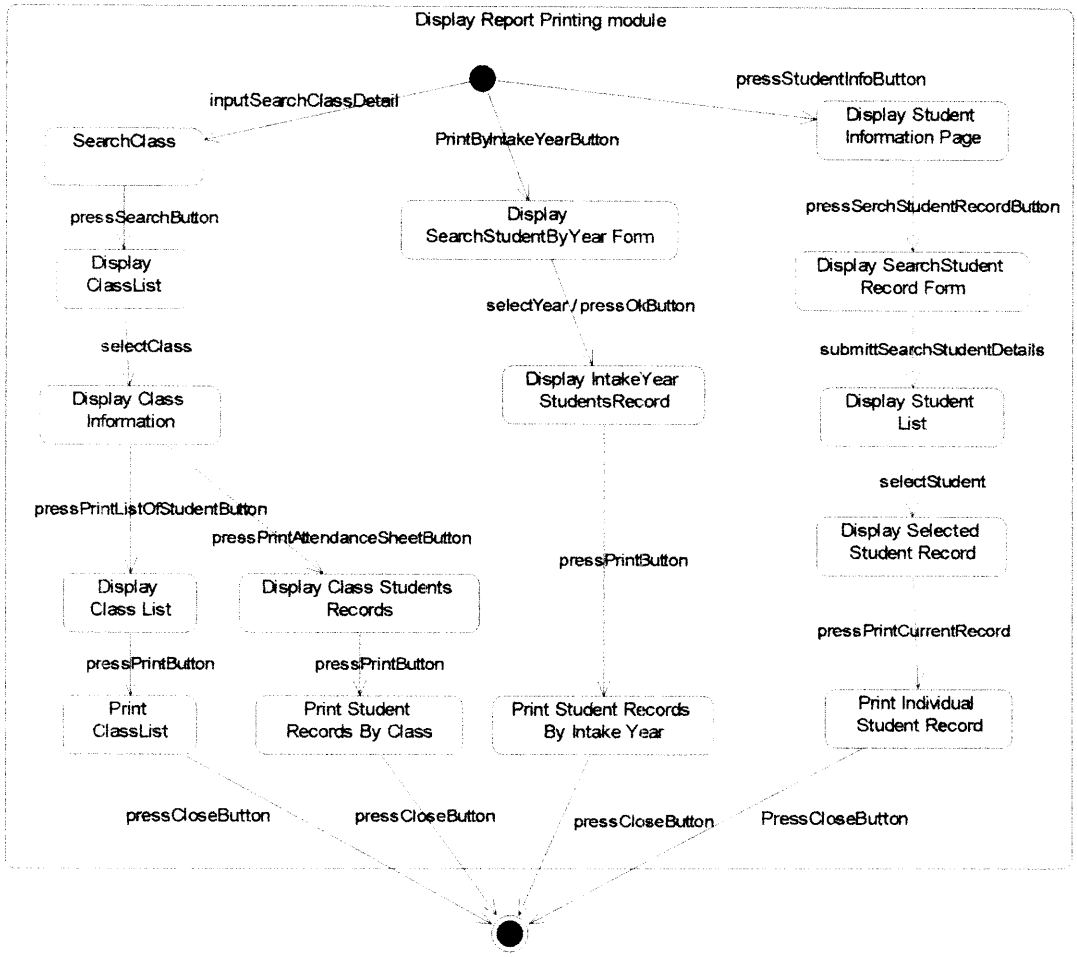
System Administration/SystemUser Administration



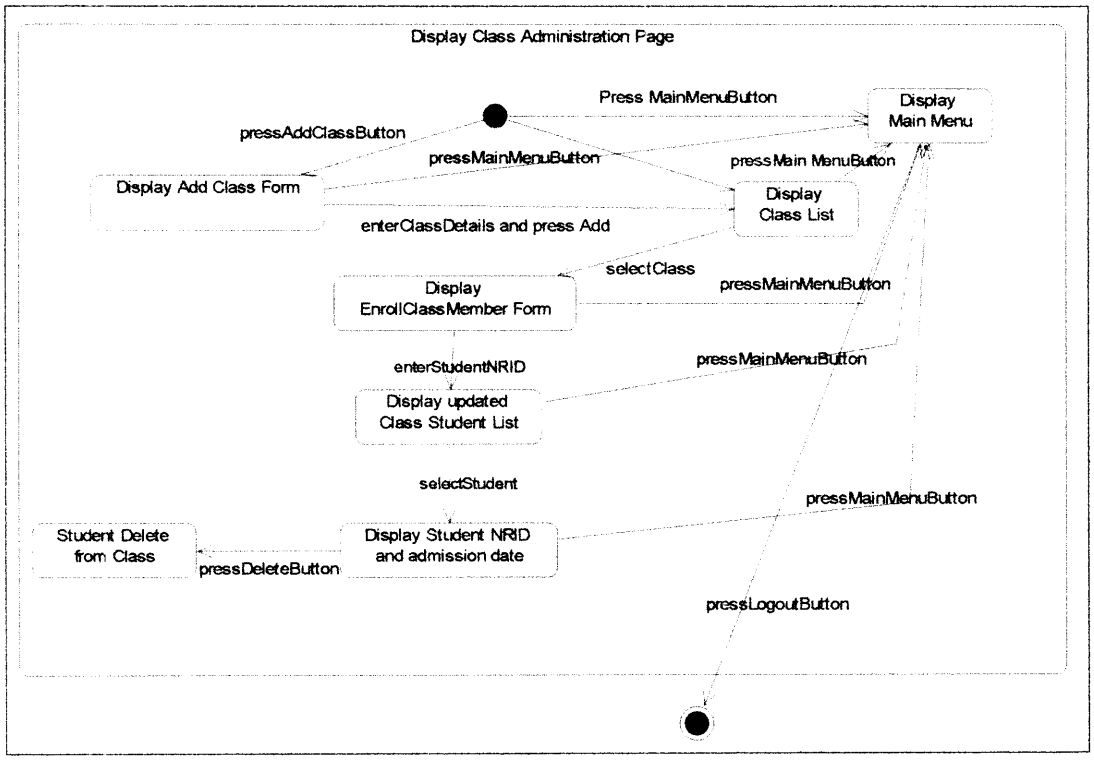
StudentEnrollment/Student Record management



System Administration/Student Account Administration



Print Management



Class Administration

APPENDIX C

Software Installation Guide

Installation Note

AssumingSQL server and IIS are installed, install the Printing Module with the program on the CD.

Next, copy the database from the "Database" folder to the pc and attach the said database to the SQL Server. Name the database "StudentRegSQL".

Copy the "SES" folder from the CD to the pc under the "Inetpub"folder.

Share the "Inetpub\SES\" to the Web and name it as "SES".

APPENDIX D

Source Codes

```
<%
Option Explicit

'Include Files 00-07EFBEEB
%>
<!-- #INCLUDE FILE="Adovas.asp" -->
<!-- #INCLUDE FILE="Classes.asp" -->
<%
'End Include Files

'Script Engine Version Check 80-A116D8C9
If ScriptEngineMajorVersion < 5 Then
    Response.Write "Sorry. This program requires VBScript 5.1 to run.<br> ↙
    You may upgrade your VBScript at http://www.microsoft.com/msdownload/ ↙
    vbscript/scripting.asp."
    Response.End
Else
    If ScriptEngineMajorVersion & ":" & ScriptEngineMinorVersion = "5:0"  ↙
    Then
        Response.Write "Due to a bug in VBScript 5.0, this program would  ↙
    crash your server. See http://support.microsoft.com/default.aspx?scid=↙
    kb;EN-US;q240811.<br>" & _
            "Upgrade your VBScript at http://www.microsoft.com/msdownload/↙
    vbscript/scripting.asp."
        Response.End
    End If
End If
'End Script Engine Version Check

'Initialise Common Variables 00-F01176F7
Dim CCSDateConstants
Dim ServerURL
Dim SecureURL
Dim TemplatesRepository
Dim EventCaller
Dim ParentPage
Dim DefaultDateFormat
Dim DefaultBooleanFormat
Dim IsMutipartEncoding
Dim objUpload
Dim UploadedFilesCount
Dim CCSConverter
Dim CCSRunTimeMessages
Set CCSRunTimeMessages = New clsRunTimeMessages

IsMutipartEncoding = False
If InStr(Request.ServerVariables("CONTENT_TYPE"),"multipart/form-data") > ↙
    0 Then
    On Error Resume Next
    Set objUpload = Server.CreateObject("Persits.Upload")
    objUpload.IgnoreNoPost = True

    UploadedFilesCount = objUpload.Save
    IsMutipartEncoding = True
    If Err.Number > 0 Then
        Response.Write "Persits uploading component ""Persits"" is not    ↙
    found. Please select another or install the component."
        Response.End
    End If
    On Error Goto 0
End If
Set TemplatesRepository = New clsCache_FileSystem

DefaultDateFormat = Array("dd", "/", "mm", "/", "yyyy")
DefaultBooleanFormat = Array("True", "False", Empty)
Set CCSConverter = New clsConverter
CCSConverter.DateFormat = DefaultDateFormat
CCSConverter.BooleanFormat = DefaultBooleanFormat
```

```
ServerURL = "http://localhost/SES/SystemAdmin/"
Set CCSDateConstants = New clsCCSDateConstants

Class clsCCSDateConstants

    Public Weekdays
    Public ShortWeekdays
    Public Months
    Public ShortMonths
    Public DateMasks

    Private Sub Class_Initialize()
        ShortWeekdays = Array("Sun", "Mon", "Tue", "Wed", "Thu", "Fri",
    "Sat")
        Weekdays = Array("Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday")
        ShortMonths =  Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
        Months = Array("January", "February", "March", "April", "May",
    "June", "July", "August", "September", "October", "November",
    "December")
        Set DateMasks = CreateObject("Scripting.Dictionary")
        DateMasks("d") = 0
        DateMasks("dd") = 2
        DateMasks("m") = 0
        DateMasks("mm") = 2
        DateMasks("mmm") = 3
        DateMasks("mmmm") = 0
        DateMasks("yy") = 2
        DateMasks("yyyy") = 4
        DateMasks("h") = 0
        DateMasks("hh") = 2
        DateMasks("H") = 0
        DateMasks("HH") = 2
        DateMasks("n") = 0
        DateMasks("nn") = 2
        DateMasks("s") = 0
        DateMasks("ss") = 2
        DateMasks("am/pm") = 2
        DateMasks("AM/PM") = 2
        DateMasks("A/P") = 1
        DateMasks("a/p") = 1
        DateMasks("w") = 0
        DateMasks("q") = 0
        DateMasks("S") = 0
    End Sub

    Private Sub Class_Terminate()
        Set DateMasks = Nothing
    End Sub

End Class

Const ccsInteger = 1
Const ccsFloat = 2
Const ccsText = 3
Const ccsDate = 4
Const ccsBoolean = 5
Const ccsMemo = 6

Const ccsGet = 1
Const ccsPost = 2
'End Initialize Common Variables

'Connection1 Connection Class 6-CB3AC8A4
Class clsDBConnection1

    Public ConnectionString
```

```
    Public User
    Public Password
    Public LastSQL
    Public Errors
    Public Converter

    Private mDateFormat
    Private mBooleanFormat
    Private objConnection
    Private blnState

    Private Sub Class_Initialize()
        ConnectionString = "Provider=SQLOLEDB.1;Persist Security Info=
    False;User ID=sa;Initial Catalog=StudentRegSQL;Data Source=(local)"
        User = "sa"
        Password = ""
        Set Converter = New clsConverter
        Converter.DateFormat = Array("yyyy", "-", "mm", "-", "dd", " ",
    "HH", ":", "nn", ":", "ss")
        Converter.BooleanFormat = Array(1, 0, Empty)
        Set objConnection = Server.CreateObject("ADODB.Connection")
        Set Errors = New clsErrors
    End Sub

    Public Property Get DateFormat()
      DateFormat = Converter.DateFormat
    End Property

    Public Property Get BooleanFormat()
      BooleanFormat = Converter.BooleanFormat
    End Property

    Sub Open()
        On Error Resume Next
        objConnection.Errors.Clear
        objConnection.Open ConnectionString, User, Password
        If Err.Number <> 0 then
            Response.Write "<div><h2>Unable to establish connection to
    database.</h2>"
            Response.Write "<ul><li><b>Error information:</b><br>"
            Response.Write Err.Source & " (0x" & Hex(Err.Number) & ")<br>"
            Response.Write Err.Description & "</li>"
            If Err.Number = -2147467259 then _
            Response.Write "<li><b>Other possible cause of this problem:</
    b><br>The database cannot be opened, most likely due to incorrect
    connection settings or insufficient security set on your database
    folder or file. <br>For more details please refer to <a href='http://
    support.microsoft.com/default.aspx?scid=kb;en-us;Q306518'>http://
    support.microsoft.com/default.aspx?scid=kb;en-us;Q306518</a></li>"
            Response.Write "</ul></div>"
            Response.End
        End If
    End Sub

    Sub Close()
        objConnection.Close
    End Sub

    Function Execute(varCMD)
        Dim ErrorMessage, objResult
        Errors.Clear
        Set objResult = Server.CreateObject("ADODB.Recordset")
        objResult.CursorType = adOpenForwardOnly
        objResult.LockType = adLockReadOnly
        If TypeName(varCMD) = "Command" Then
            Set varCMD.ActiveConnection = objConnection
            Set objResult.Source = varCMD
            LastSQL = varCMD.CommandText
```

```
        Else
            Set objResult.ActiveConnection = objConnection
            objResult.Source = varCMD
            LastSQL = varCMD
        End If
        On Error Resume Next
        objResult.Open
        Errors.AddError CCProcessError(objConnection)
        On Error Goto 0
        Set Execute = objResult
    End Function

    Property Get Connection()
        Set Connection = objConnection
    End Property

    Property Get State()
        State = objConnection.State
    End Property

    Function ToSQL(Value, ValueType)
        If CStr(Value) = "" OR IsEmpty(Value) Then
            ToSQL = "Null"
        Else
            Select Case ValueType
                Case ccsDate
                    Value = CCFormatDate(CDate(Value), DateFormat)
                Case ccsBoolean
                    Value= CCFormatBoolean(CBool(Value), BooleanFormat)
            End Select

            If ValueType = ccsInteger or ValueType = ccsFloat Then
                ToSQL = Replace(Value, ",", ".")
            ElseIf ValueType = ccsDate Then
                ToSQL = "'" & Replace(Value, "'", "''") & "'"
            Else
                ToSQL = "'" & Replace(Value, "'", "''") & "'"
            End If
        End If
    End Function


End Class
'End Connection1 Connection Class

'IIf 10-535BAA27
Function IIf(Expression, TrueResult, FalseResult)
    If CBool(Expression) Then _
        IIf = TrueResult _
    Else _
        IIf = FalseResult
End Function
'End IIf

'Print 80-06532167
Sub Print(Value)
    Response.Write CStr(Value)
End Sub
'End Print

'CCRaiseEvent 10-E59A6848
Function CCRaiseEvent(Events, EventName, Caller)
    Set EventCaller = Caller
    Dim Result : Result = Events(EventName)
    Set EventCaller = Nothing
    If VarType(Result) = vbEmpty Then _
        Result = True
    CCRaiseEvent = Result
```

```
End Function
'End CCRaiseEvent

'CCFormatError 00-3A89E5B6
Function CCFormatError(Title, Errors)
    Dim Result, i
    Result = "<p>Form: " & Title & "<br>"
    For i = 0 To Errors.Count - 1
        Result = Result & "Error: " & Errors.ErrorByNumber(i)
        If i < Errors.Count - 1 Then Result = Result & "<br>"
    Next
    Result = Result & "</p>"
    CCFormatError = Result
End Function
'End CCFormatError

'CCOpenRS 00-9E4633E0
Function CCOpenRS(RecordSet, SQL, Connection, ShowError)
    Dim ErrorMessage, Result
    Result = Empty
    Set RecordSet = Server.CreateObject("ADODB.Recordset")
    On Error Resume Next
    RecordSet.Open SQL, Connection, adOpenForwardOnly, adLockReadOnly,     ↙
    adCmdText
    ErrorMessage = CCProcessError(Connection)
    If NOT IsEmpty(ErrorMessage) Then
        If ShowError Then _
            Result = "SQL: " & CommandObject.CommandText & "<br>" & "Error↙
    : " & ErrorMessage & "<br>" _
        Else _
            Result = "Database error.<br>"
    End If
    On Error Goto 0
    CCOpenRS = Result
End Function
'End CCOpenRS

'CCOpenRSFromCmd 00-A2A33E0F
Function CCOpenRSFromCmd(RecordSet, CommandObject, ShowError)
    Dim ErrorMessage, Result
    Result = Empty
    Set RecordSet = Server.CreateObject("ADODB.Recordset")
    On Error Resume Next
    RecordSet.CursorType = adOpenForwardOnly
    RecordSet.LockType = adLockReadOnly
    RecordSet.Open CommandObject
    ErrorMessage = CCProcessError(CommandObject.ActiveConnection)
    If NOT IsEmpty(ErrorMessage) Then
        If ShowError Then _
            Result = "SQL: " & CommandObject.CommandText & "<br>" & "Error↙
    : " & ErrorMessage & "<br>" _
        Else _
            Result = "Database error.<br>"
    End If
    On Error Goto 0
    CCOpenRSFromCmd = Result
End Function
'End CCOpenRSFromCmd

'CCExecCmd 00-3DC993D0
Function CCExecCmd(CommandObject, ShowError)
    Dim ErrorMessage, Result
    Result = Empty
    On Error Resume Next
    CommandObject.Execute
    ErrorMessage = CCProcessError(CommandObject.ActiveConnection)
    If NOT IsEmpty(ErrorMessage) Then
        If ShowError Then _
```

```
            Result = "SQL: " & CommandObject.CommandText & "<br>" & "Error↙
    : " & ErrorMessage & "<br>" _
        Else _
            Result = "Database error.<br>"
    End If
    On Error Goto 0
    CCExecCmd = Result
End Function
'End CCExecCmd

'CCExecSQL @0-14C7B622
Function CCExecSQL(SQL, Connection, ShowError)
    Dim ErrorMessage, Result
    Result = Empty
    On Error Resume Next
    Connection.Execute(SQL)
    ErrorMessage = CCProcessError(Connection)
    If NOT IsEmpty(ErrorMessage) Then
        If ShowError Then _
            Result = "SQL: " & SQL & "<br>" & "Error: " & ErrorMessage & "↙
    <br>" _
        Else _
            Result = "Database error.<br>"
    End If
    On Error Goto 0
    CCExecSQL = Result
End Function
'End CCExecSQL

'CCToHTML @0-44721E9F4
Function CCToHTML(Value)
  If IsNull(Value) Then Value = ""
  CCToHTML = Server.HTMLEncode(Value)
End Function
'End CCToHTML

'CCToURL @0-23A93674
Function CCToURL(Value)
  If IsNull(Value) Then Value = ""
  CCToURL = Server.URLEncode(Value)
End Function
'End CCToURL

'CCGetValueHTML @0-30C69AED
Function CCGetValueHTML(RecordSet, FieldName)
  CCGetValueHTML = CCToHTML(CCGetValue(RecordSet, FieldName))
End Function
'End CCGetValueHTML

'CCGetValue @0-C9915067
Function CCGetValue(RecordSet, FieldName)
  Dim Result
  On Error Resume Next
  If RecordSet Is Nothing Then
    CCGetValue = Empty
  ElseIf (NOT RecordSet.EOF) AND (FieldName <> "") Then
    Result = RecordSet(FieldName)
    If IsNull(Result) Then _
      Result = Empty
    CCGetValue = Result
  Else
    CCGetValue = Empty
  End If
  On Error Goto 0
End Function
'End CCGetValue

'CCGetParam @0-4102C01B
```

```
Function CCGetDate(RecordSet, FieldName, arrDateFormat)
  Dim Result
  Result = CCGetValue(RecordSet, FieldName)
  If Not IsEmpty(arrDateFormat) Then
    If Not (VarType(Result) = vbDate OR VarType(Result) = vbEmpty) Then _
      If CCValidateDate(Result, arrDateFormat) Then _
        Result = CCParseDate(Result, arrDateFormat)
  End If
  CCGetDate = Result
End Function
'End CCGetDate

'CCGetBoolean 60-C64EED38
Function CCGetBoolean(RecordSet, FieldName, BooleanFormat)
  Dim Result
  Result = CCGetValue(RecordSet, FieldName)
  CCGetBoolean = CCParseBoolean(Result, BooleanFormat)
End Function
'End CCGetBoolean

'CCGetParam 60-1048E777
Function CCGetParam(ParameterName, DefaultValue)
  Dim ParameterValue : ParameterValue = ""

  If IsMutipartEncoding Then
    ParameterValue = objUpload.Form(ParameterName).Value
    If ParameterValue = "" Then ParameterValue = DefaultValue
  Else
    If Request.QueryString(ParameterName).Count > 0 Then
      ParameterValue = Request.QueryString(ParameterName)
    ElseIf Request.Form(ParameterName).Count > 0 Then
      ParameterValue = Request.Form(ParameterName)
    Else
      ParameterValue = DefaultValue
    End If
  End If

  CCGetParam = ParameterValue
End Function
'End CCGetParam

'CCGetFromPost 60-E740B202
Function CCGetFromPost(ParameterName, DefaultValue)
  Dim ParameterValue : ParameterValue = Empty

  If IsMutipartEncoding Then
    ParameterValue = objUpload.Form(ParameterName).Value
    If ParameterValue = "" Then ParameterValue = DefaultValue
  Else
    ParameterValue = Request.Form(ParameterName)
    If IsEmpty(ParameterValue) Then
      ParameterValue = DefaultValue
    End If
  End If

  CCGetFromPost = ParameterValue
End Function
'End CCGetFromPost

'CCGetFromGet 60-706B6115
Function CCGetFromGet(ParameterName, DefaultValue)
  Dim ParameterValue : ParameterValue = Empty
  ParameterValue = Request.QueryString(ParameterName)
  If IsEmpty(ParameterValue) Then _
    ParameterValue = DefaultValue
  CCGetFromGet = ParameterValue
End Function
'End CCGetFromGet
```

```
'CCToSQL {0-CA2C334A
Function CCToSQL(Value, ValueType)
  If CStr(Value) = "" OR IsEmpty(Value) Then
   CCToSQL = "Null"
  Else
    If ValueType = "Integer" or ValueType = "Float" Then
      CCToSQL = Replace(CDbl(Value), ",", ".")
    Else
      CCToSQL = "'" & Replace(Value, "'", "''") & "'"
    End If
  End If
End Function
'End CCToSQL

'CCDLookUp {0-7AA7ED74
Function CCDLookUp(ColumnName, TableName, Where, Connection)
  Dim RecordSet
  Dim Result
  Dim SQL
  Dim ErrorMessage

  SQL = "SELECT " & ColumnName
  If Len(CStr(TableName)) > 0 Then SQL = SQL & " FROM "  & TableName
  If Len(CStr(Where))     > 0 Then SQL = SQL & " WHERE " & Where

  Set RecordSet = Connection.Execute(SQL)
  ErrorMessage = CCProcessError(Connection)
  If NOT IsEmpty(ErrorMessage) Then
    PrintDBError "CCDLookUp function", SQL, ErrorMessage
  End If
  On Error Goto 0
  Result = CCGetValue(RecordSet, 0)
  CCDLookUp = Result
End Function
'End CCDLookUp

'PrintDBError {0-3D5DDA9A
Sub PrintDBError(Source, SQL, ErrorMessage)
  Dim CommandText
  Dim SourceText
  Dim ErrorText

  If Source <> "" Then SourceText = "<b>Source:</b> " & Source & "<br>"
  If SQL <> "" Then CommandText = "<b>Command Text:</b> " & SQL & "<br>"
  If ErrorMessage <> "" Then ErrorText = "<b>Error description:</b> " &
    ErrorMessage & "</div>"

  Response.Write "<div style=""background-color: rgb(250, 250, 250); " & _
    "border: solid 1px rgb(200, 200, 200);"">" & SourceText
  Response.Write CommandText & ErrorText
End Sub
'End PrintDBError

'CCGetCheckBoxValue {0-ABCF54E0
Function CCGetCheckBoxValue(Value, CheckedValue, UncheckedValue,
    ValueType)
  If isEmpty(Value) Then
    If UncheckedValue = "" Then
      CCGetCheckBoxValue = "Null"
    Else
      If ValueType = "Integer" or ValueType = "Float" Then
        CCGetCheckBoxValue = UncheckedValue
      Else
        CCGetCheckBoxValue = "'" & Replace(UncheckedValue, "'", "''") & "
    '"
      End If
    End If
```

```
  Else
    If CheckedValue = "" Then
      CCGetCheckBoxValue = "Null"
    Else
      If ValueType = "Integer" OR ValueType = "Float" Then
        CCGetCheckBoxValue = CheckedValue
      Else
        CCGetCheckBoxValue = "'" & Replace(CheckedValue, "'", "''") & "'"
      End If
    End If
  End If
End Function
'End CCGetCheckBoxValue

'CCGetValFromLOV 60-50418901
Function CCGetValFromLOV(Value, ListOfValues)
  Dim I
  Dim Result : Result = ""
  If (Ubound(ListOfValues) MOD 2) = 1 Then
    For I = 0 To Ubound(ListOfValues) Step 2
      If CStr(Value) = CStr(ListOfValues(I)) Then Result = ListOfValues(I ↙
    + 1)
    Next
  End If
  CCGetValFromLOV = Result
End Function
'End CCGetValFromLOV

'CCProcessError 60-A3A26540
Function CCProcessError(Connection)
  If Connection.Errors.Count > 0 Then
    If TypeName(Connection) = "Connection" Then
      CCProcessError = Connection.Errors(0).Description & " (" &          ↙
    Connection.Errors(0).Source & ")"
    Else
      CCProcessError = Connection.Errors.ToString
    End If
  ElseIf NOT (Err.Description = "") Then
    CCProcessError = Err.Description
  Else
    CCProcessError = Empty
  End If
end Function
'End CCProcessError

'CCGetRequestParam 60-227B5573
Function CCGetRequestParam(ParameterName, Method)
  Dim ParameterValue

  If Method = ccsGet Then
    ParameterValue = Request.QueryString(ParameterName)
  ElseIf Method = ccsPost Then
    If IsMutipartEncoding Then
      ParameterValue = objUpload.Form(ParameterName).Value
      If Len(ParameterValue) = 0 Then
        Dim File
        Set File = objUpload.Files(ParameterName)
        If Not File is Nothing Then ParameterValue = File.FileName
      End If
    Else
      ParameterValue = Request.Form(ParameterName)
    End If
  End If
  If CStr(ParameterValue) = "" Then _
    ParameterValue = Empty

  CCGetRequestParam = ParameterValue
End Function
```

```
Function CCGetRequestMultipleParam(ParameterName, Method)
  Dim ParameterValues(), ParamCount, i

  If Method = ccsGet Then
    ParamCount = Request.QueryString(ParameterName).Count
    ReDim ParameterValues (ParamCount)
    For i = 1 To ParamCount
      ParameterValues(i) = Request.QueryString(ParameterName)(i)
      If CStr(ParameterValues(i)) = "" Then ParameterValues(i) = Empty
    Next
  ElseIf Method = ccsPost Then

    If IsMutipartEncoding Then
      Dim TempArray
      TempArray = Split(objUpload.Form(ParameterName).Value, ", ")
      ParamCount = UBound(TempArray) + 1
      ReDim ParameterValues (ParamCount)
      For i = 0 to ParamCount - 1
        ParameterValues(i+1) = TempArray(i)
      Next
    Else
      ParamCount = Request.Form(ParameterName).Count
      ReDim ParameterValues (ParamCount)
      For i = 1 To ParamCount
        ParameterValues(i) = Request.Form(ParameterName)(i)
        If CStr(ParameterValues(i)) = "" Then ParameterValues(i) = Empty
      Next
    End If

  End If

  CCGetRequestMultipleParam = ParameterValues
End Function



'End CCGetRequestParam

'CCGetQueryString 60-CBD7B22E
Function CCGetQueryString(CollectionName, RemoveParameters)
  Dim QueryString, PostData

  If CollectionName = "Form" Then
    QueryString = CCCollectionToString(Request.Form, RemoveParameters)
  ElseIf CollectionName = "QueryString" Then
    QueryString = CCCollectionToString(Request.QueryString,
    RemoveParameters)
  ElseIf CollectionName = "All" Then
    QueryString = CCCollectionToString(Request.QueryString,
    RemoveParameters)
    PostData = CCCollectionToString(Request.Form, RemoveParameters)
    If Len(PostData) > 0 and Len(QueryString) > 0 Then _
      QueryString = QueryString & "&" & PostData _
    Else _
      QueryString = QueryString & PostData
  Else
    Err.Raise 1050, "Common Functions. CCGetQueryString Function", _
      "The CollectionName contains an illegal value."
  End If

  CCGetQueryString = QueryString
End Function
'End CCGetQueryString

'CCCollectionToString 60-92DC4E55
Function CCCollectionToString(ParametersCollection, RemoveParameters)
```

```
  Dim ItemName, ItemValue, Result, Remove, I

  For Each ItemName In ParametersCollection
    Remove = false
    If IsArray(RemoveParameters) Then
      For I = 0 To UBound(RemoveParameters)
        If RemoveParameters(I) = ItemName Then
          Remove = True
          Exit For
        End If
      Next
    End If
    If Not Remove Then
      If ParametersCollection(ItemName).Count = 1 Then
        Result = Result & _
          "&" & ItemName & "=" & Server.URLEncode(ParametersCollection
    (ItemName))
      Else
        For Each ItemValue In ParametersCollection(ItemName)
          Result = Result & _
            "&" & ItemName & "=" & Server.URLEncode(ItemValue)
        Next
      End If
    End If
  Next

  If Len(Result) > 0 Then _
    Result = Mid(Result, 2)
  CCCollectionToString = Result
End Function
'End CCCollectionToString

'CCAddZero @0-B5648410
Function CCAddZero(Value, ResultLength)
  Dim CountZero, I

  CountZero = ResultLength - Len(Value)
  For I = 1 To CountZero
    Value = "0" & Value
  Next
  CCAddZero = Value
End Function
'End CCAddZero

'CCGetAMPM @0-CB9EADB7
Function CCGetAMPM(HoursNumber, AnteMeridiem, PostMeridiem)
  If HoursNumber >= 0 And HoursNumber < 12 Then
    CCGetAMPM = AnteMeridiem
  Else
    CCGetAMPM = PostMeridiem
  End If
End Function
'End CCGetAMPM

'CC12Hour @0-12B00A37
Function CC12Hour(HoursNumber)
  If HoursNumber = 0 Then
    HoursNumber = 12
  ElseIf HoursNumber > 12 Then
    HoursNumber = HoursNumber - 12
  End If
  CC12Hour = HoursNumber
End Function
'End CC12Hour

'CCDBFormatByType @0-53172185
Function CCDBFormatByType(Variable)
  Dim Result
```

```
  If VarType(Variable) = vbString Then
    If LCase(Variable) = "null" Then
      Result = Variable
    Else
      Result = "'" & Variable & "'"
    End If
  Else
    Result = CStr(Variable)
  End If
  CCDBFormatByType = Result
End Function

'End CCDBFormatByType

'CCFormatDate 60-30B0C52B
Function CCFormatDate(DateToFormat, FormatMask)
  Dim ResultArray(), I, Result
  If VarType(DateToFormat) = vbEmpty Then
    Result = Empty
  ElseIf VarType(DateToFormat) <> vbDate Then
    Err.Raise 4000, "CCFormatDate function. Type mismatch."
  ElseIf IsEmpty(FormatMask) Then
    Result = CStr(DateToFormat)
  Else
    ReDim ResultArray(UBound(FormatMask))
    For I = 0 To UBound(FormatMask)
      Select Case FormatMask(I)
        Case "d" ResultArray(I) = Day(DateToFormat)
        Case "w" ResultArray(I) = Weekday(DateToFormat)
        Case "m" ResultArray(I) = Month(DateToFormat)
        Case "q" ResultArray(I) = (Month(DateToFormat) + 3) \ 4
        Case "y" ResultArray(I) = (DateDiff("d", "1/1" & "/" & Year        ↙
    (DateToFormat), Month(DateToFormat) & "/" & Day(DateToFormat) & "/" & ↙
    Year(DateToFormat)) + 1)
        Case "h" ResultArray(I) = CC12Hour(Hour(DateToFormat))
        Case "H" ResultArray(I) = Hour(DateToFormat)
        Case "n" ResultArray(I) = Minute(DateToFormat)
        Case "s" ResultArray(I) = Second(DateToFormat)
        Case "dd" ResultArray(I) = CCAddZero(Day(DateToFormat), 2)
        Case "ww" ResultArray(I) = (DateDiff("ww", "1/1" & "/" & Year      ↙
    (DateToFormat), Month(DateToFormat) & "/" & Day(DateToFormat) & "/" & ↙
    Year(DateToFormat)) + 1)
        Case "mm" ResultArray(I) = CCAddZero(Month(DateToFormat), 2)
        Case "yy" ResultArray(I) = Right(Year(DateToFormat), 2)
        Case "hh" ResultArray(I) = CCAddZero(CC12Hour(Hour(DateToFormat)),↙
     2)
        Case "HH" ResultArray(I) = CCAddZero(Hour(DateToFormat), 2)
        Case "nn" ResultArray(I) = CCAddZero(Minute(DateToFormat), 2)
        Case "ss" ResultArray(I) = CCAddZero(Second(DateToFormat), 2)
        Case "S" ResultArray(I) = "000"
        Case "ddd" ResultArray(I) = CCSDateConstants.ShortWeekdays(Weekday↙
    (DateToFormat) - 1)
        Case "mmm" ResultArray(I) = CCSDateConstants.ShortMonths(Month     ↙
    (DateToFormat) - 1)
        Case "A/P" ResultArray(I) = CCGetAMPM(Hour(DateToFormat), "A", "P"↙
    )
        Case "a/p" ResultArray(I) = CCGetAMPM(Hour(DateToFormat), "a", "p"↙
    )
        Case "dddd" ResultArray(I) = CCSDateConstants.Weekdays(Weekday     ↙
    (DateToFormat) - 1)
        Case "mmmm" ResultArray(I) = CCSDateConstants.Months(Month         ↙
    (DateToFormat) - 1)
        Case "yyyy" ResultArray(I) = Year(DateToFormat)
        Case "AM/PM" ResultArray(I) = CCGetAMPM(Hour(DateToFormat), "AM", ↙
    "PM")
        Case "am/pm" ResultArray(I) = CCGetAMPM(Hour(DateToFormat), "am", ↙
    "pm")
        Case "LongDate" ResultArray(I) = FormatDateTime(DateToFormat,      ↙
```

```
    vbLongDate)
        Case "LongTime" ResultArray(I) = FormatDateTime(DateToFormat,
    vbLongTime)
        Case "ShortDate" ResultArray(I) = FormatDateTime(DateToFormat,
    vbShortDate)
        Case "ShortTime" ResultArray(I) = FormatDateTime(DateToFormat,
    vbShortTime)
        Case "GeneralDate" ResultArray(I) = FormatDateTime(DateToFormat,
    vbGeneralDate)
        Case Else
          If Left(FormatMask(I), 1) = "\" Then _
            ResultArray(I) = Mid(FormatMask(I), 1) _
          Else
            ResultArray(I) = FormatMask(I)
      End Select
    Next
    Result = Join(ResultArray, "")
  End If
  CCFormatDate = Result
End Function
'End CCFormatDate

'CCFormatBoolean @0-635596FB
Function CCFormatBoolean(BooleanValue, arrFormat)
  Dim Result, TrueValue, FalseValue, EmptyValue

  If IsEmpty(arrFormat) Then
    Result = CStr(BooleanValue)
  Else
    TrueValue = arrFormat(0)
    FalseValue = arrFormat(1)
    EmptyValue = arrFormat(2)
    If IsEmpty(BooleanValue) Then
      Result = EmptyValue
    Else
      If BooleanValue Then _
        Result = TrueValue _
      Else _
        Result = FalseValue
    End If
  End If
  CCFormatBoolean = Result
End Function
'End CCFormatBoolean

'CCFormatNumber @0-5007EFEB
Function CCFormatNumber(NumberToFormat, FormatArray)
  Dim IsNegative
  Dim IsExtendedFormat, IsDecimalSeparator, DecimalSeparator,
    IsPeriodSeparator, PeriodSeparator

  If IsEmpty(NumberToFormat) Then
    CCFormatNumber = ""
    Exit Function
  End If

  If IsArray(FormatArray) Then
    IsExtendedFormat = FormatArray(0)
    IsNegative = (NumberToFormat < 0)
    NumberToFormat = ABS(NumberToFormat) * FormatArray(7)

    If IsExtendedFormat Then ' Extended format
      IsDecimalSeparator = FormatArray(1)
      DecimalSeparator = FormatArray(2)
      IsPeriodSeparator = FormatArray(3)
      PeriodSeparator = FormatArray(4)

      Dim BeforeDecimal, AfterDecimal
```

```
  Dim ObligatoryBeforeDecimal, DigitsBeforeDecimal,                    ↙
ObligatoryAfterDecimal, DigitsAfterDecimal
  Dim I, Z
  BeforeDecimal = FormatArray(5)
  AfterDecimal = FormatArray(6)
  If IsArray(BeforeDecimal) Then
    For I = 0 To UBound(BeforeDecimal)
      If BeforeDecimal(I) = "0" Then
        ObligatoryBeforeDecimal = ObligatoryBeforeDecimal + 1
        DigitsBeforeDecimal = DigitsBeforeDecimal + 1
      ElseIf BeforeDecimal(I) = "#" Then
        DigitsBeforeDecimal = DigitsBeforeDecimal + 1
      End If
    Next
  End If
  If IsArray(AfterDecimal) Then
    For I = 0 To UBound(AfterDecimal)
      If AfterDecimal(I) = "0" Then
        ObligatoryAfterDecimal = ObligatoryAfterDecimal + 1
        DigitsAfterDecimal = DigitsAfterDecimal + 1
      ElseIf AfterDecimal(I) = "#" Then
        DigitsAfterDecimal = DigitsAfterDecimal + 1
      End If
    Next
  End If

  Dim NumDigitsAfterDecimal, Result, DefaultValue
  If ObligatoryAfterDecimal = 0 And DigitsAfterDecimal = 1 Then
    NumDigitsAfterDecimal = -1
  ElseIf Not IsDecimalSeparator Then
    NumDigitsAfterDecimal = 0
  Else
    NumDigitsAfterDecimal = DigitsAfterDecimal
  End If
  NumberToFormat = FormatNumber(NumberToFormat, DigitsAfterDecimal,    ↙
False, False, False)

  Dim DefaultDecimal : DefaultDecimal = Mid(FormatNumber(10001/10, 1, ↙
True, False, True), 6, 1)
  Dim LeftPart, RightPart
  If Not InStr(CStr(NumberToFormat), DefaultDecimal) = 0 Then
    Dim NumberParts : NumberParts = Split(CStr(NumberToFormat),        ↙
DefaultDecimal)
    LeftPart = CStr(NumberParts(0))
    RightPart = CStr(NumberParts(1))
  Else
    LeftPart = CStr(NumberToFormat)
  End If

  Dim J : J = Len(LeftPart)

  If IsDecimalSeparator And DecimalSeparator = "" Then
    DefaultValue = CStr(FormatNumber(10001/10, 1, True, False, True))
    DecimalSeparator = Mid(DefaultValue, 6, 1)
  End If

  If IsPeriodSeparator And PeriodSeparator = "" Then
    DefaultValue = CStr(FormatNumber(10001/10, 1, True, False, True))
    PeriodSeparator = Mid(DefaultValue, 2, 1)
  End If

  If IsArray(BeforeDecimal) Then
    Dim RankNumber : RankNumber = 0
    For I  = UBound(BeforeDecimal) To 0 Step -1
      If BeforeDecimal(i) = "#" Or BeforeDecimal(i) = "0" Then
        If DigitsBeforeDecimal = 1 And J > 1 Then
          If Not IsPeriodSeparator Then
            Result = Left(LeftPart, j) & Result
```

```
            Else
              For z = J To 1 Step -1
                RankNumber = RankNumber + 1
                If RankNumber Mod 3 = 1 And RankNumber - 3 > 0 Then
                  Result = Mid(LeftPart, z, 1) & PeriodSeparator &      ↙
Result
                Else
                  Result = Mid(LeftPart, z, 1) & Result
                End If
              Next
            End If
          ElseIf J > 0 Then
            RankNumber = RankNumber + 1
            If RankNumber Mod 3 = 1 And RankNumber - 3 > 0 And          ↙
IsPeriodSeparator Then
              Result = Mid(LeftPart, j, 1) & PeriodSeparator & Result
            Else
              Result = Mid(LeftPart, j, 1) & Result
            End If
            J = J - 1
            ObligatoryBeforeDecimal = ObligatoryBeforeDecimal - 1
            DigitsBeforeDecimal = DigitsBeforeDecimal - 1
          Else
            If ObligatoryBeforeDecimal > 0 Then
              RankNumber = RankNumber + 1
              If RankNumber Mod 3 = 1 And RankNumber - 3 > 0 And        ↙
IsPeriodSeparator Then
                Result = "0" & PeriodSeparator & Result
              Else
                Result = "0" & Result
              End If
              ObligatoryBeforeDecimal = ObligatoryBeforeDecimal - 1
              DigitsBeforeDecimal = DigitsBeforeDecimal - 1
            End If
          End If
        Else
          BeforeDecimal(I) = Replace(BeforeDecimal(I), "##", "#")
          BeforeDecimal(I) = Replace(BeforeDecimal(I), "00", "0")
          Result = BeforeDecimal(I) & Result
        End If
      Next
    End If

    ' Left part after decimal
    Dim RightResult, IsRightResult : RightResult = "" : IsRightResult = ↙
False
    If IsArray(AfterDecimal) Then
      Dim IsZero : IsZero = True
      For I = UBound(AfterDecimal) To 0 Step -1
        If AfterDecimal(I) = "#" Or AfterDecimal(I) = "0" Then
          If DigitsAfterDecimal > ObligatoryAfterDecimal Then
            If Not Mid(RightPart, DigitsAfterDecimal, 1) = "0" Then     ↙
IsZero = False
            If Not IsZero Then
              RightResult = Mid(RightPart, DigitsAfterDecimal, 1) &     ↙
RightResult
              IsRightResult = True
            End If
            DigitsAfterDecimal = DigitsAfterDecimal - 1
          Else
            RightResult = Mid(RightPart, DigitsAfterDecimal, 1) &       ↙
RightResult
            DigitsAfterDecimal = DigitsAfterDecimal - 1
            IsRightResult = True
          End If
        Else
          AfterDecimal(I) = Replace(AfterDecimal(I), "##", "#")
          AfterDecimal(I) = Replace(AfterDecimal(I), "00", "0")
```

```
            RightResult = AfterDecimal(I) & RightResult
          End If
        Next
      End If

      If IsRightResult Then Result = Result & DecimalSeparator
      Result = Result & RightResult

      If NOT FormatArray(10) AND IsNegative Then _
         Result = "-" & Result

    Else ' Simple format
     If Not FormatArray(3) AND IsNegative Then _
       Result = "-" & FormatArray(5) & FormatNumber(NumberToFormat,
    FormatArray(1), FormatArray(2), False, FormatArray(4)) & FormatArray
    (6) _
     Else _
       Result = FormatArray(5) & FormatNumber(NumberToFormat, FormatArray
    (1), FormatArray(2), False, FormatArray(4)) & FormatArray(6)
    End If
    If Not FormatArray(8) Then Result = Server.HTMLEncode(Result)
    If Not CStr(FormatArray(9)) = "" Then _
      Result = "<FONT COLOR=""" & FormatArray(9) & """>" & Result & "</
    FONT>"
  Else
    Result = CStr(NumberToFormat)
  End If
  CCFormatNumber = Result

End Function
'End CCFormatNumber

'CCParseBoolean 60-33711A62
Function CCParseBoolean(Value, FormatMask)
  Dim Result
  Result = Empty
  If VarType(Value) = vbBoolean Then
    Result = Value
  Else
    If IsEmpty(FormatMask) Then
      Result = CBool(Value)
    Else
      If IsEmpty(Value) Then
        If CStr(FormatMask(0)) = "null" Then _
          Result = True
        If CStr(FormatMask(1)) = "null" Then _
          Result = False
      Else
        If CStr(Value) = CStr(FormatMask(0)) Then
          Result = True
        ElseIf CStr(Value) = CStr(FormatMask(1)) Then
          Result = False
        End If
      End If
    End If
  End If
  CCParseBoolean = Result
End Function
'End CCParseBoolean

'CCParseDate 80-6246E499
Function CCParseDate(ParsingDate, FormatMask)
  Dim ResultDate, ResultDateArray(8)
  Dim MaskPart, MaskLength, TokenLength
  Dim IsError
  Dim DatePosition, MaskPosition
  Dim Delimiter, BeginDelimiter
  Dim MonthNumber, MonthName, MonthArray
```

```
  Dim DatePart

  Dim IS_DATE_POS, YEAR_POS, MONTH_POS, DAY_POS, IS_TIME_POS, HOUR_POS,    ↙
    MINUTE_POS, SECOND_POS

  IS_DATE_POS = 0 : YEAR_POS = 1 : MONTH_POS = 2 : DAY_POS = 3
  IS_TIME_POS = 4 : HOUR_POS = 5 : MINUTE_POS = 6 : SECOND_POS = 7


  If IsEmpty(FormatMask) Then
    If CStr(ParsingDate) = "" Then _
      ResultDate = Empty _
    Else _
      ResultDate = CDate(ParsingDate)
  ElseIf (FormatMask(0) = "GeneralDate" Or FormatMask(0) = "LongDate" _
    Or FormatMask(0) = "ShortDate" Or FormatMask(0) = "LongTime"
    Or FormatMask(0) = "ShortTime") And Not CStr(ParsingDate) = "" Then
    ResultDate = CDate(ParsingDate)
  ElseIf CStr(ParsingDate) = "" Then
    ResultDate = Empty
  Else
    DatePosition = 1
    MaskPosition = 0
    MaskLength = UBound(FormatMask)
    IsError = False

    ' Default date
    ResultDateArray(IS_DATE_POS) = False
    ResultDateArray(IS_TIME_POS) = False
    ResultDateArray(YEAR_POS) = 0 : ResultDateArray(MONTH_POS) = 12 :      ↙
    ResultDateArray(DAY_POS) = 1
    ResultDateArray(HOUR_POS) = 0 : ResultDateArray(MINUTE_POS) = 0 :      ↙
    ResultDateArray(SECOND_POS) = 0

    While (MaskPosition <= MaskLength) AND NOT IsError
      MaskPart = FormatMask(MaskPosition)
      If CCSDateConstants.DateMasks.Exists(MaskPart) Then
        TokenLength = CCSDateConstants.DateMasks(MaskPart)
        If TokenLength > 0 Then
          DatePart = Mid(ParsingDate, DatePosition, TokenLength)
          DatePosition = DatePosition + TokenLength
        Else
          If MaskPosition < MaskLength Then
            Delimiter = FormatMask(MaskPosition + 1)
            BeginDelimiter = InStr(DatePosition, ParsingDate, Delimiter)
            If BeginDelimiter = 0 Then
              Err.Raise 4000, "ParseDate function: The number doesn't      ↙
  match the mask."
            Else
              DatePart = Mid(ParsingDate, DatePosition, BeginDelimiter -   ↙
  DatePosition)
              DatePosition = BeginDelimiter
            End If
          Else
            DatePart = Mid(ParsingDate, DatePosition)
          End If
        End If
        Select Case MaskPart
          Case "d", "dd"
            ResultDateArray(DAY_POS) = CInt(DatePart)
            ResultDateArray(IS_DATE_POS) = True
          Case "m", "mm"
            ResultDateArray(MONTH_POS) = CInt(DatePart)
            ResultDateArray(IS_DATE_POS) = True
          Case "mmm", "mmmm"
            MonthNumber = 0
            MonthName = UCase(DatePart)
            If MaskPart = "mmm" Then _
```

```
          MonthArray = CCSDateConstants.ShortMonths _
        Else _
          MonthArray = CCSDateConstants.Months
        While MonthNumber < 11 AND UCase(MonthArray(MonthNumber)) <>  ↙
MonthName
          MonthNumber = MonthNumber + 1
        Wend
        If MonthNumber = 11 Then
          If UCase(MonthArray(11)) <> MonthName Then _
            Err.Raise 4000, "ParseDate function: The number doesn't   ↙
match the mask."
        End If
        ResultDateArray(MONTH_POS) = MonthNumber + 1
        ResultDateArray(IS_DATE_POS) = True
      Case "yyyy"
        ResultDateArray(YEAR_POS) = CInt(DatePart)
        ResultDateArray(IS_DATE_POS) = True
      Case "yy"
        If CInt(DatePart) >= 50 Then ResultDateArray(YEAR_POS) = 1900 ↙
+ CInt(DatePart) _
        Else ResultDateArray(YEAR_POS) = 2000 + CInt(DatePart)
        ResultDateArray(IS_DATE_POS) = True
      Case "h", "hh"
        If CInt(DatePart) = 12 Then _
          ResultDateArray(HOUR_POS) = 0 _
        Else _
          ResultDateArray(HOUR_POS) = CInt(DatePart)
        ResultDateArray(IS_TIME_POS) = True
      Case "H", "HH"
        ResultDateArray(HOUR_POS) = CInt(DatePart)
        ResultDateArray(IS_TIME_POS) = True
      Case "n", "nn"
        ResultDateArray(MINUTE_POS) = CInt(DatePart)
        ResultDateArray(IS_TIME_POS) = True
      Case "s", "ss"
        ResultDateArray(SECOND_POS) = CInt(DatePart)
        ResultDateArray(IS_TIME_POS) = True
      Case "am/pm", "a/p", "AM/PM", "A/P"
        If Left(LCase(DatePart), 1) = "p" Then
          ResultDateArray(HOUR_POS) = ResultDateArray(HOUR_POS) + 12
        ElseIf Left(LCase(DatePart), 1) = "a" Then
          ResultDateArray(HOUR_POS) = ResultDateArray(HOUR_POS)
        End If
        ResultDateArray(IS_TIME_POS) = True
      Case "w", "q","S"
        ' Do Nothing
      Case Else
        IsError = IsError And DatePart = MaskPart
    End Select
  Else
    DatePart = Mid(ParsingDate, DatePosition, Len(FormatMask          ↙
(MaskPosition)))
    DatePosition = DatePosition + Len(FormatMask(MaskPosition))
    If FormatMask(MaskPosition) <> DatePart Then _
      IsError = True
  End If
  MaskPosition = MaskPosition + 1
Wend

If IsError Then Err.Raise 1056, "Wrong date value"

If ResultDateArray(IS_TIME_POS) AND ResultDateArray(IS_TIME_POS) Then
  ResultDate = DateSerial(ResultDateArray(YEAR_POS), ResultDateArray  ↙
(MONTH_POS), ResultDateArray(DAY_POS))
  ResultDate = DateAdd("h", ResultDateArray(HOUR_POS), ResultDate)
  ResultDate = DateAdd("n", ResultDateArray(MINUTE_POS), ResultDate)
  ResultDate = DateAdd("s", ResultDateArray(SECOND_POS), ResultDate)
  If NOT(Year(ResultDate) = ResultDateArray(YEAR_POS) _
```

```
        AND Month(ResultDate) = ResultDateArray(MONTH_POS) _
        AND Day(ResultDate) = ResultDateArray(DAY_POS) _
        AND Hour(ResultDate) = ResultDateArray(HOUR_POS) _
        AND Minute(ResultDate) = ResultDateArray(MINUTE_POS) _
        AND Second(ResultDate) = ResultDateArray(SECOND_POS)) _
      Then _
        Err.Raise 1056, "Wrong date value"
    ElseIf ResultDateArray(IS_TIME_POS) Then
      ResultDate = TimeSerial(ResultDateArray(HOUR_POS), ResultDateArray ↙
    (MINUTE_POS), ResultDateArray(SECOND_POS))
      If NOT(Hour(ResultDate) = ResultDateArray(HOUR_POS) _
        AND Minute(ResultDate) = ResultDateArray(MINUTE_POS) _
        AND Second(ResultDate) = ResultDateArray(SECOND_POS)) _
      Then _
        Err.Raise 1056, "Wrong date value"
    ElseIf ResultDateArray(IS_DATE_POS) Then
      ResultDate = DateSerial(ResultDateArray(YEAR_POS), ResultDateArray ↙
    (MONTH_POS), ResultDateArray(DAY_POS))
      If NOT(Year(ResultDate) = ResultDateArray(YEAR_POS) _
        AND Month(ResultDate) = ResultDateArray(MONTH_POS) _
        AND Day(ResultDate) = ResultDateArray(DAY_POS)) _
      Then _
        Err.Raise 1056, "Wrong date value"
    End If
  End If
  CCParseDate = ResultDate
End Function
'End CCParseDate

'CCParseNumber 00-BDE16F1E
Function CCParseNumber(NumberValue, FormatArray, DataType)
  Dim Result, NumberValueType
  NumberValueType = VarType(NumberValue)
  If NumberValueType = vbInteger OR NumberValueType = vbLong _
    OR NumberValueType = vbSingle OR NumberValueType = vbSingle _
    OR NumberValueType = vbCurrency OR NumberValueType = vbDecimal _
    OR NumberValueType = vbByte Then
    If DataType = ccsInteger Then
      Result = CLng(NumberValue)
    ElseIf DataType = ccsFloat Then
      Result = CDbl(NumberValue)
    End If
  Else
    If Not CStr(NumberValue) = "" Then
      Dim DefaultValue, DefaultDecimal
      Dim DecimalSeparator, PeriodSeparator
      DecimalSeparator = "" : PeriodSeparator = ""
      If IsArray(FormatArray) Then
        If FormatArray(0) Then
          DecimalSeparator = FormatArray(2)
          PeriodSeparator = FormatArray(4)
        End If
      End If
      If Not CStr(DecimalSeparator) = "" Then
        DefaultValue = CStr(FormatNumber(10001/10, 1, True, False, True))
        DefaultDecimal = Mid(DefaultValue, 6, 1)
        NumberValue = Replace(NumberValue, DecimalSeparator, ↙
    DefaultDecimal)
      End If
      If Not CStr(PeriodSeparator) = "" Then NumberValue = Replace ↙
    (NumberValue, PeriodSeparator, "")
      If DataType = ccsInteger Then
        Result = CLng(NumberValue)
      ElseIf DataType = ccsFloat Then
        Result = CDbl(NumberValue)
      End If
    Else
      Result = Empty
```

```
    End If
  End If
  CCParseNumber = Result
End Function
'End CCParseNumber

'CCParseInteger 60-42815927
Function CCParseInteger(NumberValue, FormatArray)
  CCParseInteger = CCParseNumber(NumberValue, FormatArray, ccsInteger)
End Function
'End CCParseInteger

'CCParseFloat 60-56667020
Function CCParseFloat(NumberValue, FormatArray)
  CCParseFloat = CCParseNumber(NumberValue, FormatArray, ccsFloat)
End Function
'End CCParseFloat

'CCValidateDate 60-3A410B19
Function CCValidateDate(ValidatingDate, FormatMask)
  Dim MaskPosition, I, Result, OneChar, IsSeparator
  Dim RegExpPattern, RegExpObject, Matches
  Dim ParsedTestDate, FormattedTestDate

  IsSeparator = False

  If ValidatingDate = "" OR IsEmpty(ValidatingDate) Then
    Result = True
  ElseIf IsEmpty(FormatMask) Then
    Result = IsDate(ValidatingDate)
  ElseIf FormatMask(0) = "GeneralDate" Or FormatMask(0) = "LongDate" _
    Or FormatMask(0) = "ShortDate" Or FormatMask(0) = "LongTime" _
    Or FormatMask(0) = "ShortTime" Then
    Result = IsDate(ValidatingDate)
  Else
    ParsedTestDate = CCParseDate(ValidatingDate, FormatMask)
    FormattedTestDate = CCFormatDate(ParsedTestDate, FormatMask)
    Result = FormattedTestDate = ValidatingDate
  End If
  CCValidateDate = Result
End Function
'End CCValidateDate

'CCValidateNumber 60-08089509
Function CCValidateNumber(NumberValue, FormatArray)
  Dim Result, NumberValueType
  NumberValueType = VarType(NumberValue)
  If NumberValueType = vbInteger OR NumberValueType = vbLong _
    OR NumberValueType = vbSingle OR NumberValueType = vbSingle _
    OR NumberValueType = vbCurrency OR NumberValueType = vbDecimal _
    OR NumberValueType = vbByte Then
      Result = True
  Else
    If Not CStr(NumberValue) = "" Then
      Dim DefaultValue, DefaultDecimal
      Dim DecimalSeparator, PeriodSeparator
      DecimalSeparator = "" : PeriodSeparator = ""
      If IsArray(FormatArray) Then
        If FormatArray(0) Then
          DecimalSeparator = FormatArray(2)
          PeriodSeparator = FormatArray(4)
        End If
      End If
      If Not CStr(DecimalSeparator) = "" Then
        DefaultValue = CStr(FormatNumber(10001/10, 1, True, False, True))
        DefaultDecimal = Mid(DefaultValue, 6, 1)
        NumberValue = Replace(NumberValue, DecimalSeparator, ↙
    DefaultDecimal)
```

```
      End If
      If Not CStr(PeriodSeparator) = "" Then NumberValue = Replace
    (NumberValue, PeriodSeparator, "")
      Result = IsNumeric(NumberValue)
    Else
      Result = True
    End If
  End If
  CCValidateNumber = Result
End Function
'End CCValidateNumber

'CCValidateBoolean 80-BECDE2060
Function CCValidateBoolean(Value, FormatMask)
  Dim Result: Result = False

  If VarType(Value) = vbBoolean Then
    Result = True
  Else
    If IsEmpty(FormatMask) Then
      On Error Resume Next
      Result = CBool(Value)
      Result = Not(Err > 0)
    Else
      If IsEmpty(Value) Or CStr(Value) = "" Then
        Result = (CStr(FormatMask(0)) = "null") Or (CStr(FormatMask(0)) =
    "Undefined") Or (CStr(FormatMask(0)) = "")
        Result = Result Or (CStr(FormatMask(1)) = "null") Or (CStr
    (FormatMask(1)) = "Undefined") Or (CStr(FormatMask(1)) = "")
        If UBound(FormatMask) = 2 Then _
          Result = Result Or (CStr(FormatMask(2)) = "null") Or (CStr
    (FormatMask(2)) = "Undefined") Or (CStr(FormatMask(2)) = "")
      Else
        Result = (CStr(Value) = CStr(FormatMask(0))) Or (CStr(Value) =
    CStr(FormatMask(1)))
        If UBound(FormatMask) = 2 Then _
          Result = Result Or (CStr(Value) = CStr(FormatMask(2)))
      End If
    End If
  End If
  CCValidateBoolean = Result
End Function
'End CCValidateBoolean

'CCAddParam 0C-7CE4BBE4
Function CCAddParam(QueryString, ParameterName, ParameterValue)
  Dim Result, ParameterValues, i, j, re

  Result = QueryString
  i = InStr(LCase(Result), LCase(ParameterName) & "=")
  If i > 0 Then j = InStr(i, LCase(Result), "&")
  While i > 0
    If j > 0 Then Result = Mid(Result, 1, i - 1) + Mid(Result, j + 1) Else
     Result = Mid(Result, 1, i - 1)
    i = InStr(LCase(Result), LCase(ParameterName) & "=")
    If i > 0 Then j = InStr(i, LCase(Result), "&")
  WEnd
  ParameterValues = Split(CStr(ParameterValue), ", ")
  If UBound(ParameterValues) > 0 Then
    For i = 0 To UBound(ParameterValues)
      Result = Result & "&" & ParameterName & "=" & Server.URLEncode
    (ParameterValues(i))
    Next
  Else
    Result = Result & "&" & ParameterName & "=" & Server.URLEncode
    (ParameterValue)
  End If
  Result = Replace(Result, "&&", "&")
```

```
  If Left(Result, 1) = "&" Then Result = Mid(Result, 2)
  CCAddParam = Result
End Function
'End CCAddParam

'CCRemoveParam 60-9BEDE43B
Function CCRemoveParam(QueryString, ParameterName)
  Dim Result
  Result = Replace(QueryString, ParameterName & "=" & Server.URLEncode
    (Request.QueryString(ParameterName)), "", 1, -1, 1)
  Result = Replace(Result, "&&", "&")
  If Left(Result, 1) = "&" Then Result = Mid(Result, 2)
  CCRemoveParam = Result
End Function
'End CCRemoveParam

'CCRegExpTest 60-9EAA5A2D
Function CCRegExpTest(TestValue, RegExpMask, IgnoreCase, GlobalTest)
  Dim Result
  If Not CStr(TestValue) = "" Then
    Dim RegExpObject
    Set RegExpObject = New RegExp
    RegExpObject.Pattern = RegExpMask
    RegExpObject.IgnoreCase = IgnoreCase
    RegExpObject.Global = GlobalTest
    Result = RegExpObject.Test(CStr(TestValue))
    Set RegExpObject = Nothing
  Else
    Result = True
  End If
  CCRegExpTest = Result
End Function


'End CCRegExpTest

'CCRegExpTest 60-4BE3AE1D
Sub CheckSSL()
  If Not UCase(Request.ServerVariables("HTTPS")) = "ON" Then
    Response.Write "SSL connection error. This page can be accessed only
    via secured connection."
    Response.End
  End If
End Sub

'End CCRegExpTest

'CCGetUserLogin 60-4306ED6C
Function CCGetUserLogin()
    CCGetUserLogin = Session("UserLogin")
End Function
'End CCGetUserLogin

'CCSecurityRedirect 60-06793B32
Sub CCSecurityRedirect(GroupsAccess, URL)
    Dim ErrorType
    Dim RetLink
    Dim RetLinkParams
    Dim Link
    ErrorType = CCSecurityAccessCheck(GroupsAccess)
    If NOT (ErrorType = "success") Then
        If IsEmpty(URL) Then _
            Link = "ADMIN_Login.asp" _
        Else _
            Link = URL
        RetLink = Request.ServerVariables("SCRIPT_NAME")
        RetLinkParams = CCRemoveParam(Request.ServerVariables(
    "QUERY_STRING"), "ccsForm")
```

```
        If NOT (RetLinkParams = "") Then _
            RetLink = RetLink & "?" & RetLinkParams
        Response.Redirect(Link & "?ret_link=" & _
            Server.URLEncode(RetLink) & "&type=" & ErrorType)
    End If
End Sub
'End CCSecurityRedirect

'CCGetUserID @0-449B3B19
Function CCGetUserID()
    CCGetUserID = Session("UserID")
End Function
'End CCGetUserID

'CCSecurityAccessCheck @0-DA9F7300
Function CCSecurityAccessCheck(GroupsAccess)
    Dim ErrorType
    Dim GroupID
    ErrorType = "success"
    If IsEmpty(CCGetUserID()) Then
        ErrorType = "notLogged"
    End If
    CCSecurityAccessCheck = ErrorType
End Function
'End CCSecurityAccessCheck

'CCGetGroupID @0-B1650479
Function CCGetGroupID()
CCGetGroupID = Session("GroupID")
End Function
'End CCGetGroupID

'CCUserInGroups @0-D6CB6C2E
Function CCUserInGroups(GroupID, GroupsAccess)
Dim Result
If NOT IsEmpty(GroupsAccess) Then
Result = NOT (InStr(";" & GroupsAccess & ";", ";" & GroupID & ";") = 0)
Else
Result = True
End If
CCUserInGroups = Result
End Function
'End CCUserInGroups

'CCLoginUser @0-2416320B
Function CCLoginUser(Login, Password)
    Dim Result
    Dim SQL
    Dim RecordSet
    Dim Connection

    Set Connection = New clsDBConnection1
    Connection.Open
    SQL = "SELECT STAFFid FROM STAFF WHERE STAFFlogin='" & Replace(Login, ↙
    "'", "''") & "' AND STAFFpwd='" & Replace(Password, "'", "''") & "'"
    Set RecordSet = Connection.Execute(SQL)
    Result = NOT RecordSet.EOF
    If Result Then
        Session("UserID") = RecordSet("STAFFid")
        Session("UserLogin") = Login
    End If
    RecordSet.Close
    Set RecordSet = Nothing
    Connection.Close
    Set Connection = Nothing
    CCLoginUser = Result
End Function
'End CCLoginUser
```

```
'CCLogoutUser E0-10136548
Sub CCLogoutUser()
    Session("UserID") = Empty
    Session("UserLogin") = Empty
End Sub
'End CCLogoutUser


%>
```

```
<%
'clsApplicationCache Class (C)-BACHOBEE
Class clsApplicationCache

' Cache uses Application variables to store 3-element arrays
' (key, number of hits, object or variable)
' PREFIX - Default prefix for cache elements
' PREFIX_LENGTH - Cached elements length

Public PREFIX
Public PREFIX_LENGTH

Private Sub Class_Initialize()
  PREFIX = "Cached:"
  PREFIX_LENGTH = 7
End Sub

' Release all cached elements from the cache
Sub Clear()
  Dim Key, Keys, KeyIndex
  Dim KeysString : KeysString = ""
  For Each Key in Application.Contents
    If Left(Key, PREFIX_LENGTH) = PREFIX Then KeysString = KeysString & "/↙
    " & Key
  Next
  Keys = Split(KeysString, "/")
  Application.Unlock
  For KeyIndex = 1 To UBound(Keys)
    Application.Contents.Remove(Keys(KeyIndex))
  Next
  Application.Lock
End Sub

' Release elements with zero hits from the cache.
' (Hits are stored in the second array element of the cache record).
Sub ClearUnused()
  Dim Key, Keys, KeyIndex
  Dim KeysString : KeysString = ""
  For Each Key in Application.Contents
    If Left(Key, PREFIX_LENGTH) = PREFIX Then
      If Application(Key)(2) = 0 Then KeysString = KeysString & "/" & Key
    End If
  Next
  Keys = Split(KeysString, "/")
  Application.Unlock
  For KeyIndex = 1 To UBound(Keys)
    Application.Contents.Remove(Keys(KeyIndex))
  Next
  Application.Lock
End Sub

' Reset all cache hit counters
Sub ResetCounters()
  Dim Key
  Dim Elements
  Application.Unlock
  For Each Key in Application.Contents
    If Left(Key, PREFIX_LENGTH) = PREFIX Then
      Elements = Application(Key)
      Elements(2) = 0
      Application(Key) = Elements
    End If
  Next
  Application.Lock
End Sub

' Retrieve the hit count for an object or variable
Function GetCounter(Key)
```

```
  Dim Elements
  Dim Result : Result = Empty
  Elements = Application(PREFIX & Key)
  If IsArray(Elements) Then Result = Elements(2)
  GetCounter = Result
End Function

' Check if a non-empty object exists within the cache
Function CacheItemExists(Key)
  Dim Elements
  Dim Result : Result = false
  Elements = Application(PREFIX & Key)
  If IsArray(Elements) Then
    Result = True
  End If
  CacheItemExists = Result
End Function

' Create a new Cache member
Sub Put(Key, CachingObject)
  Dim Elements(3)
  Elements(2) = 0
  If IsObject(CachingObject) Then
    Set Elements(3) = CachingObject
  Else
    Elements(3) = CachingObject
  End If
  Application.Unlock
  Application(PREFIX & Key) = Elements
  Application.Lock
End Sub

' Retrieve cached object or variable
Function GetCachedElement(Key)
  Dim Elements
  Elements = Application(PREFIX & Key)
  If IsArray(Elements) Then
    If IsObject(Elements(3)) Then
      Set GetCachedElement = Elements(3)
    Else
      GetCachedElement = Elements(3)
    End If
  Else
    GetCachedElement = Empty
  End If
End Function

End Class
'End clsApplicationCache Class

'FileSystem Object 80-416ECFB0
Class clsCache_FileSystem
  Private FileSystem
  Private FSCache

  Private Sub Class_Initialize()
    Set FileSystem = Server.CreateObject("Scripting.FileSystemObject")
    Set FSCache = Server.CreateObject("Scripting.Dictionary")
  End Sub

  Function ItemExists(name)
    ItemExists = FileSystem.FileExists(name)
  End Function

  Public Default Property Get Items(name)
    Dim res
    If FSCache.Exists(name) Then
      Set res = FSCache(name)
```

```
    Else
      Set res = New clsCacheItem_File
      res.Name = name
      Set res.FSO = FileSystem
      Set FSCache(Name) = res
    End If
    Set Items = res
  End Property

  Private Sub Class_Terminate()
    Set FileSystem = Nothing
  End Sub

  Property Get CacheType()
    CacheType = ccsCacheHTML
  End Property
End Class
'End FileSystem Object

'File Object 60-9B8773D1
Class clsCachcItem_File
  Private mContent
  Private IsOpen
  Private mFSO
  Private mName

  Private Sub Class_Initialize()
    IsOpen = False
    Set mFSO = Nothing
  End Sub

  Private Sub Class_Terminate()
    Set mFSO = Nothing
  End Sub

  Public Default Property Get Content()
    If NOT IsOpen Then
      mContent = GetFileContent(mFSO, mName)
      IsOpen = True
    End If
    Content = mContent
  End Property

  Property Get Name()
    Name = mName
  End Property

  Property Let Name(newName)
    If NOT IsOpen Then _
      mName = newName
  End Property

  Property Set FSO(newFSO)
    Set mFSO = newFSO
  End Property

  Sub Reset()
    IsOpen = False
  End Sub

  Private Function GetFileContent(FileSystem, FilePath)
    Dim Strm

    If NOT (FileSystem Is Nothing OR FilePath = "") Then
      If FileSystem.FileExists(FilePath) Then
        Set Strm = FileSystem.OpenTextFile(FilePath, ccsOpenForReading)
        If Strm.AtEndOfStream Then _
          GetFileContent = "" _
```

```
        Else _
          GetFileContent = Strm.ReadAll
        Set Strm = Nothing
      Else
        Err.raise 1052, "File object", "File object: GetFileContent failed↙
    . Type missmatch."
      End If
    Else
      Err.raise 1051, "File object", "File object: GetFileContent failed. ↙
    File " & FilePath & " is not found."
    End If
  End Function

End Class
'End File Object


%>
```

```
<%

'Template class 60-8F337D9E7

Const tpParse = False
Const tpParseSafe = True

Const ccsCacheHTML  = 1
Const ccsCacheArray = 2 ' Reserved
Const ccsCacheXML   = 3 ' Reserved

Const ccsOpenForReading = 1

Class clsTemplate

Public Tree         ' Array
Public TreePaths    ' Array
Private TreePath     ' String
Private TreeSize     ' Integer
Private mCache       ' Object

Private LONG_MAX_VALUE

Private BEGIN_OPEN
Private BEGIN_CLOSE
Private BEGIN_OPEN_LENGTH
Private BEGIN_CLOSE_LENGTH

Private END_OPEN
Private END_CLOSE
Private END_OPEN_LENGTH
Private END_CLOSE_LENGTH

Private BLOCK_PATH
Private BLOCK_TYPE
Private BLOCK_VALUE

Private TYPE_VARIABLE
Private TYPE_BEGIN_BLOCK
Private TYPE_END_BLOCK
Private TYPE_TEXT

Private Sub Class_Initialize()
  Tree = Array(0)
  TreePath = ""
  Set mCache = Nothing

  LONG_MAX_VALUE = 2147483647

  BEGIN_OPEN = "<!-- BEGIN "
  BEGIN_CLOSE = " -->"
  BEGIN_OPEN_LENGTH = 11
  BEGIN_CLOSE_LENGTH = 4

  END_OPEN = "<!-- END "
  END_CLOSE = " -->"
  END_OPEN_LENGTH = 8
  END_CLOSE_LENGTH = 4

  BLOCK_PATH = 0
  BLOCK_TYPE = 1
  BLOCK_VALUE = 2

  TYPE_VARIABLE = 0
  TYPE_BEGIN_BLOCK = 1
  TYPE_END_BLOCK = 2
  TYPE_TEXT = 3
```

```
End Sub

Property Set Cache(newCache)
  Set mCache = newCache
End Property

Sub LoadTemplate(FilePath)
  Dim TemplateArrays

  If NOT (mCache Is Nothing) Then
    If mCache.ItemExists(FilePath) Then
      Select Case mCache.CacheType
        Case ccsCacheHTML
          TemplateArrays = BuildArraysFromHTML(mCache.Items(FilePath).
    Content)
      End Select
      Tree = TemplateArrays(0) : TreePaths = TemplateArrays(1)
    Else
      Err.raise 1050, "Template engine", "Template engine: LoadTemplate
    failed. File " & FilePath & " not found."
    End If
  Else
    Err.raise 1050, "Template engine", "Template engine: LoadTemplate
    failed. Template repository is not set."
  End If
End Sub

Function BuildArraysFromHTML(FileContent)
  Dim TreePaths(), Tree()

  Dim RootName : RootName = "main"

  Dim CurrentPosition : CurrentPosition = 1
' Brackets - two-dimensional array contains tag and block positions.
' 0 - Curly bracket {.
' 1 - Begin block <!-- BEGIN block -->
' 2 - End block <!-- END block -->
' Array elements:
' 0 - name, 1 - beginning tag bracket/block, 2 - ending tag bracket/block,
    3 - number of template elements
  Const BLOCK_NAME = 0
  Const BLOCK_BEGIN = 1
  Const BLOCK_END = 2
  Const BLOCK_AMOUNT = 3

  Dim Brackets(3)
  Dim Counters(3), I
  Dim CurrentNode : CurrentNode = 0
  Dim Values(3)
  Dim BlockType


'  Brackets(0) = GetRegExp("\{[^:;\{\: \;<]*\.", FileContent: 12/22/2001
  Brackets(0) = GetRegExp("\{([A-z]+\d*( [A-z]+\d*)*)+\}", FileContent)
  Brackets(1) = GetRegExp(BEGIN_OPEN & "[^-]*" & BEGIN_CLOSE, FileContent)
  Brackets(2) = GetRegExp(END_OPEN & "[^-]*" & END_CLOSE, FileContent)

  For I = 0 to 2
    If Brackets(I)(BLOCK_AMOUNT) > 0 Then
      Values(I) = Brackets(I)(BLOCK_BEGIN)(0)
    Else
      Values(I) = LONG_MAX_VALUE
    End If
    Counters(I) = 0
  Next

  Dim Paths(30), PathLength : PathLength = 0 : Paths(0) = RootName
```

```
  CurrentNode = CurrentNode + 1
  ReDim Preserve Tree(CurrentNode)
  Tree(CurrentNode - 1) = Array("/" & RootName & "/*", TYPE_BEGIN_BLOCK,
    "")

  BlockType = MinValue(Values)
  While Not (BlockType = LONG_MAX_VALUE)

    If (Brackets(BlockType)(BLOCK_BEGIN)(Counters(BlockType)) -
    CurrentPosition + 1) > 0 Then
      CurrentNode = CurrentNode + 1
      ReDim Preserve Tree(CurrentNode)
      Tree(CurrentNode - 1) = Array(GetPath(Paths, "@text"), TYPE_TEXT,
    Mid(FileContent, CurrentPosition, Brackets(BlockType)(BLOCK_BEGIN)
    (Counters(BlockType)) - CurrentPosition + 1))
      CurrentPosition = Brackets(BlockType)(BLOCK_END)(Counters
    (BlockType)) + 1
    else
      CurrentPosition = Brackets(BlockType)(BLOCK_END)(Counters
    (BlockType)) + 1
    End If

    CurrentNode = CurrentNode + 1
    ReDim Preserve Tree(CurrentNode)

    Select Case BlockType
      Case TYPE_VARIABLE
        Tree(CurrentNode - 1) = Array(GetPath(Paths, GetName(Brackets
    (BlockType)(BLOCK_NAME)(Counters(BlockType)), BlockType)) & "/*",
    BlockType, "")
      Case TYPE_BEGIN_BLOCK
        PathLength = PathLength + 1
        Paths(PathLength) = GetName(Brackets(BlockType)(BLOCK_NAME)
    (Counters(BlockType)), BlockType)
        Tree(CurrentNode - 1) = Array(GetPath(Paths, Empty) & "*",
    BlockType, "")
      Case TYPE_END_BLOCK
        Tree(CurrentNode - 1) = Array(GetPath(Paths, Empty) & "*",
    BlockType, "")
        Paths(PathLength) = ""
        PathLength = PathLength - 1
    End Select

    Counters(BlockType) = Counters(BlockType) + 1
    If Counters(BlockType) = Brackets(BlockType)(BLOCK_AMOUNT) Then
      Values(BlockType) = LONG_MAX_VALUE
    Else
      Values(BlockType) = Brackets(BlockType)(BLOCK_BEGIN)(Counters
    (BlockType))
    end if

    BlockType = MinValue(Values)
  Wend

  If CurrentPosition < Len(FileContent) Then
    CurrentNode = CurrentNode + 1
    ReDim Preserve Tree(CurrentNode)
    Tree(CurrentNode - 1) = Array("/" & RootName & "/@text", TYPE_TEXT,
    Right(FileContent, Len(FileContent) - CurrentPosition + 1))
  End If

  CurrentNode = CurrentNode + 1
  ReDim Preserve Tree(CurrentNode)
  Tree(CurrentNode - 1) = Array("/" & RootName & "/*", TYPE_END_BLOCK, "")

  TreeSize = UBound(Tree) - 1
  ReDim TreePaths(TreeSize + 1)
  For I = 0 To TreeSize
```

```
    TreePaths(I) = Tree(I)(BLOCK_PATH) & ":" & I
  Next
  BuildArraysFromHTML = Array(Tree, TreePaths)
End Function

Sub PrintBlocks()
  Dim I
  Response.Write "<table border>"
  For I = 0 To TreeSize
    Response.Write "<tr>"
    Response.Write "<td>" & Tree(I)(BLOCK_PATH) & "</td>"
    Response.Write "<td>" & Tree(I)(BLOCK_TYPE) & "</td>"
    Response.Write "<td>" & Server.HTMLEncode(Tree(I)(BLOCK_VALUE)) & "</ ↙
    td>"
    Response.Write "<td>" & TreePaths(I) & "</td>"
    Response.Write "</tr>"
  Next
  Response.Write "</table>"
End Sub

Sub PrintArray()
  Dim I, J
  Response.Write "<table border=""1""">"
  For I = 0 To TreeSize
    Response.Write "<tr>"
    For J = 0 To UBound(Tree(I))
      Response.Write "<td>" & Server.HTMLEncode(Tree(I)(J)) & "</td>"
    Next
    Response.Write "</tr>"
  Next
  Response.Write "</table>"
End Sub

  Sub SetPath(NewPath)
    TreePath = NewPath
  End Sub

  Property Let Path(NewPath)
    TreePath = NewPath
  End Property

  Property Get Path()
    Path = TreePath
  End Property


Sub SetVar(VariableName, Value)
  Dim VariablePaths, I, VariablePosition, VariablePath
  VariablePaths = Filter(TreePaths, "/" & TreePath & VariableName & "/*")
  For I = 0 To UBound(VariablePaths)
    VariablePath = VariablePaths(i)
    VariablePosition = CLng(Mid(VariablePath, InStr(VariablePath, ":") + ↙
    1))
    If Tree(VariablePosition)(BLOCK_TYPE) = TYPE_VARIABLE or Tree ↙
    (VariablePosition)(BLOCK_TYPE) = TYPE_BEGIN_BLOCK Then Tree ↙
    (VariablePosition)(BLOCK_VALUE) = Value
  Next
End Sub

Function GetVar(VariableName)
  Dim VariablePaths, I, VariablePosition, VariablePath, Result, ↙
    TotalVariables
  VariablePaths = Filter(TreePaths, "/" & TreePath & VariableName & "/*")
  TotalVariables = UBound(VariablePaths)
  If TotalVariables > 0 Then
    If (Tree(VariablePosition)(BLOCK_TYPE) = TYPE_VARIABLE AND ↙
    TotalVariables > 0) OR _
      (Tree(VariablePosition)(BLOCK_TYPE) = TYPE_BEGIN_BLOCK AND ↙
```

```
    TotalVariables > 1) Then
      ' ERROR
    Else
      VariablePath = VariablePaths(0)
      VariablePosition = CLng(Mid(VariablePath, InStr(VariablePath, ":") +
     1))
      Result = Tree(VariablePosition)(BLOCK_VALUE)
    End If
  Else
    Result = ""
  End If
  GetVar = Result
End Function

Function GetHTML(BlockName)
  Dim BlockPaths, I, BlockPath
  Dim BeginBlockIndex, EndBlockIndex, TargetIndex
  Dim BlockContent, TreeDeep
  TreeDeep = 0
  BlockPaths = Filter(TreePaths, "/" & TreePath & BlockName & "/*")

  If ((UBound(BlockPaths) - LBound(BlockPaths)) <> 1) Then
    Err.Raise 1050, "Template Engine.", "Parsing function: Block """ &
    BlockName & """ cannot be found or selected."
  Else
    BlockPath = BlockPaths(0)
    BeginBlockIndex = CLng(Mid(BlockPath, InStr(BlockPath, ":") + 1))
  End If

  GetHTML = Tree(BeginBlockIndex)(BLOCK_VALUE)
End Function


Sub ParseAndPrint(BlockName, Accumulate, Output, TargetBlock, SafeParse)
  Dim BlockPaths, I, BlockPath
  Dim BeginBlockIndex, EndBlockIndex, TargetIndex
  Dim BlockContent, TreeDeep
  TreeDeep = 0
  BlockPaths = Filter(TreePaths, "/" & TreePath & BlockName & "/*")

  If ((UBound(BlockPaths) - LBound(BlockPaths)) <> 1) Then
    If NOT SafeParse Then _
      Err.Raise 1050, "Template Engine.", "Parsing function: Block """ &
    BlockName & """ cannot be found or selected."
  Else
    BlockPath = BlockPaths(0)
    BeginBlockIndex = CLng(Mid(BlockPath, InStr(BlockPath, ":") + 1))
    BlockPath = BlockPaths(1)
    EndBlockIndex = CLng(Mid(BlockPath, InStr(BlockPath, ":") + 1))

    For i = BeginBlockIndex + 1 To EndBlockIndex - 1
      If Tree(i)(BLOCK_TYPE) = TYPE_BEGIN_BLOCK Then
        TreeDeep = TreeDeep + 1
        If TreeDeep = 1 Then BlockContent = BlockContent & Tree(i)
    (BLOCK_VALUE)
      ElseIf Tree(i)(BLOCK_TYPE) = TYPE_END_BLOCK Then
        TreeDeep = TreeDeep - 1
      ElseIf TreeDeep = 0 Then
        BlockContent = BlockContent & Tree(i)(BLOCK_VALUE)
      End If
    Next

    If IsEmpty(TargetBlock) Then
      TargetIndex = BeginBlockIndex
    Else
      BlockPaths = Filter(TreePaths, "/" & TreePath & TargetBlock & "/*")
      BlockPath = BlockPaths(0)
      TargetIndex = CLng(Mid(BlockPath, InStr(BlockPath, ":") + 1))
```

```
    End If

    If Accumulate Then
      Tree(TargetIndex)(BLOCK_VALUE) = Tree(TargetIndex)(BLOCK_VALUE) &
    BlockContent
    Else
      Tree(TargetIndex)(BLOCK_VALUE) = BlockContent
    End If

    If Output Then Response.Write Tree(BeginBlockIndex)(BLOCK_VALUE)
  End If
End Sub

Sub ParseBlockByIndex(BeginBlockIndex, EndBlockIndex, Accumulate)
  Dim i, BlockContent, TreeDeep
  TreeDeep = 0
  For i = BeginBlockIndex + 1 To EndBlockIndex - 1
    If Tree(i)(BLOCK_TYPE) = TYPE_BEGIN_BLOCK Then
      TreeDeep = TreeDeep + 1
      If TreeDeep = 1 Then BlockContent = BlockContent & Tree(i)
    (BLOCK_VALUE)
    ElseIf Tree(i)(BLOCK_TYPE) = TYPE_END_BLOCK Then
      TreeDeep = TreeDeep - 1
    ElseIf TreeDeep = 0 Then
      BlockContent = BlockContent & Tree(i)(BLOCK_VALUE)
    End If
  Next

  If Accumulate Then
    Tree(BeginBlockIndex)(BLOCK_VALUE) = Tree(BeginBlockIndex)
    (BLOCK_VALUE) & BlockContent
  Else
    Tree(BeginBlockIndex)(BLOCK_VALUE) = BlockContent
  End If
End Sub

Function BlockExists(BlockName, BlockType)
  Dim BlockPaths, Result : Result = False
  BlockPaths = Filter(TreePaths, "/" & TreePath & BlockName & "/*")

  If BlockType = "block" Then
    If (UBound(BlockPaths) - LBound(BlockPaths)) = 1 Then _
      Result = True
  ElseIf BlockType = "variable" Then
    If (UBound(BlockPaths) - LBound(BlockPaths)) = 0 Then _
      Result = True
  Else
    Err.Raise 1050, "Template library", "BlockExists function: Invalid
    BlockType parameter."
  End If
  BlockExists = Result
End Function

Sub HideBlock(BlockName)
  SetVar BlockName, ""
End Sub

Sub Parse(BlockName, Accumulate)
  ParseAndPrint BlockName, Accumulate, False, Empty, tpParse
End Sub

Sub PParse(BlockName, Accumulate)
  ParseAndPrint BlockName, Accumulate, True, Empty, tpParse
End Sub

Sub ParseTo(BlockName, Accumulate, TargetBlock)
  ParseAndPrint BlockName, Accumulate, False, TargetBlock, tpParse
End Sub
```

```
Sub ParseSafe(BlockName, Accumulate)
  ParseAndPrint BlockName, Accumulate, False, Empty, tpParseSafe
End Sub

Sub ParseSafeTo(BlockName, Accumulate, TargetBlock)
  ParseAndPrint BlockName, Accumulate, False, TargetBlock, tpParseSafe
End Sub

Function GetPath(Paths, PathAdding)
  Dim Path : Path = Join(Paths, "/")
  GetPath = "/" & Left(Path, InStr(Path, "//")) & PathAdding
End Function

Function GetFormattedTree()
  Dim Result : Result = ""
  Dim I
  For I = 0 To UBound(Tree)
    If IsArray(Tree(I)) Then
      Result = Result & Tree(I)(BLOCK_PATH) & " "
    Else
      Result = Result & Tree(I) & " "
    End If
  Next
End Function

Function GetName(BlockString, BlockType)
  Select Case BlockType
    Case TYPE_VARIABLE:
      GetName = "@" & mid(BlockString, 2, len(BlockString) - 2)
    Case TYPE_BEGIN_BLOCK:
      GetName = mid(BlockString, BEGIN_OPEN_LENGTH + 1, len(BlockString) -↙
     BEGIN_CLOSE_LENGTH - BEGIN_OPEN_LENGTH)
    Case TYPE_END_BLOCK:
      GetName = mid(BlockString, END_OPEN_LENGTH + 1, len(BlockString) -  ↙
    END_CLOSE_LENGTH - END_OPEN_LENGTH)
  End Select
End Function

Function SetBlockHTML(Block, HTML)
  Tree(Block.BeginOfBlock)(BLOCK_VALUE) = HTML
End Function

Function MinValue(Values)

  Dim MinimumType, MinimumValue, I
  MinimumType = LONG_MAX_VALUE
  MinimumValue = LONG_MAX_VALUE

  For I = 0 To 2
    If Values(I) < MinimumValue Then
      MinimumValue = Values(I)
      MinimumType = I
    End If
  Next

  MinValue = MinimumType
End Function

Function GetRegExp(RegExpPattern, FileContent)
  Dim MatchesValues(), MatchesBeginIndexes(), MatchesEndIndexes()
  Dim RegExpObject, Matches, TotalMatches, I

  Set RegExpObject = New RegExp
  RegExpObject.Pattern = RegExpPattern
  RegExpObject.IgnoreCase = True
  RegExpObject.Global = True
  Set Matches = RegExpObject.Execute(FileContent)
```

```
  Set RegExpObject = Nothing
  If Matches.Count > 0 Then
    TotalMatches = Matches.Count
    ReDim MatchesValues(TotalMatches)
    ReDim MatchesBeginIndexes(TotalMatches)
    ReDim MatchesEndIndexes(TotalMatches)
  Else
    TotalMatches = 0
  End If
  For I = 0 To TotalMatches - 1
    MatchesValues(I) = Matches.Item(I).Value
    MatchesBeginIndexes(I) = Matches.Item(I).FirstIndex
    MatchesEndIndexes(I) = Matches.Item(I).FirstIndex + Len(Matches.Item ↙
    (I).Value)
  Next
  GetRegExp = Array(MatchesValues, MatchesBeginIndexes, MatchesEndIndexes,↙
     TotalMatches)
End Function

Function Block(Path)
  Dim BlockPaths, BeginBlockIndex, EndBlockIndex, BlockPath

  BlockPaths = Filter(TreePaths, "/" & Path & "/*")
  If ((UBound(BlockPaths) - LBound(BlockPaths)) <> 1) Then
    Set Block = Nothing
  Else
    BlockPath = BlockPaths(0)
    BeginBlockIndex = CLng(Mid(BlockPath, InStr(BlockPath, ":") + 1))
    BlockPath = BlockPaths(1)
    EndBlockIndex = CLng(Mid(BlockPath, InStr(BlockPath, ":") + 1))

    Dim TemplateBlock
    Set TemplateBlock = New clsTemplateBlock
    Set TemplateBlock.Template = Me
    With TemplateBlock
      .Path = Path
      .BeginOfBlock = BeginBlockIndex
      .EndOfBlock = EndBlockIndex
    End With
    Set Block = TemplateBlock
  End If
End Function

End Class

Class clsTemplateBlock
  Public Template
  Public Path
  Public BeginOfBlock
  Public EndOfBlock

  Sub SetVar(VariableName, Value)
    Template.SetVar Path & "/" & VariableName, Value
  End Sub

  Sub Clear()
    Template.SetVar Path, ""
  End Sub

  Property Let HTML(NewValue)
    Template.SetBlockHTML Me, NewValue
  End Property

  Function GetVar(VariableName, Value)
    GetVar = Template.GetVar(Path & "/" & VariableName)
  End Function

  Sub Show()
```

```
    Template.ParseBlockByIndex BeginOfBlock, EndOfBlock, False
  End Sub

  Sub Parse(Accumulate)
    Template.ParseBlockByIndex BeginOfBlock, EndOfBlock, Accumulate
  End Sub

  Sub PParse(Accumulate)
    Template.PParse Path, Accumulate
  End Sub

  Sub ParseTo(Accumulate, TargetBlock)
    Template.ParseTo Path, Accumulate, TargetBlock.Path
  End Sub

  Function BlockExists(BlockName, BlockType)
    BlockExists = Template.BlockExists(Path & "/" & BlockName, BlockType)
  End Function

  Function Block(NewPath)
    Set Block = Template.Block(Path & "/" & NewPath)
  End Function

  Property Get Variable(VarName)
    Variable = Template.GetVar(Path & "/@" & VarName)
  End Property

  Property Let Variable(VarName, NewValue)
    Template.SetVar Path & "/@" & VarName, NewValue
  End Property

  Property Let Visible(Value)
    If Value Then
      Template.Parse Path, False
    Else
      Template.SetVar Path, ""
    End If
  End Property

End Class

'End Template class


%>
```

```
<%

'Sorter Class 10-E34B5B65

Function CCCreateSorter(SorterName, Parent, FileName)
  Dim Sorter
  Set Sorter = New clsSorter
  With Sorter
    .ComponentName = SorterName
    .FileName = FileName
    Set .Parent = Parent
  End With
  Set CCCreateSorter = Sorter
End Function

Class clsSorter

  Public ComponentName, CCSEvents

  Dim OrderDirection
  Dim TargetName
  Dim FileName
  Dim Visible

  Private mParent
  Private CCSEventResult

  Private Sub Class_Initialize()
    Visible = True
    Set mParent = Nothing
    Set CCSEvents = CreateObject("Scripting.Dictionary")
  End Sub

  Private Sub Class_Terminate()
    Set mParent = Nothing
  End Sub

  Property Set Parent(newParent)
    Set mParent = newParent
  End Property

  Sub Show(Template)
    Dim IsOn, IsAsc
    Dim QueryString, SorterBlock
    Dim AscOnExist, AscOffExist, DescOnExist, DescOffExist
    Dim AscOn, AscOff, DescOn, DescOff

    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)

    If NOT Visible Then _
      Exit Sub

    TargetName = mParent.ComponentName
    IsOn = (mParent.ActiveSorter = ComponentName)
    IsAsc = (isEmpty(mParent.SortingDirection) OR mParent.SortingDirection↙
     = "ASC")

    Set SorterBlock = Template.Block("Sorter " & ComponentName)

    AscOnExist = SorterBlock.BlockExists("Asc_On", "block")
    If AscOnExist Then _
      Set AscOn = SorterBlock.Block("Asc_On")

    AscOffExist = SorterBlock.BlockExists("Asc_Off", "block")
    If AscOffExist Then _
      Set AscOff = SorterBlock.Block("Asc_Off")

    DescOnExist = SorterBlock.BlockExists("Desc_On", "block")
```

```
    If DescOnExist Then _
      Set DescOn = SorterBlock.Block("Desc_On")

    DescOffExist = SorterBlock.BlockExists("Desc_Off", "block")
    If DescOffExist Then _
      Set DescOff = SorterBlock.Block("Desc_Off")

    QueryString = CCGetQueryString("QueryString", Array(TargetName & "Page↙
    ", "ccsForm"))
    QueryString = CCAddParam(QueryString, TargetName & "Order",          ↙
    ComponentName)

    If IsOn then
      If IsAsc then
        OrderDirection = "DESC"
        If AscOnExist Then AscOn.Visible = True
        If AscOffExist Then AscOff.Visible = False
        If DescOnExist Then DescOn.Visible = False
        If DescOffExist Then
          DescOff.Variable("Desc_URL") = FileName & "?" & CCAddParam      ↙
    (QueryString, TargetName & "Dir", OrderDirection)
          DescOff.Visible = True
        End If
      Else
        OrderDirection = "ASC"
        If AscOnExist Then AscOn.Visible = False
        If AscOffExist Then
          AscOff.Variable("Asc_URL") = FileName & "?" & CCAddParam        ↙
    (QueryString, TargetName & "Dir", OrderDirection)
          AscOff.Visible = True
        End If
        If DescOnExist Then DescOn.Visible = True
        If DescOffExist Then DescOff.Visible = False
      End if
    Else
      OrderDirection = "ASC"
      If AscOnExist Then AscOn.Visible = False
      If AscOffExist Then
        AscOff.Variable("Asc_URL") = FileName & "?" & CCAddParam          ↙
    (QueryString, TargetName & "Dir", "ASC")
        AscOff.Visible = True
      End If
      If DescOnExist Then DescOn.Visible = False
      If DescOffExist Then
        DescOff.Variable("Desc_URL") = FileName & "?" & CCAddParam        ↙
    (QueryString, TargetName & "Dir", "DESC")
        DescOff.Visible = True
      End If
    End If

    QueryString = CCAddParam(QueryString, TargetName & "Dir",             ↙
    OrderDirection)
    SorterBlock.Variable("Sort_URL") = FileName & "?" & QueryString
    SorterBlock.Visible = True
  End Sub

End Class
'End Sorter Class


%>
```

```
<%

'Navigator Class 60-4D211748
Const tpSimple   = 00001
Const tpCentered = 00002
Const tpMoving   = 00003

Function CCCreateNavigator(Target, Name, FileName, NumberPages,
    NavigatorType)
  Dim Navigator
  Set Navigator = New clsNavigator
  Navigator.Init Target, Name, FileName, NumberPages, NavigatorType
  Set CCCreateNavigator = Navigator
End Function

Class clsNavigator
  Public ComponentName, CCSEvents
  Public Visible

  Public QueryString
  Public NavigatorBlock
  Public FirstOn, FirstOff, PrevOn, PrevOff, NextOn, NextOff, LastOn,
    LastOff
  Public Pages, PageOn, PageOff

  Dim TargetName
  Dim DataSource
  Dim PageNumber
  Dim FileName
  Dim NumberPages
  Dim NavigatorType

  Private CCSEventResult

  Private Sub Class_Initialize()
    Set CCSEvents = CreateObject("Scripting.Dictionary")
    Visible = True
  End Sub

  Sub Show(Template)
    Dim LastPage
    Dim BeginPage, EndPage, J

    QueryString = CCGetQueryString("QueryString", Array(TargetName & "Page
    ", "ccsForm"))
    Set NavigatorBlock = Template.Block("Navigator " & ComponentName)

    With NavigatorBlock
      Set FirstOn = .Block("First_On")
      Set FirstOff = .Block("First_Off")
      Set PrevOn = .Block("Prev_On")
      Set PrevOff = .Block("Prev_Off")
      Set NextOn = .Block("Next_On")
      Set NextOff = .Block("Next_Off")
      Set LastOn = .Block("Last_On")
      Set LastOff = .Block("Last_Off")
      Set Pages = .Block("Pages")
    End With

    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
    If Not Visible Then Exit Sub

    LastPage = DataSource.PageCount
    If LastPage = 0 Then
      If Not DataSource.RecordSet.Eof Then LastPage = PageNumber + 1
      If LastPage = 0 Then LastPage = 1
    End If
```

```
' Parse First and Prev Blocks
If PageNumber <= 1 Then
  If NOT FirstOff IS Nothing Then FirstOff.Visible = True
  If NOT PrevOff IS Nothing Then PrevOff.Visible = True
Else
  If NOT FirstOn IS Nothing Then
    With FirstOn
      .Variable("First_URL") = FileName & "?" & CCAddParam(QueryString↙
, TargetName & "Page", "1")
      .Visible = True
    End With
  End If
  If NOT PrevOn IS Nothing Then
    With PrevOn
      .Variable("Prev_URL") = FileName & "?" & CCAddParam(QueryString,↙
 TargetName & "Page", PageNumber - 1)
      .Visible = True
    End With
  End If
End If

If NavigatorType = tpSimple Then
  ' Set Page Number
  Set Pages = NavigatorBlock.Block("Pages")
  If Not Pages Is Nothing Then
    Set PageOff = Pages.Block("Page_Off")
            If Not PageOff Is Nothing Then
                PageOff.Variable("Page_Number") = PageNumber
                PageOff.ParseTo True, Pages
            End If
        Else
    NavigatorBlock.Variable("Page_Number") = PageNumber
  End If
ElseIf NavigatorType = tpCentered OR NavigatorType = tpMoving Then
  Set Pages = NavigatorBlock.Block("Pages")

  If NOT Pages IS Nothing Then
    Set PageOn = Pages.Block("Page_On")
    Set PageOff = Pages.Block("Page_Off")

    If NOT (PageOn IS Nothing OR PageOff IS Nothing) Then

      Select Case NavigatorType

        Case tpCentered
          BeginPage = PageNumber - (NumberPages - 1) \ 2
          If BeginPage < 1 Then BeginPage = 1
          EndPage = BeginPage + NumberPages - 1
          If EndPage > LastPage Then
            BeginPage = BeginPage - EndPage + LastPage
            If BeginPage < 1 Then BeginPage = 1
            EndPage = LastPage
          End If
          For J = BeginPage To EndPage
            If CLng(J) = CLng(PageNumber) Then
              With PageOff
                .Variable("Page_Number") = J
                .ParseTo True, Pages
              End With
            Else
              With PageOn
                .Variable("Page_Number") = J
                .Variable("Page_URL") = FileName & "?" & CCAddParam   ↙
(QueryString, TargetName & "Page", J)
                .ParseTo True, Pages
              End With
            End If
          Next
```

```
        Case tpMoving
          Dim GroupNumber, GroupFloat
          GroupFloat = PageNumber / NumberPages
          GroupNumber = Int(GroupFloat)
          If GroupFloat > GroupNumber Then GroupNumber = GroupNumber +↙
 1
          BeginPage = 1 + NumberPages * (GroupNumber - 1)
          EndPage = NumberPages * GroupNumber
          If BeginPage < 1 Then BeginPage = 1
          If EndPage > LastPage Then EndPage = LastPage
          If BeginPage > 1 Then
            With PageOn
              .Variable("Page_Number") = "&lt;" & (BeginPage - 1)
              .Variable("Page_URL") = FileName & "?" & CCAddParam      ↙
(QueryString, TargetName & "Page", BeginPage - 1)
              .ParseTo True, Pages
            End With
          End If
          For J = BeginPage To EndPage
            If CLng(J) = CLng(PageNumber) Then
              With PageOff
                .Variable("Page_Number") = J
                .ParseTo True, Pages
              End With
            Else
              With PageOn
                .Variable("Page_Number") = J
                .Variable("Page_URL") = FileName & "?" & CCAddParam    ↙
(QueryString, TargetName & "Page", J)
                .ParseTo True, Pages
              End With
            End If
          Next
          If EndPage < LastPage Then
            With PageOn
              .Variable("Page_Number") = (EndPage + 1) & "&gt;"
              .Variable("Page_URL") = FileName & "?" & CCAddParam      ↙
(QueryString, TargetName & "Page", EndPage + 1)
              .ParseTo True, Pages
            End With
          End If
      End Select
    End If
        Else
    NavigatorBlock.Variable("Page_Number") = PageNumber
  End If
End If

' Set Total Pages
NavigatorBlock.Variable("Total_Pages") = LastPage

' Parse Last and Next blocks
If CLng(PageNumber) >= CLng(LastPage) Then
  If NOT NextOff IS Nothing Then NextOff.Visible = True
  If NOT LastOff IS Nothing Then LastOff.Visible = True
Else
  If NOT NextOn IS Nothing Then
    With NextOn
      .Variable("Next_URL") = FileName & "?" & CCAddParam(QueryString,↙
 TargetName & "Page", PageNumber + 1)
      .Visible = True
    End With
  End If
  If NOT LastOn IS Nothing Then
    With LastOn
      .Variable("Last_URL") = FileName & "?" & CCAddParam(QueryString,↙
 TargetName & "Page", LastPage)
```

```
          .Visible = True
        End With
      End If
    End If

    NavigatorBlock.Visible = True
  End Sub

  Sub SetDataSource ( objDataSource )
    Set DataSource = objDataSource
  End Sub

  Sub Init ( Target, Name, NewFileName, NewNumberPages, NewNavigatorType )
    TargetName = Target
    ComponentName = Name
    FileName      = NewFileName
    NumberPages   = NewNumberPages
    NavigatorType = NewNavigatorType
    PageNumber = CLng(CCGetParam(TargetName & "Page", 1))
  End Sub

  Private Sub Class_Terminate()
    Set DataSource = Nothing
  End Sub

End Class
'End Navigator Class


%>
```

```
<%
'---------------------------------------------------------------------
' Microsoft ADO
'
' Copyright (c) 1996-1998 Microsoft Corporation.
'
'
'
' ADO constants include file for VBScript
'
'---------------------------------------------------------------------

'---- CursorTypeEnum Values ----
Const adOpenForwardOnly = 0
Const adOpenKeyset = 1
Const adOpenDynamic = 2
Const adOpenStatic = 3

'---- CursorOptionEnum Values ----
Const adHoldRecords = &H00000100
Const adMovePrevious = &H00000200
Const adAddNew = &H01000400
Const adDelete = &H01000800
Const adUpdate = &H01008000
Const adBookmark = &H00002000
Const adApproxPosition = &H00004000
Const adUpdateBatch = &H00010000
Const adResync = &H00020000
Const adNotify = &H00040000
Const adFind = &H00080000
Const adSeek = &H00400000
Const adIndex = &H00800000

'---- LockTypeEnum Values ----
Const adLockReadOnly = 1
Const adLockPessimistic = 2
Const adLockOptimistic = 3
Const adLockBatchOptimistic = 4

'---- ExecuteOptionEnum Values ----
Const adAsyncExecute = &H00000010
Const adAsyncFetch = &H00000020
Const adAsyncFetchNonBlocking = &H00000040
Const adExecuteNoRecords = &H00000080
Const adExecuteStream = &H00000400

'---- ConnectOptionEnum Values ----
Const adAsyncConnect = &H00000010

'---- ObjectStateEnum Values ----
Const adStateClosed = &H00000000
Const adStateOpen = &H00000001
Const adStateConnecting = &H00000002
Const adStateExecuting = &H00000004
Const adStateFetching = &H00000008

'---- CursorLocationEnum Values ----
Const adUseServer = 2
Const adUseClient = 3

'---- DataTypeEnum Values ----
Const adEmpty = 0
Const adTinyInt = 16
Const adSmallInt = 2
Const adInteger = 3
Const adBigInt = 20
Const adUnsignedTinyInt = 17
Const adUnsignedSmallInt = 18
```

```
Const adUnsignedInt = 19
Const adUnsignedBigInt = 21
Const adSingle = 4
Const adDouble = 5
Const adCurrency = 6
Const adDecimal = 14
Const adNumeric = 131
Const adBoolean = 11
Const adError = 10
Const adUserDefined = 132
Const adVariant = 12
Const adIDispatch = 9
Const adIUnknown = 13
Const adGUID = 72
Const adDate = 7
Const adDBDate = 133
Const adDBTime = 134
Const adDBTimeStamp = 135
Const adBSTR = 8
Const adChar = 129
Const adVarChar = 200
Const adLongVarChar = 201
Const adWChar = 130
Const adVarWChar = 202
Const adLongVarWChar = 203
Const adBinary = 128
Const adVarBinary = 204
Const adLongVarBinary = 205
Const adChapter = 136
Const adFileTime = 64
Const adPropVariant = 138
Const adVarNumeric = 139
Const adArray = &H2000

'---- FieldAttributeEnum Values ----
Const adFldMayDefer = &H00000002
Const adFldUpdatable = &H00000004
Const adFldUnknownUpdatable = &H00000008
Const adFldFixed = &H00000010
Const adFldIsNullable = &H00000020
Const adFldMayBeNull = &H00000040
Const adFldLong = &H00000080
Const adFldRowID = &H00000100
Const adFldRowVersion = &H00000200
Const adFldCacheDeferred = &H00001000
Const adFldIsChapter = &H00002000
Const adFldNegativeScale = &H00004000
Const adFldKeyColumn = &H00008000
Const adFldIsRowURL = &H00010000
Const adFldIsDefaultStream = &H00020000
Const adFldIsCollection = &H00040000

'---- EditModeEnum Values ----
Const adEditNone = &H0000
Const adEditInProgress = &H0001
Const adEditAdd = &H0002
Const adEditDelete = &H0004

'---- RecordStatusEnum Values ----
Const adRecOK = &H0000000
Const adRecNew = &H0000001
Const adRecModified = &H0000002
Const adRecDeleted = &H0000004
Const adRecUnmodified = &H0000008
Const adRecInvalid = &H0000010
Const adRecMultipleChanges = &H0000040
Const adRecPendingChanges = &H0000080
Const adRecCanceled = &H0000100
```

```
Const adRecCantRelease = &H0000400
Const adRecConcurrencyViolation = &H0000800
Const adRecIntegrityViolation = &H0001000
Const adRecMaxChangesExceeded = &H0002000
Const adRecObjectOpen = &H0004000
Const adRecOutOfMemory = &H0008000
Const adRecPermissionDenied = &H0010000
Const adRecSchemaViolation = &H0020000
Const adRecDBDeleted = &H0040000

'---- GetRowsOptionEnum Values ----
Const adGetRowsRest = -1

'---- PositionEnum Values ----
Const adPosUnknown = -1
Const adPosBOF = -2
Const adPosEOF = -3

'---- BookmarkEnum Values ----
Const adBookmarkCurrent = 0
Const adBookmarkFirst = 1
Const adBookmarkLast = 2

'---- MarshalOptionsEnum Values ----
Const adMarshalAll = 0
Const adMarshalModifiedOnly = 1

'---- AffectEnum Values ----
Const adAffectCurrent = 1
Const adAffectGroup = 2
Const adAffectAllChapters = 4

'---- ResyncEnum Values ----
Const adResyncUnderlyingValues = 1
Const adResyncAllValues = 2

'---- CompareEnum Values ----
Const adCompareLessThan = 0
Const adCompareEqual = 1
Const adCompareGreaterThan = 2
Const adCompareNotEqual = 3
Const adCompareNotComparable = 4

'---- FilterGroupEnum Values ----
Const adFilterNone = 0
Const adFilterPendingRecords = 1
Const adFilterAffectedRecords = 2
Const adFilterFetchedRecords = 3
Const adFilterConflictingRecords = 5

'---- SearchDirectionEnum Values ----
Const adSearchForward = 1
Const adSearchBackward = -1

'---- PersistFormatEnum Values ----
Const adPersistADTG = 0
Const adPersistXML = 1

'---- StringFormatEnum Values ----
Const adClipString = 2

'---- ConnectPromptEnum Values ----
Const adPromptAlways = 1
Const adPromptComplete = 2
Const adPromptCompleteRequired = 3
Const adPromptNever = 4

'---- ConnectModeEnum Values ----
```

```
Const adModeUnknown = 0
Const adModeRead = 1
Const adModeWrite = 2
Const adModeReadWrite = 3
Const adModeShareDenyRead = 4
Const adModeShareDenyWrite = 8
Const adModeShareExclusive = &Hc
Const adModeShareDenyNone = &H10
Const adModeRecursive = &H400000

'---- RecordCreateOptionsEnum Values ----
Const adCreateCollection = &H00002000
Const adCreateStructDoc = &H80000000
Const adCreateNonCollection = &H00000000
Const adOpenIfExists = &H02000000
Const adCreateOverwrite = &H04000000
Const adFailIfNotExists = -1

'---- RecordOpenOptionsEnum Values ----
Const adOpenRecordUnspecified = -1
Const adOpenOutput = &H00800000
Const adOpenAsync = &H00001000
Const adDelayFetchStream = &H00004000
Const adDelayFetchFields = &H00008000
Const adOpenExecuteCommand = &H00010000

'---- IsolationLevelEnum Values ----
Const adXactUnspecified = &Hffffffff
Const adXactChaos = &H00000010
Const adXactReadUncommitted = &H00000100
Const adXactBrowse = &H00000100
Const adXactCursorStability = &H00001000
Const adXactReadCommitted = &H00001000
Const adXactRepeatableRead = &H00010000
Const adXactSerializable = &H00100000
Const adXactIsolated = &H00100000

'---- XactAttributeEnum Values ----
Const adXactCommitRetaining = &H00020000
Const adXactAbortRetaining = &H00040000

'---- PropertyAttributesEnum Values ----
Const adPropNotSupported = &H0000
Const adPropRequired = &H0001
Const adPropOptional = &H0002
Const adPropRead = &H0200
Const adPropWrite = &H0400

'---- ErrorValueEnum Values ----
Const adErrProviderFailed = &Hbb8
Const adErrInvalidArgument = &Hbb9
Const adErrOpeningFile = &Hbba
Const adErrReadFile = &Hbbb
Const adErrWriteFile = &Hbbc
Const adErrNoCurrentRecord = &Hbcd
Const adErrIllegalOperation = &Hc93
Const adErrCantChangeProvider = &Hc94
Const adErrInTransaction = &Hcae
Const adErrFeatureNotAvailable = &Hcb3
Const adErrItemNotFound = &Hcc1
Const adErrObjectInCollection = &Hd27
Const adErrObjectNotSet = &Hd5c
Const adErrDataConversion = &Hd5d
Const adErrObjectClosed = &He78
Const adErrObjectOpen = &He79
Const adErrProviderNotFound = &He7a
Const adErrBoundToCommand = &He7b
Const adErrInvalidParamInfo = &He7c
```

```
Const adErrInvalidConnection = &He7d
Const adErrNotReentrant = &He7e
Const adErrStillExecuting = &He7f
Const adErrOperationCancelled = &He80
Const adErrStillConnecting = &He81
Const adErrInvalidTransaction = &He82
Const adErrUnsafeOperation = &He84
Const adwrnSecurityDialog = &He85
Const adwrnSecurityDialogHeader = &He86
Const adErrIntegrityViolation = &He87
Const adErrPermissionDenied = &He88
Const adErrDataOverflow = &He89
Const adErrSchemaViolation = &He8a
Const adErrSignMismatch = &He8b
Const adErrCantConvertvalue = &He8c
Const adErrCantCreate = &He8d
Const adErrColumnNotOnThisRow = &He8e
Const adErrURLIntegrViolSetColumns = &He8f
Const adErrURLDoesNotExist = &He8f
Const adErrTreePermissionDenied = &He90
Const adErrInvalidURL = &He91
Const adErrResourceLocked = &He92
Const adErrResourceExists = &He93
Const adErrCannotComplete = &He94
Const adErrVolumeNotFound = &He95
Const adErrOutOfSpace = &He96
Const adErrResourceOutOfScope = &He97
Const adErrUnavailable = &He98
Const adErrURLNamedRowDoesNotExist = &He99
Const adErrDelResOutOfScope = &He9a
Const adErrPropInvalidColumn = &He9b
Const adErrPropInvalidOption = &He9c
Const adErrPropInvalidValue = &He9d
Const adErrPropConflicting = &He9e
Const adErrPropNotAllSettable = &He9f
Const adErrPropNotSet = &Hea0
Const adErrPropNotSettable = &Hea1
Const adErrPropNotSupported = &Hea2
Const adErrCatalogNotSet = &Hea3
Const adErrCantChangeConnection = &Hea4
Const adErrFieldsUpdateFailed = &Hea5
Const adErrDenyNotSupported = &Hea6
Const adErrDenyTypeNotSupported = &Hea7

'---- ParameterAttributesEnum Values ----
Const adParamSigned = &H0010
Const adParamNullable = &H0040
Const adParamLong = &H0080

'---- ParameterDirectionEnum Values ----
Const adParamUnknown = &H0000
Const adParamInput = &H0001
Const adParamOutput = &H0002
Const adParamInputOutput = &H0003
Const adParamReturnValue = &H0004

'---- CommandTypeEnum Values ----
Const adCmdUnknown = &H0008
Const adCmdText = &H0001
Const adCmdTable = &H0002
Const adCmdStoredProc = &H0004
Const adCmdFile = &H0100
Const adCmdTableDirect = &H0200

'---- EventStatusEnum Values ----
Const adStatusOK = &H0000001
Const adStatusErrorsOccurred = &H0000002
Const adStatusCantDeny = &H0000003
```

```
Const adStatusCancel = &H0000004
Const adStatusUnwantedEvent = &H0000005

'---- EventReasonEnum Values ----
Const adRsnAddNew = 1
Const adRsnDelete = 2
Const adRsnUpdate = 3
Const adRsnUndoUpdate = 4
Const adRsnUndoAddNew = 5
Const adRsnUndoDelete = 6
Const adRsnRequery = 7
Const adRsnResynch = 8
Const adRsnClose = 9
Const adRsnMove = 10
Const adRsnFirstChange = 11
Const adRsnMoveFirst = 12
Const adRsnMoveNext = 13
Const adRsnMovePrevious = 14
Const adRsnMoveLast = 15

'---- SchemaEnum Values ----
Const adSchemaProviderSpecific = -1
Const adSchemaAsserts = 0
Const adSchemaCatalogs = 1
Const adSchemaCharacterSets = 2
Const adSchemaCollations = 3
Const adSchemaColumns = 4
Const adSchemaCheckConstraints = 5
Const adSchemaConstraintColumnUsage = 6
Const adSchemaConstraintTableUsage = 7
Const adSchemaKeyColumnUsage = 8
Const adSchemaReferentialConstraints = 9
Const adSchemaTableConstraints = 10
Const adSchemaColumnsDomainUsage = 11
Const adSchemaIndexes = 12
Const adSchemaColumnPrivileges = 13
Const adSchemaTablePrivileges = 14
Const adSchemaUsagePrivileges = 15
Const adSchemaProcedures = 16
Const adSchemaSchemata = 17
Const adSchemaSQLLanguages = 18
Const adSchemaStatistics = 19
Const adSchemaTables = 20
Const adSchemaTranslations = 21
Const adSchemaProviderTypes = 22
Const adSchemaViews = 23
Const adSchemaViewColumnUsage = 24
Const adSchemaViewTableUsage = 25
Const adSchemaProcedureParameters = 26
Const adSchemaForeignKeys = 27
Const adSchemaPrimaryKeys = 28
Const adSchemaProcedureColumns = 29
Const adSchemaDBInfoKeywords = 30
Const adSchemaDBInfoLiterals = 31
Const adSchemaCubes = 32
Const adSchemaDimensions = 33
Const adSchemaHierarchies = 34
Const adSchemaLevels = 35
Const adSchemaMeasures = 36
Const adSchemaProperties = 37
Const adSchemaMembers = 38
Const adSchemaTrustees = 39
Const adSchemaFunctions = 40
Const adSchemaActions = 41
Const adSchemaCommands = 42
Const adSchemaSets = 43

'---- FieldStatusEnum Values ----
```

```
Const adFieldOK = 0
Const adFieldCantConvertValue = 2
Const adFieldIsNull = 3
Const adFieldTruncated = 4
Const adFieldSignMismatch = 5
Const adFieldDataOverflow = 6
Const adFieldCantCreate = 7
Const adFieldUnavailable = 8
Const adFieldPermissionDenied = 9
Const adFieldIntegrityViolation = 10
Const adFieldSchemaViolation = 11
Const adFieldBadStatus = 12
Const adFieldDefault = 13
Const adFieldIgnore = 15
Const adFieldDoesNotExist = 16
Const adFieldInvalidURL = 17
Const adFieldResourceLocked = 18
Const adFieldResourceExists = 19
Const adFieldCannotComplete = 20
Const adFieldVolumeNotFound = 21
Const adFieldOutOfSpace = 22
Const adFieldCannotDeleteSource = 23
Const adFieldReadOnly = 24
Const adFieldResourceOutOfScope = 25
Const adFieldAlreadyExists = 26
Const adFieldPendingInsert = &H10000
Const adFieldPendingDelete = &H20000
Const adFieldPendingChange = &H40000
Const adFieldPendingUnknown = &H80000
Const adFieldPendingUnknownDelete = &H100000

'---- SeekEnum Values ----
Const adSeekFirstEQ = &H1
Const adSeekLastEQ = &H2
Const adSeekAfterEQ = &H4
Const adSeekAfter = &H8
Const adSeekBeforeEQ = &H10
Const adSeekBefore = &H20

'---- ADUPROP_UPDATECRITERIA_ENUM Values ----
Const adCriteriaKey = 0
Const adCriteriaAllCols = 1
Const adCriteriaUpdCols = 2
Const adCriteriaTimeStamp = 3

'---- ADUPROP_ASYNCTHREADPRIORITY_ENUM Values ----
Const adPriorityLowest = 1
Const adPriorityBelowNormal = 2
Const adPriorityNormal = 3
Const adPriorityAboveNormal = 4
Const adPriorityHighest = 5

'---- ADCPROP_AUTORECALC_ENUM Values ----
Const adRecalcUpFront = 0
Const adRecalcAlways = 1

'---- ADCPROP_UPDATERESYNC_ENUM Values ----

'---- ADCPROP_UPDATERESYNC_ENUM Values ----

'---- MoveRecordOptionsEnum Values ----
Const adMoveUnspecified = -1
Const adMoveOverWrite = 1
Const adMoveDontUpdateLinks = 2
Const adMoveAllowEmulation = 4

'---- CopyRecordOptionsEnum Values ----
Const adCopyUnspecified = -1
```

```
Const adCopyOverWrite = 1
Const adCopyAllowEmulation = 4
Const adCopyNonRecursive = 2

'---- StreamTypeEnum Values ----
Const adTypeBinary = 1
Const adTypeText = 2

'---- LineSeparatorEnum Values ----
Const adLF = 10
Const adCR = 13
Const adCRLF = -1

'---- StreamOpenOptionsEnum Values ----
Const adOpenStreamUnspecified = -1
Const adOpenStreamAsync = 1
Const adOpenStreamFromRecord = 4

'---- StreamWriteEnum Values ----
Const adWriteChar = 0
Const adWriteLine = 1

'---- SaveOptionsEnum Values ----
Const adSaveCreateNotExist = 1
Const adSaveCreateOverWrite = 2

'---- FieldEnum Values ----
Const adDefaultStream = -1
Const adRecordURL = -2

'---- StreamReadEnum Values ----
Const adReadAll = -1
Const adReadLine = -2

'---- RecordTypeEnum Values ----
Const adSimpleRecord = 0
Const adCollectionRecord = 1
Const adStructDoc = 2
%>
```

```
<%

'File Description 00-AE10E379
'=================================================================================
'
'  This file contains the following classes:
'       Class clsSQLParameters
'       Class clsSQLParameter
'       Class clsFields
'       Class clsControls
'       Class clsControl
'       Class clsField
'       Class clsButton
'       Class clsFileUpload
'       Class clsDatePicker
'       Class clsListControl
'       Class clsErrors
'       Class clsEmptyDataSource
'       Class clsDataSource
'       Class clsCommand
'       Class clsStringBuffer
'       Class clsConverter
'
'=================================================================================
'End File Description

'Constant List 00-52D70043

' ------- Controls ---------------
Const ccsLabel              = 00001
Const ccsLink               = 00002
Const ccsTextBox            = 00003
Const ccsTextArea           = 00004
Const ccsListBox            = 00005
Const ccsRadioButton        = 00006
Const ccsButton             = 00007
Const ccsCheckBox           = 00008
Const ccsImage              = 00009
Const ccsImageLink          = 00010
Const ccsHidden             = 00011
Const ccsCheckBoxList       = 00012
Const ccsDatePicker         = 00013

Dim ccsControlTypes(13)
ccsControlTypes(ccsLabel)         = "Label"
ccsControlTypes(ccsLink)          = "Link"
ccsControlTypes(ccsTextBox)       = "TextBox"
ccsControlTypes(ccsTextArea)      = "TextArea"
ccsControlTypes(ccsListBox)       = "ListBox"
ccsControlTypes(ccsRadioButton)   = "RadioButton"
ccsControlTypes(ccsButton)        = "Button"
ccsControlTypes(ccsCheckBox)      = "CheckBox"
ccsControlTypes(ccsImage)         = "Image"
ccsControlTypes(ccsImageLink)     = "ImageLink"
ccsControlTypes(ccsHidden)        = "Hidden"
ccsControlTypes(ccsCheckBoxList)  = "CheckBoxList"
ccsControlTypes(ccsDatePicker)    = "DatePicker"

' ------- Operators ---------------
Const opEqual               = 00001
Const opNotEqual            = 00002
Const opLessThan            = 00003
Const opLessThanOrEqual     = 00004
Const opGreaterThan         = 00005
Const opGreaterThanOrEqual  = 00006
Const opBeginsWith          = 00007
Const opNotBeginsWith       = 00008
Const opEndsWith            = 00009
```

```
Const opNotEndsWith          = 00010
Const opContains             = 00011
Const opNotContains          = 00012
Const opIsNull               = 00013
Const opNotNull              = 00014

' ------- Datasource types -------
Const dsTable                = 00001
Const dsSQL                  = 00002
Const dsProcedure            = 00003
Const dsListOfValues         = 00004
Const dsEmpty                = 00005

' ------- Command types -------
Const cmdOpen                = 00001
Const cmdExec                = 00002

' ------- Parse types ------------
Const ccsParseAccumulate     = True
Const ccsParseOverwrite      = False

' ------- Listbox populating types ------------
Const ccsJoins               = 0
Const ccsStringConcats       = 1

' ------- CheckBox states --------
Const ccsChecked             = True
Const ccsUnchecked           = False
'End Constant List

'clsSQLParameters Class 80-3D6F8092

Class clsSQLParameters

  Public Connection
  Public Criterion()
  Public AssembledWhere
  Public ParameterSources
  Public Errors
  Public DataSource

  Public ParametersList

  Private Sub Class_Initialize()
    ReDim Criterion(100)
    Set ParametersList = Server.CreateObject("Scripting.Dictionary")
    Set DataSource = Nothing
  End Sub

  Private Sub Class_Terminate()
    Set ParametersList = Nothing
  End Sub

  Public Default Property Get Parameters(Name)
    Set Parameters = ParametersList(Name)
  End Property

  Public Property Set Parameters(Name, NewParameter)
    Set ParametersList(Name) = NewParameter
  End Property

  Property Get Count()
    Count = ParametersList.Count
  End Property

  Function AddParameter(ID, ParameterSource, DataType, Format, DBFormat, ↙
    DefaultValue, UseIsNull)
    Dim SQLParameter
```

```
    Set SQLParameter = New clsSQLParameter
    With SQLParameter
      Set .Connection = Connection
      .DataType = DataType
      .Format = Format
      .DBFormat = DBFormat
      .Caption = ParameterSource
      .DefaultValue = DefaultValue
      .UseIsNull = UseIsNull
      Set .DataSource = DataSource
      If IsObject(ParameterSources) Then
        If IsEmpty(DefaultValue) Or ((DataType <> ccsText And DataType <> ↙
    ccsMemo) And CStr(DefaultValue) = "") Then
          .Text = ParameterSources(ParameterSource)
          If (.DataType = ccsText Or .DataType = ccsMemo) And CStr(.Text) ↙
    <> "" Then .Text = Replace(.Text, "'", "''")
        Else
          If IsEmpty(ParameterSources(ParameterSource)) Or ((DataType <>  ↙
    ccsText And DataType <> ccsMemo) And ParameterSources(ParameterSource)↙
     = "") Then
            .Value = DefaultValue
          Else
            .Text = ParameterSources(ParameterSource)
            If (.DataType = ccsText Or .DataType = ccsMemo) And CStr(.    ↙
    Text) <> "" Then .Text = Replace(.Text, "'", "''")
          End If
        End If
      End If
    End With
    Set ParametersList(ID) = SQLParameter
    Set SQLParameter = Nothing
  End Function

  Function getParamByID(ID)
    Set getParamByID = ParametersList(ID)
  End Function

  Property Get AllParamsSet()
    Dim ParametersItems, I, Result

    Result = True
    I = 0
    ParametersItems = ParametersList.Items
    While Result AND (I <= UBound(ParametersItems))
      Result = NOT IsEmpty(ParametersItems(I).Value)
      I = I + 1
    Wend
    AllParamsSet = Result
  End Property

  Function GetError()
    Dim ParametersItems, I, Result

    ParametersItems = ParametersList.Items
    For I = 0 To UBound(ParametersItems)
      Result = Result & ParametersItems(I).Errors.ToString
    Next
    GetError = Result
  End Function

  Function opAND(Brackets, LeftPart, RightPart)
    Dim Result
    If NOT IsEmpty(LeftPart) Then
      If NOT IsEmpty(RightPart) Then
        Result = LeftPart & " and " & RightPart
        If Brackets Then Result = " (" & Result & ") "
      Else
        Result = LeftPart
```

```
      End If
    Else
      If NOT IsEmpty(RightPart) Then
        Result = RightPart
      End If
    End If
    opAND = Result
  End Function

  Function opOR(Brackets, LeftPart, RightPart)
    Dim Result
    If NOT IsEmpty(LeftPart) Then
      If NOT IsEmpty(RightPart) Then
        Result = LeftPart & " or " & RightPart
        If Brackets Then Result = " (" & Result & ") "
      Else
        Result = LeftPart
      End If
    Else
      If NOT IsEmpty(RightPart) Then
        Result = RightPart
      End If
    End If
    opOR = Result
  End Function

  Function Operation(Operator, Brackets, FieldName, Parameter)
    If CStr(Parameter.Text) <> "" Then
      Dim Result
      Dim Value, SQLValue
      Value = Parameter.SQLText
      SQLValue = Connection.ToSQL(Value, Parameter.DataType)
      Value = Replace(Value, "'", "''")

      Select Case Operator
        Case opEqual
          Result = FieldName & " = " & SQLValue
        Case opNotEqual
          Result = FieldName & " <> " & SQLValue
        Case opLessThan
          Result = FieldName & " < " & SQLValue
        Case opLessThanOrEqual
          Result = FieldName & " <= " & SQLValue
        Case opGreaterThan
          Result = FieldName & " > " & SQLValue
        Case opGreaterThanOrEqual
          Result = FieldName & " >= " & SQLValue
        Case opBeginsWith
          Result = FieldName & " like '" & Value & "%'"
        Case opNotBeginsWith
          Result = FieldName & " not like '" & Value & "%'"
        Case opEndsWith
          Result = FieldName & " like '%" & Value & "'"
        Case opNotEndsWith
          Result = FieldName & " not like '%" & Value & "'"
        Case opContains
          Result = FieldName & " like '%" & Value & "%'"
        Case opNotContains
          Result = FieldName & " not like '%" & Value & "%'"
        Case opIsNull
          Result = FieldName & " is null"
        Case opNotNull
          Result = FieldName & " is not null"
      End Select
      Operation = Result
    Else
     If Parameter.UseIsNull Then
       Select Case Operator
```

```
          Case opNotEqual, opNotBeginsWith, opNotEndsWith, opNotContains, ↙
    opNotNull
            Result = FieldName & " is not null"
          Case Else
            Result = FieldName & " is null"
        End Select
        Operation = Result
      Else
        Operation = Empty
      End If
    End If
  End Function
End Class
'End clsSQLParameters Class

'clsSQLParameter Class 00-7B4F1941
Class clsSQLParameter
  Public Errors
  Public DataType
  Public Format
  Public DBFormat
  Public Caption
  Public Connection
  Public DataSource
  Public DefaultValue
  Public UseIsNull

  Private VarValue
  Private SQLTextValue
  Private TextValue

  Private Sub Class_Initialize()
    VarValue = Empty
    SQLTextValue = Empty
    TextValue = Empty
    UseIsNull = False

    DataType = ccsText
    Set Errors = New clsErrors
  End Sub

  Private Sub Class_Terminate()
    Set Errors = Nothing
  End Sub

  Function GetParsedValue(ParsingValue, MaskFormat)
    Dim Result

    If Not IsEmpty(ParsingValue) Then
      Select Case DataType
        Case ccsDate
          If VarType(ParsingValue) = vbDate Then
            Result = ParsingValue
          ElseIf CCValidateDate(ParsingValue, MaskFormat) Then
            Result = CCParseDate(ParsingValue, MaskFormat)
          Else
            If IsArray(Format) Then
              PrintDBError "", "", "The value in field " & Caption & " is ↙
    not valid. Use the following format: " & Join(Format, "") & "."
            Else
              PrintDBError "", "", "The value in field " & Caption & " is ↙
    not valid."
            End If
          End If
        Case ccsBoolean
          Result = CCParseBoolean(ParsingValue, MaskFormat)
        Case ccsInteger
          If CCValidateNumber(ParsingValue, MaskFormat) Then
```

```
          Result = CCParseInteger(ParsingValue, MaskFormat)
        Else
          PrintDBError "", "", "The value in field " & Caption & " is
  not valid."
        End If
      Case ccsFloat
        If CCValidateNumber(ParsingValue, MaskFormat) Then
          Result = CCParseFloat(ParsingValue, MaskFormat)
        Else
          PrintDBError "", "", "The value in field " & Caption & " is
  not valid."
        End If
      Case ccsText, ccsMemo
        Result = CStr(ParsingValue)
    End Select
  End If

  GetParsedValue = Result
End Function

Function GetFormattedValue(MaskFormat)
  Dim Result, Value

  If IsEmpty(VarValue) Then
    Value = DefaultValue
  Else
    Value = VarValue
  End If

  Select Case DataType
    Case ccsDate
      Result = CCFormatDate(Value, MaskFormat)
    Case ccsBoolean
      Result = CCFormatBoolean(Value, MaskFormat)
    Case ccsInteger, ccsFloat
      Result = CCFormatNumber(Value, MaskFormat)
    Case ccsText, ccsMemo
      Result = CStr(Value)
  End Select

  GetFormattedValue = Result
End Function

Property Let Value(NewValue)
  VarValue = Empty
  SQLTextValue = Empty
  TextValue = Empty
  If NOT IsEmpty(NewValue) And Not (NewValue="") Then
    Select Case DataType
      Case ccsDate
        VarValue = CDate(NewValue)
      Case ccsBoolean
        VarValue = CBool(NewValue)
      Case ccsInteger
        VarValue = CLng(NewValue)
      Case ccsFloat
        VarValue = CDbl(NewValue)
      Case ccsText, ccsMemo
        VarValue = CStr(NewValue)
    End Select
  End If
End Property

Property Get Value()
  If IsEmpty(VarValue) Then
    Value = DefaultValue
  Else
    Value = VarValue
```

```
    End If
  End Property

  Property Let Text(NewText)
    If Not IsEmpty(NewText) Then
      SQLTextValue = Empty
      TextValue = NewText
      VarValue = GetParsedValue(TextValue, Format)
    End If
  End Property

  Property Get Text()
    If IsEmpty(TextValue) Then TextValue = GetFormattedValue(Format)
    Text = TextValue
  End Property

  Property Let SQLText(varNewSQLText)
    SQLTextValue = varNewSQLText
  End Property

  Property Get SQLText()
    If IsEmpty(SQLTextValue) Then
      SQLTextValue = GetFormattedValue(DBFormat)
    End If
    SQLText = SQLTextValue
  End Property

End Class

'End clsSQLParameter Class

'clsFields Class {0-791D3D1C
Class clsFields
  Private objFields
  Private Items
  Private Counter

  Private Sub Class_Initialize()
    Set objFields = CreateObject("Scripting.Dictionary")
  End Sub

  Sub AddFields(Fields) ' Add new objects to Object array
    Dim I
    If IsArray(Fields) Then
      For I = LBound(Fields) To UBound(Fields)
        Set objFields(Fields(I).Name) = Fields(I)
      Next
    End If
  End Sub

  Public Default Property Get Item(Name)
    Set Item = objFields(Name)
  End Property

  Sub InitEnum()
    Items = objFields.Items
    Counter = 0
  End Sub

  Function NextItem()
    Set NextItem = Items(Counter)
    Counter = Counter + 1
  End Function

  Function EndOfEnum()
    EndOfEnum = (Counter > UBound(Items))
  End Function
```

```
  Function Exists(Name)
    Exists = objFields.Exists(Name)
  End Function

End Class
'End clsFields Class

'CCCreateCollection Function @0-61899C71
Function CCCreateCollection(Block, TargetBlock, Accumulate, Controls)
  Dim Collection
  Set Collection = New clsControls
  With Collection
    Set .Block = Block
    If NOT IsNull(TargetBlock) Then
      Set .TargetBlock = TargetBlock
    End If
    .Accumulate = Accumulate
    .AddControls Controls
  End With
  Set CCCreateCollection = Collection
End Function

'End CCCreateCollection Function

'clsControls Class @0-77F5EEBC
Class clsControls
  Private Objects ' Dictionary object
  Private CCSEventResult

  Public Block
  Public Accumulate

  Private objTargetBlock
  Private isSetTargetBlock
  Private mVisible

  Private Sub Class_Initialize()
    Set Objects = Server.CreateObject("Scripting.Dictionary")
    mVisible = True
  End Sub

  Private Sub Class_Terminate()
    Set Objects = Nothing
  End Sub

  Sub AddControls(Controls) ' Add new objects to Object array
    Dim ArraySize, NumberControls, I

    If IsArray(Controls) Then
      NumberControls = UBound(Controls)
      ArraySize = Objects.Count

      For i = ArraySize To ArraySize + NumberControls
        Objects.Add i,Controls(I)
      Next
    End If
  End Sub

  Sub AddControl(Control) ' Add a new object to Object array
    If TypeName(Control) = "clsControls" Then
      Objects.Add Objects.Count, Control
    Else
      Objects.Add Control.Name, Control
    End If
  End Sub

  Property Get Items(ItemName)
    If Objects.Exists(ItemName) Then
```

```
      Set Items = Objects(ItemName)
    Else
      Set Items = Nothing
    End If
  End Property

  Property Let Items(ItemName, NewItem)
    If Objects.Exists(ItemName) Then
      Objects(ItemName) = NewItem
    Else
      Objects.Add ItemName, NewItem
    End If
  End Property

  Sub Show()
    Dim Element, Obj

    If NOT mVisible Then Exit Sub

    For Each Element In Objects
      Set Obj = Objects.Item(element)
      If TypeName(Obj) = "clsControls" Then
        Obj.Show
      Else
        Obj.Show Block
      End If
    Next

    If Not IsEmpty(Accumulate) Then
      If isSetTargetBlock Then
        Block.ParseTo Accumulate, objTargetBlock
      Else
        Block.Parse Accumulate
      End If
    End If

  End Sub

  Function Validate()
    Dim Validation, Element

    Validation = True
    For Each Element In Objects
      Objects.Item(Element).Validate
      Validation = Validation And (Objects.Item(Element).Errors.Count = 0)
    Next

    Validate = Validation
  End Function

  Function GetErrors()
    Dim Errors, Element
    For Each Element In Objects
      Errors = Errors & Objects.Item(Element).Errors.ToString
    Next
    GetErrors = Errors
  End Function

  Property Set TargetBlock(NewBlock)
    isSetTargetBlock = True
    Set objTargetBlock = NewBlock
  End Property

  Sub InitEnum()
    Items = objFields.Items
    Counter = 0
  End Sub
```

```
  Function NextItem()
    Set NextItem = Items(Counter)
    Counter = Counter + 1
  End Function

  Function EndOfEnum()
    EndOfEnum = (Counter > UBound(Items))
  End Function

  Property Let Visible(newValue)
    mVisible = CBool(newValue)
  End Property

  Property Get Visible()
    Visible = mVisible
  End Property
End Class

'End clsControls Class

'CCCreateControl Function {0-706BB4D6
Function CCCreateControl(ControlType,  Name, Caption, DataType, Format,   ↙
    InitValue)
    Dim Control
    Set Control = New clsControl
    With Control
        .ControlType = ControlType
        .Name = Name
        .BlockName = ccsControlTypes(ControlType) & " " & Name
        .ControlTypeName = ccsControlTypes(ControlType)
        .Caption = Caption
        .DataType = DataType
        .Format = Format
    If NOT IsEmpty(InitValue) Then
      If ControlType = ccsCheckBox Then
          .State = True
      Else
            .Text = InitValue
      End If
        End If
    End With
    Set CCCreateControl = Control
End Function
'End CCCreateControl Function

'clsControl Class {0-8A73557B
Class clsControl
  Public Errors
  Public DataType
  Public Format
  Public DBFormat
  Public Caption
  Public ControlType
  Public ControlTypeName
  Public Name
  Public BlockName
  Public ExternalName
  Public HTML
  Public Required
  Public CheckedValue
  Public UncheckedValue
  Public State
  Public Visible
  Public TemplateBlock

  Public Parameters

  Private mPage
```

```
  Private VarValue
  Private TextValue

  Public CCSEvents
  Private CCSEventResult

  Private Sub Class_Initialize()
    VarValue = Empty
    TextValue = Empty
    Visible = True
    ExternalName = Empty

    DataType = ccsText
    HTML = False
    Required = False
    Set Errors = New clsErrors
    Set CCSEvents = CreateObject("Scripting.Dictionary")
    Parameters = ""
  End Sub

  Private Sub Class_Terminate()
    Set Errors = Nothing
  End Sub

  Function Validate()
    If Required And CStr(VarValue) = "" And Errors.Count = 0 Then
      Errors.addError(CCSRunTimeMessages.GetMessage("RequiredField", Array↙
    ("FieldName", Caption)))
    End If
    Validate = CCRaiseEvent(CCSEvents, "OnValidate", Me)
  End Function

  Function GetParsedValue(ParsingValue, MaskFormat)
    Dim Result

    Result = CCSConverter.StringToType(DataType, ParsingValue, MaskFormat)
    If CCSConverter.ParseError Then
      If DataType = ccsDate AND (IsArray(Format) OR IsArray(CCSConverter. ↙
    DateFormat)) Then
        If IsArray(Format) Then _
          Errors.addError(CCSRunTimeMessages.GetMessage("IncorrectFormat",↙
     Array("FieldName", Caption, "Format", Join(Format, "")))) _
        Else _
          Errors.addError(CCSRunTimeMessages.GetMessage("IncorrectFormat",↙
     Array("FieldName", Caption, "Format", Join(CCSConverter.DateFormat,  ↙
    ""))))
      Else
        Errors.addError(CCSRunTimeMessages.GetMessage("IncorrectValue",   ↙
    Array("FieldName", Caption)))
      End If
    End If

    GetParsedValue = Result
  End Function

  Property Get Link()
    Dim Result

    If Parameters = "" Then
      Result = mPage
    Else
      Result = mPage & "?" & Parameters
    End If

    Link = Result
  End Property

  Property Let Link(newLink)
```

```
    Dim parsedLink
    If CStr(newLink) = "" Then
      mPage = ""
      Parameters = ""
    Else
      parsedLink = Split(newLink, "?")
      mPage = parsedLink(0)
      If UBound(parsedLink) = 1 Then
        Parameters = parsedLink(1)
      Else
        Parameters = ""
      End If
    End If
  End Property

  Property Get Page()
    Page = mPage
  End Property

  Property Let Page(newPage)
    mPage = newPage
  End Property

  Function GetFormattedValue(MaskFormat)
    GetFormattedValue = CCSConverter.TypeToString(DataType, VarValue,
    MaskFormat)
  End Function

  Sub Show(Template)
    Dim NeedShow, sTmpValue

    Set TemplateBlock = Template.Block(ControlTypeName & " " & Name)

    If TemplateBlock Is Nothing Then
      Set TemplateBlock = Template
      NeedShow = False
    Else
      NeedShow = True
      TemplateBlock.HTML = ""
    End If

    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
    If NOT Visible Then Exit Sub

    If IsEmpty(ExternalName) Then
      TemplateBlock.Variable(Name & "_Name") = Name
    Else
      TemplateBlock.Variable(Name & "_Name") = ExternalName
    End If

    If IsEmpty(TextValue) Then
      TextValue = GetFormattedValue(Format)
    End If

    Select Case ControlType
      Case ccsLabel, ccsTextBox, ccsTextArea, ccsHidden
        If HTML Then
          TemplateBlock.Variable(Name) = TextValue
        Else
          sTmpValue = Server.HTMLEncode(TextValue)
          If ControlType = ccsLabel Then
            sTmpValue = Replace(sTmpValue, vbCrLf, "<BR>")
          End If
          TemplateBlock.Variable(Name) = sTmpValue
        End If
      Case ccsImage
          sTmpValue = Server.HTMLEncode(TextValue)
          If ControlType = ccsLabel Then
```

```
            sTmpValue = Replace(sTmpValue, vbCrLf, "<BR>")
          End If
          TemplateBlock.Variable(Name) = sTmpValue
      Case ccsLink
        If HTML Then
          TemplateBlock.Variable(Name) = TextValue
        Else
          TemplateBlock.Variable(Name) = Replace(Server.HTMLEncode
    (TextValue), vbCrLf, "<BR>")
        End If
        TemplateBlock.Variable(Name & "_Src") = Me.Link
      Case ccsImageLink
        TemplateBlock.Variable(Name & "_Src") = Server.HTMLEncode
    (TextValue)
        TemplateBlock.Variable(Name) = Me.Link
      Case ccsCheckBox
        If State Then
          TemplateBlock.Variable(Name) = "CHECKED"
        Else
          TemplateBlock.Variable(Name) = ""
        End If
    End Select
    If NeedShow Then TemplateBlock.Show
    Set TemplateBlock = Nothing
  End Sub

  Property Let Value(NewValue)
    VarValue = Empty
    TextValue = Empty

    VarValue = CCSConverter.VBSConvert(DataType, NewValue)

    If ControlType = ccsCheckBox Then
      If DataType = ccsBoolean Then
        If IsEmpty(NewValue) Or (NewValue="") Then
          State = False
        Else
          State = VarValue
        End If
      Else
        if DataType = ccsDate Then
          State = (VarValue = CDate(CheckedValue))
        Else
          State = (VarValue = CheckedValue)
        End if
      End If
    End If

  End Property

  Property Get Value()
    If ControlType = ccsCheckBox Then
      If IsEmpty(State) Then
        Value = UncheckedValue
      Else
        Value = IIf(State, CheckedValue, UncheckedValue)
      End If
    Else
        Value = VarValue
    End If
  End Property

  Property Let Text(NewText)
    VarValue = Empty
    TextValue = NewText

    VarValue = GetParsedValue(TextValue, Format)
    If ControlType = ccsCheckBox Then
```

```
      State = (VarValue = CheckedValue)
    End If
  End Property

  Property Get Text()
    If IsEmpty(TextValue) Then
      TextValue = GetFormattedValue(Format)
    End If
    Text = TextValue
  End Property

End Class
'End clsControl Class

'CCCreateField Function 00-A187BD87
Function CCCreateField(Name, DBFieldName, DataType, DBFormat, DataSource)
    Dim Field
    Set Field = New clsField
    With Field
        .Name = Name
        .DBFieldName = DBFieldName
        .DataType = DataType
        .DBFormat = DBFormat
        Set .DataSource = DataSource
    End With
    Set CCCreateField = Field
End Function
'End CCCreateField Function

'clsField Class 00-F4A4ACF3
Class clsField
  Public DataType
  Public DBFormat
  Public Name
  Public DBFieldName
  Public Errors

  Private mDataSource
  Private mConverter
  Private mValue
  Private mSQLText

  Private Sub Class_Initialize()
    mValue = Empty
    mSQLText = Empty
    DataType = ccsText
    Set Errors = New clsErrors
  End Sub

  Private Sub Class_Terminate()
    Set Errors = Nothing
  End Sub

  Public Function GetParsedValue(ParsingValue, MaskFormat)
    Dim Result, ValueType
    Result = Empty

    If NOT IsEmpty(ParsingValue) Then
      ValueType = VarType(ParsingValue)
      If ValueType = vbString Then
        Result = mConverter.StringToType(DataType, ParsingValue,
    MaskFormat)
      Else
        Result = mConverter.VBSConvert(DataType, ParsingValue)
      End If
      If mConverter.ParseError Then
        Errors.addError(CCSRunTimeMessages.GetMessage("IncorrectValue",
    Array("FieldName", Name)))
```

```
      End If
    End If
    GetParsedValue = Result
  End Function

  Public Function GetFormattedValue(MaskFormat)
    GetFormattedValue = CCSConverter.TypeToString(DataType, mValue,
    MaskFormat)
  End Function

  Public Property Let Value(vData)
    mSQLText = Empty
    mValue = mConverter.VBSConvert(DataType, vData)
  End Property

  Public Default Property Get Value()
    Dim RS, Result, ResultType

    Result = mValue
    If IsObject(DataSource.Recordset) Then
      Set RS = DataSource.Recordset
      If RS.State = adStateOpen Then
        If NOT RS.EOF Then _
          Result = CCGetValue(RS, DBFieldName)
      End If
      Result = GetParsedValue(Result, DBFormat)
    End If
    Value = Result
  End Property

  Public Property Let SQLText(vData)
    mSQLText = vData
  End Property

  Public Property Get SQLText()
    If IsEmpty(mSQLText) Then
      mSQLText = GetFormattedValue(DBFormat)
    End If
    SQLText = mSQLText
  End Property

  Public Property Set DataSource(oRef)
    Set mDataSource = oRef
    If IsObject(mDataSource.Connection) Then
      Set mConverter = mDataSource.Connection.Converter
    Else
      Set mConverter = CCSConverter
    End If
  End Property

  Public Property Get DataSource()
    Set DataSource = mDataSource
  End Property

End Class
'End clsField Class

'CCCreateFileUpload Function C0-A734B792
Function CCCreateFileUpload(Name,Caption,TemporaryFolder,FileFolder,
    AllowedFileMasks,DisallowedFileMasks,FileSizeLimit,Required)
    Dim FileUpload
    Set FileUpload = New clsFileUpload

    With FileUpload
        .Name                 = Name
        .DeleteControlName    = Name & "_Delete"
        .Caption              = Caption
        .TemporaryFolder      = TemporaryFolder & "\"
```

```
        .FileFolder          = FileFolder & "\"
        .AllowedFileMasks    = AllowedFileMasks
        .DisallowedFileMasks = DisallowedFileMasks
        .FileSizeLimit       = FileSizeLimit
        .Required            = Required
    End With

    Set CCCreateFileUpload = FileUpload
End Function
'End CCCreateFileUpload Function

'clsFileUpload Class 00-DCDAA860
Class clsFileUpload
  Public Name
  Public CCSEvents
  Public Visible
  Public ExternalName
  Public Errors
  Public Caption
  Public Required
  Public TemplateBlock

  Public AllowedFileMasks
  Public DisallowedFileMasks
  Public FileSizeLimit

  Public IsUploaded
  Public FileSize
  Public fso
  Public DeleteControlName
  Public ExternalDeleteControlName

  Private VarTemporaryFolder
  Private VarFileFolder
  Private VarValue
  Private VarText
  Private StateArray

  Private CCSEventResult

  Private Sub Class_Initialize()
    Set CCSEvents = CreateObject("Scripting.Dictionary")
    Set fso       = CreateObject("Scripting.FileSystemObject")

    Set Errors    = New clsErrors

    ExternalName  = Empty
    Visible       = True
    IsUploaded    = False
    FileSize      = 0
    ReDim StateArray(1)
    StateArray(0) = Empty
    StateArray(1) = Empty
  End Sub

  Private Sub Class_Terminate()
    Set CCSEvents = Nothing
    Set Errors    = Nothing
    Set fso       = Nothing
  End Sub

  Public Function Upload(CurrentRow)
  On Error Resume Next
    Dim f, FieldName, NewFileName
    If Not IsEmpty(CurrentRow) Then
      ExternalName = Name & "_" & CStr(CurrentRow)
      ExternalDeleteControlName = Name & "_Delete_" & CStr(CurrentRow)
    End If
```

```
    If CCGetRequestParam("ccsForm",ccsGet) <> "" Then
      SetState CCGetRequestParam(IIf(Not IsEmpty(ExternalName),          ↙
    ExternalName, Name), ccsPost)
      Value = StateArray(0)
    End If

    If UploadedFilesCount > 0 Then
      Set f = objUpload.Files(IIf(Not IsEmpty(ExternalName), ExternalName,↙
     Name) & "_File")

      If Not (f is Nothing) Then
        FileSize = f.Size
        NewFileName = GetValidFileName(f.FileName) & f.FileName
        f.SaveAs VarTemporaryFolder & NewFileName
        StateArray(1) = NewFileName
        If Not IsEmpty(StateArray(0)) And StateArray(1) <> StateArray(0)  ↙
    Then DeleteFile
        Value = NewFileName
      Else
        If IsEmpty(StateArray(0)) Then
          VarValue = ""
          FileSize = 0
        End If
        StateArray(1) = Empty
      End If
    End If

    If CCGetRequestParam(IIf(Not IsEmpty(ExternalDeleteControlName),     ↙
    ExternalDeleteControlName, DeleteControlName), ccsPost) <> "" Then   ↙
    DeleteFile

    If Not Err.Number = 0 Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Response.Write "An error occured when uploading file specified in " ↙
    & FieldName & ". Error description: " & CStr(Err.Source) & ", " & CStr↙
    (Err.Description) & "."
      Response.End
    End If
  End Function

  Public Function GetFile(CurrentRow)
  On Error Resume Next
    Dim f, FieldName
    If Not IsEmpty(CurrentRow) Then
      ExternalName = Name & "_" & CStr(CurrentRow)
      ExternalDeleteControlName = Name & "_Delete_" & CStr(CurrentRow)
    End If

    If CCGetRequestParam("ccsForm",ccsGet) <> "" Then
      SetState CCGetRequestParam(IIf(Not IsEmpty(ExternalName),          ↙
    ExternalName, Name), ccsPost)
      Text = StateArray(0)
      Value = Text
    End If

    If UploadedFilesCount > 0 Then
      Set f = objUpload.Files(IIf(Not IsEmpty(ExternalName), ExternalName,↙
     Name) & "_File")

      If Not (f is Nothing) Then
        FileSize = f.Size
        Text = f.FileName
      Else
        If IsEmpty(StateArray(0)) Then
          Text = ""
          FileSize = 0
```

```
        End If
      End If
    End If

    If Not Err.Number = 0 Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Response.Write "An error occured when uploading file specified in " ↙
    & FieldName & ". Error description: " & CStr(Err.Source) & ", " & CStr↙
    (Err.Description) & "."
      Response.End
    End If
  End Function

  Public Function MoveFromTempFolder
  On Error Resume Next
    Dim FieldName
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeProcessFile", Me)
    If (fso.FileExists(VarTemporaryFolder & VarValue)) Then
      fso.MoveFile VarTemporaryFolder & VarValue, VarFileFolder & VarValue
      StateArray(0) = VarValue
      StateArray(1) = VarValue
    End If
    CCSEventResult = CCRaiseEvent(CCSEvents, "AfterProcessFile", Me)
    If Not Err.Number = 0 Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Response.Write "An error occured when uploading file specified in " ↙
    & FieldName & ". Error description: " & CStr(Err.Source) & ", " & CStr↙
    (Err.Description) & "."
      Response.End
    End If
  End Function

  Public Function DeleteFile()
  On Error Resume Next
    Dim FieldName, FileName
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeDeleteFile", Me)
    FileName = VarValue
    If IsEmpty(FileName) Then FileName = VarText
    If (fso.FileExists(VarTemporaryFolder & FileName)) Then
      fso.DeleteFile VarTemporaryFolder & FileName, True
      VarValue = ""
    End If
    If (fso.FileExists(VarFileFolder & FileName)) Then
      fso.DeleteFile VarFileFolder & FileName, True
      VarValue = ""
    End If
    CCSEventResult = CCRaiseEvent(CCSEvents, "AfterDeleteFile", Me)
    If Not Err.Number = 0 Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Response.Write "An error occured when uploading file specified in " ↙
    & FieldName & ". Error description: " & CStr(Err.Source) & ", " & CStr↙
    (Err.Description) & "."
      Response.End
    End If
  End Function

  Property Let TemporaryFolder(NewText)
    Dim FieldName, regEx
    VarTemporaryFolder = NewText
    If UCase(Left(NewText, 5)) = "%TEMP" Then
      VarTemporaryFolder = fso.GetSpecialFolder(2) & Mid(NewText, 6)
    End If
    If NewText = "" Then
      VarTemporaryFolder = Server.MapPath(".\") & "\"
    End If
```

```
    Set regEx = New RegExp
    regEx.Pattern = "^[a-z]:.*"
    regEx.IgnoreCase = True
    If Not regEx.Test(VarTemporaryFolder) Then VarTemporaryFolder = Server↙
    .MapPath(VarTemporaryFolder) & "\"

    If Not fso.FolderExists(VarTemporaryFolder) Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Response.Write "Unable to upload the file specified in " & CStr      ↙
    (FieldName) & " - temporary upload folder doesn't exist."
      Response.End
    End If
  End Property

  Property Get TemporaryFolder()
    TemporaryFolder = VarTemporaryFolder
  End Property

  Property Let FileFolder(NewText)
    Dim FieldName, regEx
    VarFileFolder = NewText
    If UCase(Left(NewText, 5)) = "%TEMP" Then
      VarFileFolder = fso.GetSpecialFolder(2) & Mid(NewText, 6)
    End If

    Set regEx = New RegExp
    regEx.Pattern = "^[a-z]:.*"
    regEx.IgnoreCase = True
    If Not regEx.Test(VarFileFolder) Then VarFileFolder = Server.MapPath  ↙
    (VarFileFolder) & "\"

    If Not fso.FolderExists(VarFileFolder) Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Response.Write "Unable to upload the file specified in " & CStr      ↙
    (FieldName) & " - upload folder doesn't exist."
      Response.End
    End If
  End Property

  Property Get FileFolder()
    FileFolder = CStr(VarFileFolder)
  End Property

  Property Let Value(NewValue)
  On Error Resume Next
    Dim f, FieldName

    If Not IsEmpty(NewValue) Then
      If Len(NewValue) > 0 Then

        If (fso.FileExists(VarTemporaryFolder & NewValue)) Then
          VarValue   = NewValue
          VarText  = VarValue
          StateArray(0) = NewValue
          StateArray(1) = Empty
          IsUploaded = True

          Set f = fso.GetFile(VarTemporaryFolder & NewValue)
          FileSize = f.Size
          VarText = f.Path

        ElseIf (fso.FileExists(VarFileFolder & NewValue)) Then

          VarValue   = NewValue
          VarText  = VarValue
```

```
          StateArray(0) = NewValue
          StateArray(1) = Empty
          IsUploaded = True

          Set f = fso.GetFile(VarFileFolder & NewValue)
          FileSize = f.Size

        End If

      End If
    End If
    If Not IsEmpty(ExternalName) And NewValue = "" Then
      VarValue = ""
      VarText = ""
      StateArray(0) = Empty
      StateArray(1) = Empty
      IsUploaded = False
      FileSize = 0
    End If

    If Not Err.Number = 0 Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Response.Write "An error occured when uploading file specified in " ↙
    & FieldName & ". Error description: " & CStr(Err.Source) & ", " & CStr↙
    (Err.Description) & "."
      Response.End
    End If
  End Property

  Property Get Value()
    Value = VarValue
  End Property

  Property Let Text(NewText)
    Dim f
    VarText = NewText
  End Property

  Property Get Text()
    Text = CStr(VarText)
  End Property

  Public Function GetValidFileName(FileName)
  On Error Resume Next
    Dim dta, tm, index, prefix, FieldName
    dta = Date()
    tm = time()
    index = 0

    Do
      prefix = Year(dta) & Month(dta) & Day(dta) & Hour(tm) & Minute(tm) &↙
     Second(tm) & CStr(index) & "."
      index = index + 1
    Loop While fso.FileExists(VarTemporaryFolder & prefix & FileName) Or  ↙
    fso.FileExists(VarFileFolder & prefix & FileName)

    GetValidFileName = prefix
    If Not Err.Number = 0 Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Response.Write "An error occured when uploading file specified in " ↙
    & FieldName & ". Error description: " & CStr(Err.Source) & ", " & CStr↙
    (Err.Description) & "."
      Response.End
    End If
  End Function
```

```
  Public Function GetOriginFileName(FileName)
    Dim nPos
    nPos = InStr(FileName,".")
    If nPos > 0 Then
      GetOriginFileName = Mid(FileName,nPos+1)
    Else
      GetOriginFileName = FileName
    End If
  End Function

  Function Validate()
    Dim FieldName
    If Required And CStr(VarText) = "" Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Errors.addError("The file attachment in field " & FieldName & " is  ↙
    required.")
    End If
    If Not CStr(Text) = "" And DisallowedFileMasks <> "" And CCRegExpTest ↙
    (VarText, Replace(Replace(Replace(DisallowedFileMasks, "?", "[^\.]"), ↙
    "*", "[^\.]*"), ";", "|"), True, True) And Errors.Count = 0 Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Errors.addError("The file type specified in field " & FieldName & " ↙
    is not allowed.")
    End If
    If Not CStr(Text) = "" And AllowedFileMasks <> "" And AllowedFileMasks↙
     <> "*" And Not CCRegExpTest(VarText, Replace(Replace(Replace        ↙
    (AllowedFileMasks, "?", "[^\.]"), "*", "[^\.]*"), ";", "|"), True,   ↙
    True) And Errors.Count = 0 Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Errors.addError("The file type specified in field " & FieldName & " ↙
    is not allowed.")
    End If
    If Not IsUploaded And FileSize > FileSizeLimit And Errors.Count = 0  ↙
    Then
      If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName = ↙
    Name
      Errors.addError("The file size in field " & FieldName & " is too   ↙
    large.")
    End If
    If Errors.Count > 0 And fso.FileExists(VarTemporaryFolder & VarText)  ↙
    Then DeleteFile
    Validate = CCRaiseEvent(CCSEvents, "OnValidate", Me)
  End Function

  Sub Show(Template)
    Dim TemplateBlock, UploadBlock, InfoBlock, DeleteControlBlock

    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)

    If Visible Then

      Set TemplateBlock      = Template.Block("FileUpload " & Name)
      Set UploadBlock        = TemplateBlock.Block("Upload")
      Set InfoBlock          = TemplateBlock.Block("Info")
      Set DeleteControlBlock = TemplateBlock.Block("DeleteControl")

      If Not (TemplateBlock Is Nothing) Then

          If IsEmpty(ExternalName) Then
            TemplateBlock.Variable("ControlName") = Name
          Else
            TemplateBlock.Variable("ControlName") = ExternalName
          End If
          TemplateBlock.Variable("State") = GetState()
```

```
        If (Not IsUploaded Or Required) And Not (UploadBlock Is Nothing)↙
    Then
          If IsEmpty(ExternalName) Then
            UploadBlock.Variable("FileControl") = Name & "_File"
          Else
            UploadBlock.Variable("FileControl") = ExternalName & "_File"
          End If
          UploadBlock.Parse ccsParseOverwrite
          InfoBlock.Visible = False
          DeleteControlBlock.Visible = False
        End If

        If IsUploaded And Not (InfoBlock Is Nothing) Then
          InfoBlock.Variable("FileName")      = GetOriginFileName        ↙
  (VarValue)
          InfoBlock.Variable("FileSize")      = FileSize
          InfoBlock.Parse ccsParseOverwrite
          UploadBlock.Visible = Required
        End If

        If IsUploaded And Not Required And Not (DeleteControlBlock Is   ↙
  Nothing) Then
          If IsEmpty(ExternalDeleteControlName) Then
            DeleteControlBlock.Variable("DeleteControl") =              ↙
  DeleteControlName
          Else
            DeleteControlBlock.Variable("DeleteControl") =              ↙
  ExternalDeleteControlName
          End If
          DeleteControlBlock.Parse ccsParseOverwrite
          UploadBlock.Visible = Required
        End If

        TemplateBlock.Parse ccsParseOverwrite
      End if

  End If
End Sub

Function OnClick()
  OnClick = CCRaiseEvent(CCSEvents, "OnClick", Me)
End Function

Private Function GenerateStateKey()
  Dim dta, tm, random_number
  dta = Date()
  tm = time()
  Randomize
  random_number = Abs(Int((2147483647 + 2147483648 + 1) * Rnd -        ↙
  2147483648))
  GenerateStateKey = CStr(random_number) & Day(dta) & Hour(tm) & Minute ↙
  (tm) & Second(tm)
End Function

Private Function GetState()
  Dim ControlStateKey
  If StateArray(0) = Empty Then StateArray(0) = Value
  ControlStateKey = GenerateStateKey()
  Session(ControlStateKey) = StateArray
  GetState = ControlStateKey
End Function

Private Function SetState(value)
  If IsArray(Session(value)) Then
    StateArray(0) = Session(value)(0)
    StateArray(1) = Session(value)(1)
  Else
    StateArray(0) = Empty
```

```
      StateArray(1) = Empty
    End If
  End Function

End Class

'End clsFileUpload Class

'CCCreateButton Function @0-E6E95E6F
Function CCCreateButton(Name)
    Dim Button
    Set Button = New clsButton
    Button.Name = Name
    Set CCCreateButton = Button
End Function
'End CCCreateButton Function

'clsButton Class @0-FE2F9E6E
Class clsButton
  Public Name
  Public CCSEvents
  Public Visible
  Public ExternalName

  Private CCSEventResult

  Private Sub Class_Initialize()
    Set CCSEvents = CreateObject("Scripting.Dictionary")
    ExternalName = Empty
    Visible = True
  End Sub

  Private Sub Class_Terminate()
    Set CCSEvents = Nothing
  End Sub

  Sub Show(Template)

    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)

    If Visible Then

      If Template.BlockExists("Button " & Name, "block") Then
        Template.Block("Button " & Name).Variable("Button_Name") = Name
        Template.Block("Button " & Name).Parse ccsParseOverwrite
      End If

    End If
  End Sub

  Function OnClick()
    OnClick = CCRaiseEvent(CCSEvents, "OnClick", Me)
  End Function
End Class

'End clsButton Class

'CCCreateDatePicker Function @0-C5ED8C84
Function CCCreateDatePicker(Name, FormName, ControlName)
  Dim DatePicker
  Set DatePicker = New clsDatePicker
  With DatePicker
    .Name = Name
    .FormName = FormName
    .ControlName = ControlName
  End With
  Set CCCreateDatePicker = DatePicker
End Function
```

```
'End CCCreateDatePicker Function

'clsDatePicker Class @0-B0B324C1
Class clsDatePicker
  Public Name
  Public ExternalName
  Public FormName
  Public ControlName
  Public ExternalControlName
  Public Visible

  Private Sub Class_Initialize()
    ExternalName = Empty
    ExternalControlName = Empty
    Visible = True
  End Sub

  Sub Show(Template)
    Dim TemplateBlock
    If Visible Then
      Set TemplateBlock = Template.Block("DatePicker " & Name)
      If Template.BlockExists("DatePicker " & Name, "block") Then
        TemplateBlock.Variable("Name") = CStr(FormName) & "_" & CStr(Name)
        TemplateBlock.Variable("FormName") = CStr(FormName)
        If IsEmpty(ExternalControlName) Then
          TemplateBlock.Variable("DateControl") = CStr(ControlName)
        Else
          TemplateBlock.Variable("DateControl") = CStr                    ↙
    (ExternalControlName)
        End If
        TemplateBlock.Parse ccsParseOverwrite
      End If
    End If
  End Sub

End Class
'End clsDatePicker Class

'CCCreateList Function @0-54B52904
Function CCCreateList(ControlType, Name, Caption, DataType, InitValue,     ↙
    DataSource)
    Dim Control
    Set Control = New clsListControl
    With Control
        .Name = Name
        .ControlType = ControlType
        .Caption = Caption
        .DataType = DataType
        .ControlTypeName = ccsControlTypes(ControlType)
        If IsArray(InitValue) Then
            .MultipleValues = InitValue
        Else
            .Text = InitValue
        End If
        If IsObject(DataSource) Then
            Set .DataSource = DataSource
        End If
    End With
    Set CCCreateList = Control
End Function
'End CCCreateList Function

'clsListControl Class @0-4BC484CD
Class clsListControl
  Private Control
  Private DataTypeValue

  Public CCSEvents
```

```
  Public DataSource
  Public Recordset
  Public Errors
  Public Name
  Public ControlType
  Public Caption
  Public Required
  Public TemplateBlock
  Public Visible
  Public HTML
  Public MultipleValues
  Public IsMultiple
  Public ExternalName
  Public ControlTypeName

  Public TextColumn
  Public BoundColumn
  Private CCSEventResult
  Private mPopulatingType

  Public IsPopulated
  Public ItemsList()
  Public KeysList()
  Public ItemsCount


  Private Sub Class_Initialize()
    Required = False
    BoundColumn = 0
    TextColumn = 1
    Visible = True
    PopulatingType = ccsStringConcats
    HTML = False
    ExternalName = Empty
    IsPopulated = False
    ItemsCount = 0

    Set Control = New clsControl
    Set CCSEvents = CreateObject("Scripting.Dictionary")
    Set Errors = New clsErrors
  End Sub

  Private Sub Class_Terminate()
    Set Control = Nothing
    Set Errors = Nothing
    Set DataSource = Nothing
    Set Recordset = Nothing
    Set CCSEvents = Nothing
  End Sub

  Public Function AddValue(NewValue)
    Dim NumberOfValues
    NumberOfValues = Ubound(MultipleValues)
    ReDim Preserve MultipleValues(NumberOfValues + 1)
    MultipleValues(NumberOfValues + 1) = NewValue
  End Function

  Public Function HasMultipleValues
    If IsArray(MultipleValues) Then
      HasMultipleValues = (Ubound(MultipleValues) > 0)
    Else
      HasMultipleValues = False
    End If
  End Function

  Property Let Value(NewValue)
    If IsMultiple Then
      If HasMultipleValues Then
```

```
        MultipleValues(1) = NewValue
      Else
        AddValue NewValue
      End If
    End If
    Control.Value = NewValue
  End Property

  Property Get Value()
    If IsMultiple And HasMultipleValues Then
      Value = MultipleValues(1)
    Else
      Value = Control.Value
    End If
  End Property

  Public Property Let PopulatingType(vType)
      mPopulatingType = vType
  End Property

  Public Property Get PopulatingType()
      PopulatingType = mPopulatingType
  End Property

  Property Let DataType(NewDataType)
    DataTypeValue = NewDataType
    Control.DataType = DataTypeValue
  End Property

  Property Get DataType()
    DataType = DataTypeValue
  End Property

  Property Get SQLValue()
    SQLValue = Control.SQLValue
  End Property

  Function Validate()
    Dim FieldName,Passed

    If Required Then
      If IsMultiple Then
        Passed = (Ubound(MultipleValues) = 0)
      Else
        Passed = (CStr(Control.Value) = "")
      End If

      If Passed Then
        If Not CStr(Caption) = "" Then FieldName = Caption Else FieldName ↙
    = Name
        Errors.addError(CCSRunTimeMessages.GetMessage("RequiredField",     ↙
    Array("FieldName", FieldName)))
      End If

    End If
    Validate = CCRaiseEvent(CCSEvents, "OnValidate", Me)
  End Function


  Public Sub RePopulate
    Dim cmdErrors, MaxBound, i
    Dim RSFields

    If NOT IsObject(DataSource) Then Exit Sub

    Set cmdErrors = new clsErrors
    Set Recordset = DataSource.Exec(cmdErrors)
```

```
    Set RSFields = new clsFields
    RSFields.AddFields Array(CCCreateField("BoundColumn", BoundColumn,
    DataTypeValue, Empty, Recordset), _
      CCCreateField("TextColumn", TextColumn, ccsText, Empty, Recordset))

    Set Recordset.FieldsCollection = RSFields

    If cmdErrors.Count > 0 Then
      Dim ErrorString
      If ControlType = ccsRadioButton Then
        ErrorString = "RadioButton " & Name
      ElseIf ControlType = ccsListBox Then
        ErrorString = "ListBox " & Name
      Else
        ErrorString = "CheckBoxList" & Name
      End If
      PrintDBError ErrorString, "", cmdErrors.ToString()
    Else
      MaxBound = 25: i = 1
      ReDim ItemsList(MaxBound)
      ReDim KeysList (MaxBound)

      While NOT Recordset.EOF

        If i >= MaxBound Then
          MaxBound = MaxBound + 25
          ReDim Preserve ItemsList(MaxBound)
          ReDim Preserve KeysList (MaxBound)
        End If

        ItemsList(i) = Recordset.Fields("TextColumn")
        KeysList(i)  = Recordset.Fields("BoundColumn")

        i = i + 1
        Recordset.MoveNext
      Wend
    End If

    Recordset.Close

    IsPopulated = True
    ItemsCount = i - 1
    Set cmdErrors = Nothing

  End Sub

  Sub Show(Template)
    Dim Result, Selected, Recordset, ResultBuffer, i, j
    Dim cmdErrors
    Dim NeedShow

    If NOT IsObject(DataSource) Then Exit Sub
    If Not IsPopulated Then RePopulate()

    Set TemplateBlock = Template.Block(ControlTypeName & " " & Name)
    NeedShow = NOT (TemplateBlock Is Nothing)
    If ControlType = ccsListBox Then
      If NOT NeedShow Then _
        Set TemplateBlock = Template
    End If

    If IsEmpty(ExternalName) Then
      TemplateBlock.Variable(Name & "_Name") = Name
    Else
      TemplateBlock.Variable(Name & "_Name") = ExternalName
    End If

    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
```

```
    If NOT Visible Then Exit Sub

    If ControlType = ccsRadioButton or ControlType = ccsCheckBoxList Then
      TemplateBlock.Clear

      For j = 1 To ItemsCount
        Selected = ""
        If IsMultiple Then
          For i = 1 To Ubound(MultipleValues)
            If UCase(CStr(KeysList(j))) = UCase(CStr(MultipleValues(i)))  ↙
    Then
              Selected = " CHECKED"
            End If
          Next
        Else
          If UCase(CStr(KeysList(j))) = UCase(CStr(Value)) Then
            Selected = " CHECKED"
          End If
        End If
        TemplateBlock.Variable("Value") = CCSConverter.TypeToString       ↙
    (DataType, KeysList(j), Empty)
        TemplateBlock.Variable("Check") = Selected
        If HTML Then
          TemplateBlock.Variable("Description") = CStr(ItemsList(j))
        Else
          TemplateBlock.Variable("Description") = Server.HTMLEncode(CStr  ↙
    (ItemsList(j)))
        End If
        TemplateBlock.Parse True
      Next

    ElseIf ControlType = ccsListBox Then

      Set ResultBuffer = new clsStringBuffer
      Result = ""

      If mPopulatingType = ccsStringConcats Then

        For j = 1 To ItemsCount
          Selected = ""

          If IsMultiple Then
            For i = 1 To Ubound(MultipleValues)
              If UCase(CStr(KeysList(j))) = UCase(CStr(MultipleValues(i)))↙
     Then
                Selected = " SELECTED"
                Exit For
              End If
            Next
          Else
            If UCase(CStr(KeysList(j))) = UCase(CStr(Value)) Then
              Selected = " SELECTED"
            End If
          End If
          Result = Result & "<OPTION VALUE="""  _
            & CCSConverter.TypeToString(DataType, KeysList(j), Empty)  _
            & """" & Selected & ">" & ItemsList(j) & "</OPTION>" &       ↙
    vbNewLine
        Next
      Else
        For j = 1 To ItemsCount
          Selected = ""
          If IsMultiple Then
            For i = 1 To Ubound(MultipleValues)
              If UCase(CStr(KeysList(j))) = UCase(CStr(MultipleValues(i)))↙
     Then
                Selected = " SELECTED"
```

```
    ReDim Errors(1)
  End Sub

  Property Get Count()
    Count = ErrorsCount
  End Property

  Property Get ErrorByNumber(ErrorNumber)
    If ErrorNumber > ErrorsCount OR ErrorNumber < 0 Then
      Err.Raise 4001, "Error class, ErrorByNumber function. Parameter out ↙
    of range."
    End If
    ErrorByNumber = Errors(ErrorNumber)
  End Property

  Property Get ToString()
    If ErrorsCount > 0 Then
      ToString = Join(Errors, ErrorDelimiter) & ErrorDelimiter
    Else
      ToString = ""
    End If
  End Property

End Class
'End clsErrors Class

'CCCreateDataSource Function {0-07E5C726
Function CCCreateDataSource(DataSourceType, Connection, CommandSource)
  Dim Cmd

  Set Cmd = New clsCommand
  If DataSourceType <> dsListOfValues Then
    Set Cmd.Connection = Connection
    Set Cmd.WhereParameters.Connection = Connection
  End If
  Cmd.CommandType = DataSourceType
  Cmd.CommandOperation = cmdOpen
  Cmd.ActivePage = -1

  Select Case DataSourceType
    Case dsTable
      Cmd.SQL = CommandSource(0)
      Cmd.Where = CommandSource(1)
      Cmd.OrderBy = CommandSource(2)
    Case dsSQL
      Cmd.SQL = CommandSource
    Case dsProcedure
      Set Cmd.SQL = CommandSource
    Case dsListOfValues
      Cmd.LOV = CommandSource
  End Select
  Set CCCreateDataSource = Cmd
End Function
'End CCCreateDataSource Function

'clsEmptyDataSource Class {0-03551674
Class clsEmptyDataSource

  Public Errors

  Private Sub Class_Initialize()
    Set CCSEvents = CreateObject("Scripting.Dictionary")
    Set Errors = New clsErrors
  End Sub

  Private Sub Class_Terminate()
    Set CCSEvents = Nothing
    Set Errors = Nothing
```

```
  End Sub

  Function Open(Cmd)
    Set Open = Me
  End Function

  Property Get EOF()
    EOF = True
  End Property

  Property Get State()
    State = adStateClosed
  End Property

  Property Get Fields(Name)
    Fields = Empty
  End Property
End Class

'End clsEmptyDataSource Class

'clsDataSource Class 00-5C773C21
Class clsDataSource

  Public DataSourceType
  Public DataSource
  Public Errors, Connection, Parameters, CCSEvents
  Public Recordset
  Public PageSize
  Public Command

  Private mRecordCount
  Public Order

  Private objFields
  Private AbsolutePage
  Private prvRecordsCount
  Private builtSQL
  Private Opened
  Private MemoFields

  Private Sub Class_Initialize()
    Set CCSEvents = CreateObject("Scripting.Dictionary")
    Set MemoFields = CreateObject("Scripting.Dictionary")
    Set Errors = New clsErrors
    Set Parameters = New clsSQLParameters
    Set Parameters.DataSource = Me
    AbsolutePage = 0
    RecordCount = -1
    Opened = False
  End Sub

  Sub Close()
    If Recordset.State = adStateOpen Then
      Recordset.Close
    End If
    Opened = False
    Set Recordset = Nothing
  End Sub

  Function GetData(Pages)
    If Pages = 0 Then
      Set GetData = Recordset
    End If
  End Function

  Property Get Fields(Name)
    If IsNumeric(Name) Then
```

```
      Fields = CCGetValue(Recordset, CInt(Name))
    ElseIf IsObject(objFields) Then
      If Not objFields is Nothing Then
        If objFields.Exists(Name) Then
          If MemoFields.Exists(objFields(Name).DBFieldName) Then
            Fields = MemoFields(objFields(Name).DBFieldName)
          Else
            Fields = objFields(Name).Value
            If objFields(Name).DataType = ccsMemo Then
              MemoFields.Add objFields(Name).DBFieldName, Fields
            End If
          End If
        Else
          Fields = CCGetValue(Recordset, Name)
        End If
      Else
        Fields = CCGetValue(Recordset, Name)
      End If
    Else
      Fields = CCGetValue(Recordset, Name)
    End If
  End Property


  Property Set FieldsCollection(NewFieldsCollection)
    Set objFields = NewFieldsCollection
    If Not objFields Is Nothing Then
      objFields.InitEnum
      While Not objFields.EndOfEnum
        Set objFields.NextItem.DataSource = Me
      Wend
    End If
  End Property

  Property Get EOF()
    EOF = Recordset.EOF
  End Property

  Property Get State()
    If IsObject(Recordset) Then
      State = Recordset.State
    Else
      State = False
    End If
  End Property

  Sub MoveNext()
    Recordset.MoveNext
    MemoFields.RemoveAll
  End Sub

  Sub MoveFirst()
    Recordset.MoveFirst
    MemoFields.RemoveAll
  End Sub

  Function GetOrder(DefaultSorting, Sorter, Direction, MapArray)
    Dim OrderValue, I, ActiveSorter

    If NOT IsEmpty(Sorter) Then
      ' Select sorted column
      I = 0
      Do While I <= UBound(MapArray)
        If MapArray(I)(0) = Sorter Then
          ActiveSorter = I
          Exit Do
        End If
        I = I + 1
```

```
      Loop
      If NOT IsEmpty(ActiveSorter) Then
        If NOT IsEmpty(Direction) AND (Direction = "ASC" OR Direction =
    "DESC") Then
          If Direction = "ASC" Then
            OrderValue = MapArray(ActiveSorter)(1)
          ElseIf Direction = "DESC" Then
            OrderValue = MapArray(ActiveSorter)(2)
          End If
          If OrderValue = "" Then
            OrderValue = MapArray(ActiveSorter)(1) & " DESC"
          End If
        Else
          OrderValue = MapArray(ActiveSorter)(1)
        End If
      End If
    End If
    If Len(OrderValue) > 0 Then
      Order = OrderValue
    Else
      Order = DefaultSorting
    End If
    GetOrder = Order
  End Function

  Public Property Let RecordCount(vData)
    mRecordCount = vData
  End Property

  Public Property Get RecordCount()
    If mRecordCount < 0 Then
      mRecordCount = Command.ExecuteCount
    End If
    RecordCount = mRecordCount
  End Property

  Function MoveToPage(Page)
    Dim PageCounter
    Dim RecordCounter

    If Recordset.State = adStateOpen Then
      PageCounter = 1
      RecordCounter = 1
      While NOT Recordset.EOF AND PageCounter < Page
        If RecordCounter MOD Command.PageSize = 0 Then
          PageCounter = PageCounter + 1
        End If

        RecordCounter = RecordCounter + 1
        Recordset.MoveNext
      Wend
    End If
    Command.ActivePage = PageCounter
  End Function

  Function PageCount()
    Dim Result
    If Command.PageSize > 0 Then
      Result = RecordCount \ Command.PageSize
      If (RecordCount MOD Command.PageSize) > 0 Then
        Result = Result + 1
      End If
    Else
      Result = 1
    End If
    PageCount = Result
  End Function
```

```
  Private Sub Class_Terminate()
    Set Command = Nothing
    Set Errors = Nothing
    Set Parameters = Nothing
    Set CCSEvents = Nothing
  End Sub

End Class

'End clsDataSource Class

'clsCommand Class E0-B011FE53
Class clsCommand

  Private mCommandType
  Private mCommandOperation
  Private mPrepared

  Private mSQL
  Private mCountSQL
  Private mWhere
  Private mOrderBy
  Private mLOV
  Private mSP

  Private mPageSize
  Private mActivePage

  Private RecordsCount

  Public Errors, Connection, CCSEvents
  Public WhereParameters, Parameters
  Public CommandParameters

  Private Sub Class_Initialize()
    Set CCSEvents = CreateObject("Scripting.Dictionary")
    Set WhereParameters = New clsSQLParameters
    Set WhereParameters.ParameterSources = CreateObject("Scripting.
    Dictionary")
    Set Parameters = New clsSQLParameters
    Set Parameters.ParameterSources = CreateObject("Scripting.Dictionary")
    ActivePage = 0
    Prepared = False
  End Sub

  Public Function Exec(Err)
    Set Errors = Err
    Select Case CommandOperation
      Case cmdOpen
        Set Exec = DoOpen
      Case cmdExec
        DoExec
    End Select
  End Function

  Private Function OpenRecordset(sSQL)
    Dim Command
    Dim Recordset

    Set Command = CreateObject("ADODB.Command")
    Command.CommandType = adCmdText
    Command.CommandText = sSQL
    Set Command.ActiveConnection = Connection.Connection
    Set Recordset = Connection.Execute(Command)

    If Connection.Errors.Count > 0 Then
      Errors.AddError Connection.Errors.ToString & sSQL & Errors.
    ErrorDelimiter
```

```
    End If

    Set OpenRecordset = Recordset
    Set Command = Nothing
  End Function

  Private Function ParseParams(sSQL, Params)
    Dim I
    Dim NewSQL
    Dim ParamKeys
    Dim ParamItems

    NewSQL = sSQL
    If CommandType = dsSQL Then
      If Not Params is Nothing Then
        ParamItems = Params.ParametersList.Items
        ParamKeys = Params.ParametersList.Keys
        For I = 0 To UBound(ParamItems)
         NewSQL = Replace(NewSQL, "{" & ParamKeys(I) & "}", ParamItems(I).↙
    SQLText)
        Next
      End If
    End If
    ParseParams = NewSQL
  End Function

  Private Function DoOpen()
    Dim Command
    Dim builtSQL
    Dim DataSource
    Dim CountRecordset
    Dim ResultRecordset
    Dim CCSEventResult
    Dim ParameterValue
    Dim Parameter

    Set DataSource = new clsDataSource
    If IsObject(Connection) Then _
      Set DataSource.Connection = Connection
    Set DataSource.Command = Me

    Select Case CommandType
      Case dsTable, dsSQL
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        builtSQL = ParseParams(SQL & IIf(Len(Where) > 0, " WHERE " & Where↙
    , "") & IIf(Len(OrderBy) > 0, " ORDER BY " & OrderBy, ""),          ↙
    WhereParameters)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",   ↙
    Me)
        Set DataSource.Recordset = OpenRecordset(builtSQL)
        If ActivePage > 0 Then
          DataSource.MoveToPage ActivePage
        End If

        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        If Errors.Count > 0 Then
          DataSource.Errors.AddErrors Errors
        End If
        Set DoOpen = DataSource

      Case dsProcedure
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Set Command = CreateSP()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",   ↙
    Me)
        Set DataSource.Recordset = Connection.Execute(Command)

        If ActivePage > 0 Then
```

```
        DataSource.MoveToPage ActivePage
      End If

      Do Until DataSource.Recordset Is Nothing
         If DataSource.Recordset.Fields.Count >0 Then Exit Do
         Set DataSource.Recordset = DataSource.Recordset.NextRecordset
      Loop

      If Connection.Errors.Count > 0 Then
        DataSource.Errors.AddError Connection.Errors.ToString & mSP &    ↙
  Errors.ErrorDelimiter
      End If

      Set Command = Nothing
      Set DoOpen = DataSource
    Case dsListOfValues
      Dim I
      CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
      Set DataSource.Recordset = CreateObject("ADODB.Recordset")
      DataSource.Recordset.Fields.Append "bound", adBSTR, 256,           ↙
  adFldCacheDeferred + adFldUpdatable
      DataSource.Recordset.Fields.Append "text", adBSTR, 256,            ↙
  adFldCacheDeferred + adFldUpdatable
      CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
  Me)
      DataSource.Recordset.Open
      For I = 0 To UBound(mLOV(0))
        DataSource.Recordset.AddNew
        DataSource.Recordset.Fields("bound").Value = mLOV(0)(I)
        DataSource.Recordset.Fields("text").Value = mLOV(1)(I)
      Next
      DataSource.Recordset.Update
      DataSource.Recordset.MoveFirst
      CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
      Set DoOpen = DataSource
  End Select
End Function

Public Function ExecuteCount()
  Dim Result: Result = 0
  Dim builtSQL: builtSQL = ""
  Dim CountRecordset

  If Len(CountSQL) > 0 Then
    builtSQL = ParseParams(CountSQL & IIf(Len(Where) > 0, " WHERE " &    ↙
  Where, ""), WhereParameters)
    Set CountRecordset = OpenRecordset(builtSQL)
    If CountRecordset.State = adStateOpen Then
      Result = CLng(CountRecordset.Fields(0).Value)
    End If
    Set CountRecordset = Nothing
  End If
  ExecuteCount = Result
End Function

Private Function CreateSP()
  Dim Command, I, ParameterValue, Parameter, Sources

  Set Command = Server.CreateObject("ADODB.Command")
  Set Command.ActiveConnection = Connection.Connection
  Command.CommandType = adCmdStoredProc
  Command.CommandText = mSP
  If IsArray(CommandParameters) Then
    Set Sources = Parameters.ParameterSources
    For I = 0 To UBound(CommandParameters)
      ParameterValue = Sources(CommandParameters(I)(1))
      If IsEmpty(ParameterValue) Then
        ParameterValue = CommandParameters(I)(7)
```

```
        End If
        If IsEmpty(ParameterValue) Then
          ParameterValue = Null
        End If
        Set Parameter = Command.CreateParameter(CommandParameters(I)(0),  ↙
    CommandParameters(I)(2), CommandParameters(I)(3), CommandParameters(I)↙
    (4), ParameterValue)
        If Parameter.Type = adNumeric Then
          Parameter.NumericScale = CommandParameters(I)(5)
          Parameter.Precision = CommandParameters(I)(6)
        End If
        Command.Parameters.Append Parameter
      Next
      Set Sources = Nothing
    End If
    Set CreateSP = Command
  End Function

  Private Sub DoExec
    Dim Command, I
    Dim builtSQL
    Dim ParameterValue

    If CommandType = dsProcedure Then
      Set Command = CreateSP()
    Else
      Set Command = CreateObject("ADODB.Command")
      Command.CommandType = adCmdText

      builtSQL = SQL
      If CommandType = dsSQL Then
        builtSQL = ParseParams(builtSQL, WhereParameters)
        builtSQL = ParseParams(builtSQL, Parameters)
      Else
        If IsArray(CommandParameters) Then
          For I = 0 To UBound(CommandParameters)
            If IsEmpty(CommandParameters(I)(4)) Then
              ParameterValue = Null
            Else
              ParameterValue = CommandParameters(I)(4)
            End If
            Command.Parameters.Append Command.CreateParameter              ↙
    (CommandParameters(I)(0), CommandParameters(I)(1), CommandParameters  ↙
    (I)(2), CommandParameters(I)(3), ParameterValue)
          Next
        End If
      End If

      Command.CommandText = builtSQL
      Command.Prepared = Prepared

      Set Command.ActiveConnection = Connection.Connection
    End If
    Connection.Execute(Command)

    If Connection.Errors.Count > 0 Then
      Errors.AddError Connection.Errors.ToString & SQL & Errors.          ↙
    ErrorDelimiter
    End If

    Set Command = Nothing
  End Sub

  Public Property Let ActivePage(vData)
      mActivePage = vData
  End Property

  Public Property Get ActivePage()
```

```
    ActivePage = mActivePage
End Property

Public Property Let PageSize(vData)
    mPageSize = vData
End Property

Public Property Get PageSize()
    PageSize = mPageSize
End Property

Public Property Let CommandOperation(vData)
    mCommandOperation = vData
End Property

Public Property Get CommandOperation()
    CommandOperation = mCommandOperation
End Property

Public Property Let LOV(vData)
    mLOV = vData
End Property

Public Property Get LOV()
    LOV = mLOV
End Property

Public Property Let SP(vData)
    mSP = vData
End Property

Public Property Get SP()
    SP = mSP
End Property

Public Property Let CountSQL(vData)
    mCountSQL = vData
End Property

Public Property Get CountSQL()
    CountSQL = mCountSQL
End Property

Public Property Let SQL(vData)
    mSQL = vData
End Property

Public Property Get SQL()
    SQL = mSQL
End Property

Public Property Let Prepared(vData)
    mPrepared = vData
End Property

Public Property Get Prepared()
    Prepared = mPrepared
End Property

Public Property Let CommandType(vData)
    mCommandType = vData
End Property

Public Property Get CommandType()
    CommandType = mCommandType
End Property

Public Property Let OrderBy(vData)
```

```
      mOrderBy = vData
  End Property

  Public Property Get OrderBy()
      OrderBy = mOrderBy
  End Property

  Public Property Let Order(vData)
      mOrderBy = vData
  End Property

  Public Property Get Order()
      Order = mOrderBy
  End Property

  Public Property Let Where(vData)
      mWhere = vData
  End Property

  Public Property Get Where()
      Where = mWhere
  End Property
End Class

'End clsCommand Class

'clsConverter Class {0-5DD84A62
Class clsConverter
  Private mDateFormat
  Private mBooleanFormat
  Private mIntegerFormat
  Private mFloatFormat
  Private mParseError

  Private Sub Class_Initialize()
    mParseError = False
  End Sub

  Property Let DateFormat(newDateFormat)
    mDateFormat = newDateFormat
  End Property

  Property Get DateFormat()
    DateFormat = mDateFormat
  End Property

  Property Let BooleanFormat(newFormat)
    mBooleanFormat = newFormat
  End Property

  Property Get BooleanFormat()
    BooleanFormat = mBooleanFormat
  End Property

  Property Let IntegerFormat(newFormat)
    mIntegerFormat = newFormat
  End Property

  Property Get IntegerFormat()
    IntegerFormat = mIntegerFormat
  End Property

  Property Let FloatFormat(newFormat)
    mFloatFormat = newFormat
  End Property

  Property Get FloatFormat()
    FloatFormat = mFloatFormat
```

```
  End Property

  Property Get ParseError()
    ParseError = mParseError
  End Property

  Public Function VBSConvert(DataType, Value)
    Dim Result

    mParseError = False
    Result = Empty

    If IsEmpty(Value) Then
      VBSConvert = Result
      Exit Function
    End If

    On Error Resume Next
    Select Case DataType
      Case ccsDate
        Result = CDate(Value)
      Case ccsBoolean
        Result = CBool(Value)
      Case ccsInteger
        Result = CLng(Value)
      Case ccsFloat
        Result = CDbl(Value)
      Case ccsText, ccsMemo
        Result = CStr(Value)
    End Select
    If Err.Number <> 0 Then _
      mParseError = True
    On Error Goto 0
    VBSConvert = Result
  End Function

  Public Function StringToType(DataType, Value, Format)
    Dim CurrentFormat
    Dim Result

    mParseError = False
    Result = Empty
    If IsEmpty(Value) Then
      StringToType = Result
      Exit Function
    End If
    If IsEmpty(Format) Then
      Select Case DataType
        Case ccsDate
          CurrentFormat = mDateFormat
        Case ccsBoolean
          CurrentFormat = mBooleanFormat
        Case ccsInteger
          CurrentFormat = mIntegerFormat
        Case ccsFloat
          CurrentFormat = mFloatFormat
      End Select
    Else
      CurrentFormat = Format
    End If

    On Error Resume Next
    Select Case DataType
      Case ccsDate
        Result = CCParseDate(Value, CurrentFormat)
      Case ccsBoolean
        Result = CCParseBoolean(Value, CurrentFormat)
      Case ccsInteger
```

```
        Result = CCParseInteger(Value, CurrentFormat)
      Case ccsFloat
        Result = CCParseFloat(Value, CurrentFormat)
      Case ccsText, ccsMemo
        Result = CStr(Value)
    End Select
    If Err.Number <> 0 Then _
      mParseError = True
    On Error Goto 0
    StringToType = Result
  End Function

  Public Function TypeToString(DataType, Value, Format)
    Dim CurrentFormat
    Dim Result
    Dim VarDataType
    Dim CurrentDataType

    VarDataType = VarType(Value)
    Select Case VarDataType
      Case vbInteger, vbLong, vbByte:
        CurrentDataType = ccsInteger
      Case vbSingle, vbDouble, vbCurrency, vbDecimal:
        CurrentDataType = ccsFloat
      Case vbDate:
        CurrentDataType = ccsDate
      Case vbBoolean:
        CurrentDataType = ccsBoolean
      Case vbString:
        CurrentDataType = ccsText
      Case vbArray, vbDataObject, vbVariant, vbError, vbObject, vbNull:
        Err.Raise 1057, "Type mismatch"
    End Select
    If DataType = ccsMemo AND CurrentDataType = ccsText Then _
       CurrentDataType = ccsMemo
    If (VarDataType <> vbEmpty) AND (CurrentDataType <> DataType) Then _
        Err.Raise 1057, "Type mismatch"

    If IsEmpty(Format) Then
      Select Case DataType
        Case ccsDate
          CurrentFormat = mDateFormat
        Case ccsBoolean
          CurrentFormat = mBooleanFormat
        Case ccsInteger
          CurrentFormat = mIntegerFormat
        Case ccsFloat
          CurrentFormat = mFloatFormat
      End Select
    Else
      CurrentFormat = Format
    End If

    Select Case DataType
      Case ccsDate
        Result = CCFormatDate(Value, CurrentFormat)
      Case ccsBoolean
        Result = CCFormatBoolean(Value, CurrentFormat)
      Case ccsInteger
        Result = CCFormatNumber(Value, CurrentFormat)
      Case ccsFloat
        Result = CCFormatNumber(Value, CurrentFormat)
      Case ccsText, ccsMemo
        Result = CStr(Value)
    End Select
    TypeToString = Result
  End Function
End Class
```

```
'End clsConverter Class

'clsRunTimeMessages Class 00-40EC4C7C
Class clsRunTimeMessages
  Private mMessages

  Private Sub Class_Initialize()
    Set mMessages = Server.CreateObject("Scripting.Dictionary")
    mMessages.Add "IncorrectFormat", "The value in field {FieldName} is ↙
    not valid. Use the following format: {Format}."
    mMessages.Add "IncorrectValue", "The value in field {FieldName} is not↙
     valid."
    mMessages.Add "RequiredField", "The value in field {FieldName} is ↙
    required."
  End Sub

  Function GetMessage(MsgID, Params)
    Dim Result, I
    Result = ""

    If mMessages.Exists(MsgID) Then
      Result = mMessages(MsgID)
      If IsArray(Params) Then
        For I = 0 To UBound(Params) Step 2
          Result = Replace(Result, "{" & Params(I) & "}", Params(I+1))
        Next
      End If
    End If
    GetMessage = Result
  End Function

  Private Sub Class_Terminate()
    Set mMessages = Nothing
  End Sub
End Class
'End clsRunTimeMessages Class

'clsStringBuffer 00-0A3F192B
Class clsStringBuffer
  Private incremetRate
  Private itemCount
  Private items

  Private Sub Class_Initialize()
    incremetRate = 50
    itemCount = 0
    ReDim items(incremetRate)
  End Sub

  Public Sub Append(ByVal strValue)
    If itemCount > UBound(items) Then
      ReDim Preserve items(UBound(items) + incremetRate)
    End If

    items(itemCount) = strValue
    itemCount = itemCount + 1
  End Sub

  Public Function ToString()
    ToString = Join(items, "")
  End Function

End Class

'End clsStringBuffer


%>
```

```
<%
'Include Common Files @1-D6EDE132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-74245E08
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Page controls
Dim Login1

Redirect = ""
TemplateFileName = "ADMIN_Login.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "ADMIN_Login.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-FA8B07EB

' Controls
Set Login1 = new clsRecordLogin1

' Events
%>
<!-- #INCLUDE FILE="ADMIN_Login_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-174D3587
Login1.Operation
'End Execute Components

'Go to destination page @1-6D35F4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-4641E7BA
Login1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-E3D633D8
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    Set Login1 = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordLogin1 'Login1 Class @8-14679A0A

'Login1 Variables @8-55290F7E

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim login
    Dim password
    Dim Button_DoLogin
'End Login1 Variables

'Login1 Class_Initialize Event @8-D0B89977
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Login1")
            End If
            If UBound(OperationMode) > 0 Then
```

```
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Login1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set login = CCCreateControl(ccsTextBox, "login", "login",      ↙
    ccsText, Empty, CCGetRequestParam("login", Method))
            login.Required = True
            Set password = CCCreateControl(ccsTextBox, "password",         ↙
    "password", ccsText, Empty, CCGetRequestParam("password", Method))
            password.Required = True
            Set Button_DoLogin = CCCreateButton("Button_DoLogin")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(login, password)
        End If
    End Sub
'End Login1 Class_Initialize Event

'Login1 Class_Terminate Event @8-32B647C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Login1 Class_Terminate Event

'Login1 Validate Method @8-7AE5264D
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Login1 Validate Method

'Login1 Operation Method @8-DED6F479
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoLogin"
            If Not IsEmpty(CCGetParam("Button_DoLogin", Empty)) Then
                PressedButton = "Button_DoLogin"
            End If
        End If
        Redirect = FileName & ""
        If Validate() Then
            If PressedButton = "Button_DoLogin" Then
                If NOT Button_DoLogin.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_Menu.asp?" & CCGetQueryString(       ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Login1 Operation Method

'Login1 Show Method @8-89987474
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.              ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Login1" & IIf(EditMode, "↙
    :Edit", ""))
```

```
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,            ↙
    ccsParseOverwrite, _
            Array(login, password, Button_DoLogin))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Login1_Show_Method

End Class 'End Login1_Class_C8-A61BA894


%>
```

```
<%
'BindEvents Method @1-194D6F1A
Sub BindEvents()
    Set Login1.Button_DoLogin.CCSEvents("OnClick") = GetRef(
    "Login1_Button_DoLogin_OnClick")
End Sub
'End BindEvents Method

Function Login1_Button_DoLogin_OnClick() 'Login1_Button_DoLogin_OnClick @9
    -394D3726

'Login @10-055F700C
    With Login1
        If NOT CCLoginUser(.login.Value, .password.Value) Then
            .Errors.addError("Login or Password is incorrect.")
            Login1_Button_DoLogin_OnClick = False
            .password.Value = ""
        Else
            If Not IsEmpty(CCGetParam("ret_link", Empty)) Then _
                Redirect = CCGetParam("ret_link", Empty)
            Login1_Button_DoLogin_OnClick = True
        End If
    End With
'End Login

End Function 'Close Login1_Button_DoLogin_OnClick @9-54C94B3E


%>
```

```
<%
'Include Common Files @1-D6EDE132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-137E7116
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim STAFF

Redirect = ""
TemplateFileName = "ADMIN_Logout.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "ADMIN_Logout.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-9410C1BD
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set STAFF = new clsRecordSTAFF
STAFF.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-DBC543BB
STAFF.Operation
'End Execute Components

'Go to destination page @1-6D35F4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-1C138EB0
STAFF.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-001CD4BE
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set STAFF = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordSTAFF 'STAFF Class @5-[illegible]

'STAFF Variables @5-1[illegible]

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button1
'End STAFF Variables

'STAFF Class_Initialize Event @5-63E6AF01
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsSTAFFDataSource
        Set Command = New clsCommand
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
```

```
                FormSubmitted = (OperationMode(0) = "STAFF")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "STAFF"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button1 = CCCreateButton("Button1")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End STAFF Class_Initialize Event

'STAFF Initialize Method 65-AFB984CF
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlSTAFFid") = CCGetRequestParam("STAFFid",
    ccsGET)
        End With
    End Sub
'End STAFF Initialize Method

'STAFF Class_Terminate Event 65-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End STAFF Class_Terminate Event

'STAFF Validate Method 65-7AE5264C
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End STAFF Validate Method

'STAFF Operation Method 65-E7044DBE
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button1"
            If Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            End If
        End If
        Redirect = "ADMIN_Logout.asp?" & CCGetQueryString("QueryString",
    Array("ccsForm", "Button1"))
        If Validate() Then
            If PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_Login.asp?" & CCGetQueryString(
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
```

```
'End STAFF Operation Method

'STAFF Show Method 65-9E58C7C9
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "STAFF" & IIf(EditMode, ":↙
    Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(Button1))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record STAFF", DataSource.Connection.      ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                    End If
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End STAFF Show Method

End Class 'End STAFF Class 65-A61BA892

Class clsSTAFFDataSource 'STAFFDataSource Class 65-EC2F65AE

'DataSource Variables 65-184E7B43
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
```

```
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

'End DataSource Variables

'DataSource Class_Initialize Event @5-04536067
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM STAFF"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM STAFF"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method @5-1B2F0633
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlSTAFFid", ccsText, Empty, Empty, Empty,  ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STAFFid", .       ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method @5-DF951BED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
```

```
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event 65-41B4B08D
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End STAFFDataSource Class 65-A61BA892


%>
```

```
<%
'Include Common Files @1-D6EBE132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-9D512830
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim STAFF

Redirect = ""
TemplateFileName = "ADMIN_Menu.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "ADMIN_Menu.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-9410C1BD
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set STAFF = new clsRecordSTAFF
STAFF.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-DB0543BB
STAFF.Operation
'End Execute Components

'Go to destination page @1-6D35F4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-1C138EB0
STAFF.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-001CD4BE
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set STAFF = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordSTAFF 'STAFF Class @2-0E3A4D42

'STAFF Variables @2-D46210AE

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Label1
    Dim Button1
    Dim Button2
    Dim Button3
    Dim Button4
'End STAFF Variables

'STAFF Class_Initialize Event @2-99A731DB
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsSTAFFDataSource
        Set Command = New clsCommand
        Dim Method
        Dim OperationMode
```

```
        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "STAFF")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "STAFF"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Label1 = CCCreateControl(ccsLabel, "Label1", "Label1",
    ccsText, Empty, CCGetRequestParam("Label1", Method))
            Set Button1 = CCCreateButton("Button1")
            Set Button2 = CCCreateButton("Button2")
            Set Button3 = CCCreateButton("Button3")
            Set Button4 = CCCreateButton("Button4")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End STAFF Class_Initialize Event

'STAFF Initialize Method 62-AFB984CF
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlSTAFFid") = CCGetRequestParam("STAFFid",
    ccsGET)
        End With
    End Sub
'End STAFF Initialize Method

'STAFF Class_Terminate Event 62-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End STAFF Class_Terminate Event

'STAFF Validate Method 62-7AE52842
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End STAFF Validate Method

'STAFF Operation Method 62-9A7DB7B1
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button1"
            If Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            ElseIf Not IsEmpty(CCGetParam("Button2", Empty)) Then
                PressedButton = "Button2"
            ElseIf Not IsEmpty(CCGetParam("Button3", Empty)) Then
                PressedButton = "Button3"
            ElseIf Not IsEmpty(CCGetParam("Button4", Empty)) Then
                PressedButton = "Button4"
            End If
        End If
        Redirect = "ADMIN_Menu.asp?" & CCGetQueryString("QueryString",
```

```
    Array("ccsForm", "Button1", "Button2", "Button3", "Button4"))
        If Validate() Then
            If PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_StaffList.asp?" & CCGetQueryString( ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button2" Then
                If NOT Button2.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "DataEntry.asp?" & CCGetQueryString(       ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button3" Then
                If NOT Button3.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_ChangePwd.asp?" & CCGetQueryString( ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button4" Then
                If NOT Button4.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_Logout.asp?" & CCGetQueryString(    ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End STAFF Operation Method

'STAFF Show Method 62-539CD509
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.             ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "STAFF" & IIf(EditMode, ":↙
    Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,            ↙
    ccsParseOverwrite, _
            Array(Label1, Button1, Button2, Button3, Button4))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record STAFF", DataSource.Connection.   ↙
    LastSQL, Recordset.Errors.ToString
```

```
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                    End If
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
            Label1.Value = Session("UserLogin")
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End STAFF Show Method

End Class 'End STAFF Class @2-A61BA892

Class clsSTAFFDataSource 'STAFFDataSource Class @2-EC2F85AE

'DataSource Variables @2-184B7B43
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

'End DataSource Variables

'DataSource Class_Initialize Event @2-04536065
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM STAFF"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM STAFF"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event
```

```
'BuildTableWhere Method 62-1B2F0633
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlSTAFFid", ccsText, Empty, Empty, Empty,  ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STAFFid", .       ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 62-2F951BED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",   ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event 62-41B4B08D
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End STAFFDataSource Class 62-A61BA892
```

```
<%
'Include Common Files @1-D6EC3E132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sender.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-E18BBB24
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim NewRecord1
Dim STAFF

Redirect = ""
TemplateFileName = "ADMIN_ChangePwd.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "ADMIN_ChangePwd.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-07C3376F
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set NewRecord1 = new clsRecordNewRecord1
Set STAFF = new clsRecordSTAFF
STAFF.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-2755C9AA
NewRecord1.Operation
STAFF.Operation
'End Execute Components

'Go to destination page @1-6C3574FC
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
```

```
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page 61-AE40C715
NewRecord1.Show Tpl
STAFF.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page 61-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub 61-60A9CC17
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set NewRecord1 = Nothing
    Set STAFF = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordNewRecord1 'NewRecord1 Class 69-11C1D1C9

'NewRecord1 Variables 69-1C90372E

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button1
'End NewRecord1 Variables

'NewRecord1 Class_Initialize Event 69-17906A94
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode
```

```
        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "NewRecord1")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "NewRecord1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button1 = CCCreateButton("Button1")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End NewRecord1 Class_Initialize Event

'NewRecord1 Class_Terminate Event @9-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End NewRecord1 Class_Terminate Event

'NewRecord1 Validate Method @9-7AE5264D
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End NewRecord1 Validate Method

'NewRecord1 Operation Method @9-CC003130
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button1"
            If Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            End If
        End If
        Redirect = FileName & "?" & CCGetQueryString("QueryString", Array ↙
    ("ccsForm", "Button1"))
        If Validate() Then
            If PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_Menu.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End NewRecord1 Operation Method

'NewRecord1 Show Method @9-5DA0752E
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.              ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "NewRecord1" & IIf         ↙
    (EditMode, ":Edit", ""))
```

```
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,            ↙
    ccsParseOverwrite, _
            Array(Button1))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End NewRecord1 Show Method

End Class 'End NewRecord1 Class @9-A61BA892

Class clsRecordSTAFF 'STAFF Class @2-0D3A4D42

'STAFF Variables @2-9BA0623B

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public UpdateAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim STAFFid
    Dim STAFFname
    Dim STAFFlogin
    Dim STAFFpwd
    Dim Button_Update
'End STAFF Variables

'STAFF Class_Initialize Event @2-9B5371B6
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsSTAFFDataSource
        Set Command = New clsCommand
        UpdateAllowed = True
        Dim Method
        Dim OperationMode
```

```
        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "STAFF")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "STAFF"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set STAFFid = CCCreateControl(ccsLabel, "STAFFid", "Id",        ↙
    ccsText, Empty, CCGetRequestParam("STAFFid", Method))
            Set STAFFname = CCCreateControl(ccsLabel, "STAFFname", "Name",↙
     ccsText, Empty, CCGetRequestParam("STAFFname", Method))
            Set STAFFlogin = CCCreateControl(ccsTextBox, "STAFFlogin",      ↙
    "Login", ccsText, Empty, CCGetRequestParam("STAFFlogin", Method))
            Set STAFFpwd = CCCreateControl(ccsTextBox, "STAFFpwd", "Pwd", ↙
    ccsText, Empty, CCGetRequestParam("STAFFpwd", Method))
            Set Button_Update = CCCreateButton("Button_Update")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STAFFlogin, STAFFpwd)
        End If
    End Sub
'End STAFF Class_Initialize Event

'STAFF Initialize Method 02-EF4CAACE
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("sesUSERid") = Session("USERid")
        End With
    End Sub
'End STAFF Initialize Method

'STAFF Class_Terminate Event 02-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End STAFF Class_Terminate Event

'STAFF Validate Method 02-7AE5264D
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End STAFF Validate Method

'STAFF Operation Method 02-45979D8A
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "")
            If Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            End If
        End If
        Redirect = "ADMIN_ChangePwd.asp?" & CCGetQueryString("QueryString"↙
    , Array("ccsForm", "Button_Update"))
        If Validate() Then
            If PressedButton = "Button_Update" Then
```

```
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_Menu.asp?" & CCGetQueryString(        ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End STAFF Operation Method

'STAFF UpdateRow Method 62-07685976
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.STAFFlogin.Value = STAFFlogin.Value
        DataSource.STAFFpwd.Value = STAFFpwd.Value
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End STAFF UpdateRow Method

'STAFF Show Method 62-2093F6EA
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.            ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "STAFF" & IIf(EditMode, ":↙
    Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,           ↙
    ccsParseOverwrite,
            Array(STAFFid, STAFFname, STAFFlogin, STAFFpwd,              ↙
    Button_Update))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record STAFF", DataSource.Connection.  ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        STAFFlogin.Value = Recordset.Fields("STAFFlogin")
                        STAFFpwd.Value = Recordset.Fields("STAFFpwd")
                    End If
                    STAFFid.Value = Recordset.Fields("STAFFid")
```

```
                    STAFFname.Value = Recordset.Fields("STAFFname")
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Update.Visible = EditMode AND UpdateAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End STAFF Show Method

End Class 'End STAFF Class 62-A61BA892

Class clsSTAFFDataSource 'STAFFDataSource Class 62-EC2F65AE

'DataSource Variables 62-70A06A65
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STAFFid
    Public STAFFname
    Public STAFFlogin
    Public STAFFpwd
'End DataSource Variables

'DataSource Class_Initialize Event 62-C6678C6D
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STAFFid = CCCreateField("STAFFid", "STAFFid", ccsText, Empty, ↙
    Recordset)
        Set STAFFname = CCCreateField("STAFFname", "STAFFname", ccsText, ↙
    Empty, Recordset)
        Set STAFFlogin = CCCreateField("STAFFlogin", "STAFFlogin", ccsText↙
    , Empty, Recordset)
        Set STAFFpwd = CCCreateField("STAFFpwd", "STAFFpwd", ccsText, ↙
    Empty, Recordset)
        Fields.AddFields Array(STAFFid, STAFFname, STAFFlogin, STAFFpwd)
```

```
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *   " & _
        "FROM STAFF"
        CountSQL = "SELECT COUNT(*)   " & _
        "FROM STAFF"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method 62-C4E7EF57
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "sesUSERid", ccsText, Empty, Empty, Empty,    ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STAFFid", .        ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 61-3F9518E3
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method
```

```
'DataSource Class_Terminate Event 62-41B4B0E3
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Update Method 62-CD9185E9
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE STAFF SET " & _
            "STAFFlogin=" & Connection.ToSQL(STAFFlogin.SQLText,
    STAFFlogin.DataType) & ", " & _
            "STAFFpwd=" & Connection.ToSQL(STAFFpwd.SQLText, STAFFpwd.
    DataType) & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

End Class 'End STAFFDataSource Class 62-A61BA898


%>
```

```
<%
'Include Common Files @1-D6EBF132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-2F0939E3
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim NewRecord1
Dim STAFF

Redirect = ""
TemplateFileName = "ADMIN_RegisterStaff.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "ADMIN_RegisterStaff.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-979C376F
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set NewRecord1 = new clsRecordNewRecord1
Set STAFF = new clsRecordSTAFF
STAFF.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-2755C9AA
NewRecord1.Operation
STAFF.Operation
'End Execute Components

'Go to destination page @1-6535F4F3
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
```

```
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page 61-AE40D715
NewRecord1.Show Tpl
STAFF.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page 61-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub 61-60A9CC17
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set NewRecord1 = Nothing
    Set STAFF = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordNewRecord1 'NewRecord1 Class 617-11C1D1C6

'NewRecord1 Variables 617-22DD0617

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button1
    Dim Button2
    Dim Button3
'End NewRecord1 Variables

'NewRecord1 Class_Initialize Event 617-71D98076
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
```

```
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "NewRecord1")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "NewRecord1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button1 = CCCreateButton("Button1")
            Set Button2 = CCCreateButton("Button2")
            Set Button3 = CCCreateButton("Button3")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End NewRecord1 Class_Initialize Event

'NewRecord1 Class_Terminate Event @17-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End NewRecord1 Class_Terminate Event

'NewRecord1 Validate Method @17-7AF52642
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End NewRecord1 Validate Method

'NewRecord1 Operation Method @17-6BA7C47E
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button1"
            If Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            ElseIf Not IsEmpty(CCGetParam("Button2", Empty)) Then
                PressedButton = "Button2"
            ElseIf Not IsEmpty(CCGetParam("Button3", Empty)) Then
                PressedButton = "Button3"
            End If
        End If
        Redirect = FileName & "?" & CCGetQueryString("QueryString", Array ↙
    ("ccsForm", "Button1", "Button2", "Button3"))
        If Validate() Then
            If PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_RegisterStaff.asp?" &                ↙
    CCGetQueryString("QueryString", Array("ccsForm", "STAFFid"))

                End If
            ElseIf PressedButton = "Button2" Then
                If NOT Button2.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_StaffList.asp?" & CCGetQueryString( ↙
    "QueryString", Array("ccsForm", "STAFFid"))

                End If
```

```
            ElseIf PressedButton = "Button3" Then
                If NOT Button3.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_Menu.asp?" & CCGetQueryString(        ↙
    "QueryString", Array("ccsForm", "STAFFid"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End NewRecord1 Operation Method

'NewRecord1 Show Method @17-316F51FA
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.               ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "NewRecord1" & IIf          ↙
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,              ↙
    ccsParseOverwrite, _
            Array(Button1, Button2, Button3))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End NewRecord1 Show Method

End Class 'End NewRecord1 Class @17-A61BA892

Class clsRecordSTAFF 'STAFF Class @2-0C3A4D42

'STAFF Variables @2-09F01108

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public InsertAllowed
```

```
    Public UpdateAllowed
    Public DeleteAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim STAFFname
    Dim STAFFid
    Dim STAFFlogin
    Dim STAFFpwd
    Dim STAFFpost
    Dim STAFFstatus
    Dim STAFFsysadmin
    Dim STAFFadmin
    Dim STAFFprint
    Dim STAFFtimestamp
    Dim Button_Insert
    Dim Button_Update
    Dim Button_Delete
'End STAFF Variables

'STAFF Class_Initialize Event @2-82BCFE37
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsSTAFFDataSource
        Set Command = New clsCommand
        InsertAllowed = True
        UpdateAllowed = True
        DeleteAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "STAFF")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "STAFF"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set STAFFname = CCCreateControl(ccsTextBox, "STAFFname", "Name↙
    ", ccsText, Empty, CCGetRequestParam("STAFFname", Method))
            Set STAFFid = CCCreateControl(ccsTextBox, "STAFFid", "Id",     ↙
    ccsText, Empty, CCGetRequestParam("STAFFid", Method))
            STAFFid.Required = True
            Set STAFFlogin = CCCreateControl(ccsTextBox, "STAFFlogin",     ↙
    "Login", ccsText, Empty, CCGetRequestParam("STAFFlogin", Method))
            Set STAFFpwd = CCCreateControl(ccsTextBox, "STAFFpwd", "Pwd", ↙
    ccsText, Empty, CCGetRequestParam("STAFFpwd", Method))
            Set STAFFpost = CCCreateControl(ccsTextBox, "STAFFpost", "Post↙
    ", ccsText, Empty, CCGetRequestParam("STAFFpost", Method))
            Set STAFFstatus = CCCreateControl(ccsTextBox, "STAFFstatus",   ↙
    "Status", ccsText, Empty, CCGetRequestParam("STAFFstatus", Method))
            Set STAFFsysadmin = CCCreateControl(ccsCheckBox,               ↙
    "STAFFsysadmin", "Sysadmin", ccsBoolean, Array("True", "False", Empty)↙
    , CCGetRequestParam("STAFFsysadmin", Method))
            STAFFsysadmin.CheckedValue = True
            STAFFsysadmin.UncheckedValue = False
            Set STAFFadmin = CCCreateControl(ccsCheckBox, "STAFFadmin",    ↙
    "Admin", ccsBoolean, Array("True", "False", Empty), CCGetRequestParam ↙
    ("STAFFadmin", Method))
```

```
            STAFFadmin.CheckedValue = True
            STAFFadmin.UncheckedValue = False
            Set STAFFprint = CCCreateControl(ccsCheckBox, "STAFFprint",
    "Print", ccsBoolean, Array("True", "False", Empty), CCGetRequestParam
    ("STAFFprint", Method))
            STAFFprint.CheckedValue = True
            STAFFprint.UncheckedValue = False
            Set STAFFtimestamp = CCCreateControl(ccsLabel, "STAFFtimestamp
    ", "Timestamp", ccsDate, Array("dd", "/", "mm", "/", "yyyy"),
    CCGetRequestParam("STAFFtimestamp", Method))
            Set Button_Insert = CCCreateButton("Button_Insert")
            Set Button_Update = CCCreateButton("Button_Update")
            Set Button_Delete = CCCreateButton("Button_Delete")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STAFFname, STAFFid,
    STAFFlogin, STAFFpwd, STAFFpost, STAFFstatus, STAFFsysadmin,
    STAFFadmin, STAFFprint)
        End If
    End Sub
'End STAFF Class_Initialize Event

'STAFF Initialize Method 62-AFB984CF
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlSTAFFid") = CCGetRequestParam("STAFFid",
    ccsGET)
        End With
    End Sub
'End STAFF Initialize Method

'STAFF Class_Terminate Event 62-92BF4709
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End STAFF Class_Terminate Event

'STAFF Validate Method 62-7AE52640
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End STAFF Validate Method

'STAFF Operation Method 62-BC9E1217
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "Button_Insert"
    )
            If Not IsEmpty(CCGetParam("Button_Insert", Empty)) Then
                PressedButton = "Button_Insert"
            ElseIf Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            ElseIf Not IsEmpty(CCGetParam("Button_Delete", Empty)) Then
                PressedButton = "Button_Delete"
            End If
        End If
        Redirect = "ADMIN_RegisterStaff.asp?" & CCGetQueryString(
    "QueryString", Array("ccsForm", "Button_Insert", "Button_Update",
    "Button_Delete"))
```

```
        If PressedButton = "Button_Delete" Then
            If NOT Button_Delete.OnClick OR NOT DeleteRow() Then
                Redirect = ""
            Else
                Redirect = "ADMIN_StaffList.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm"))
            End If
        ElseIf Validate() Then
            If PressedButton = "Button_Insert" Then
                If NOT Button_Insert.OnClick() OR NOT InsertRow() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_StaffList.asp?" & CCGetQueryString( ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_StaffList.asp?" & CCGetQueryString( ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End STAFF Operation Method

'STAFF InsertRow Method 62-4CA4CB02
    Function InsertRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeInsert", Me)
        If NOT InsertAllowed Then InsertRow = False : Exit Function
        DataSource.STAFFname.Value = STAFFname.Value
        DataSource.STAFFid.Value = STAFFid.Value
        DataSource.STAFFlogin.Value = STAFFlogin.Value
        DataSource.STAFFpwd.Value = STAFFpwd.Value
        DataSource.STAFFpost.Value = STAFFpost.Value
        DataSource.STAFFstatus.Value = STAFFstatus.Value
        DataSource.STAFFsysadmin.Value = STAFFsysadmin.Value
        DataSource.STAFFadmin.Value = STAFFadmin.Value
        DataSource.STAFFprint.Value = STAFFprint.Value
        DataSource.Insert(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInsert", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Insert Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        InsertRow = (Errors.Count = 0)
    End Function
'End STAFF InsertRow Method

'STAFF UpdateRow Method 62-1F5CA2F1
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.STAFFname.Value = STAFFname.Value
        DataSource.STAFFid.Value = STAFFid.Value
        DataSource.STAFFlogin.Value = STAFFlogin.Value
        DataSource.STAFFpwd.Value = STAFFpwd.Value
        DataSource.STAFFpost.Value = STAFFpost.Value
        DataSource.STAFFstatus.Value = STAFFstatus.Value
```

```
        DataSource.STAFFsysadmin.Value = STAFFsysadmin.Value
        DataSource.STAFFadmin.Value = STAFFadmin.Value
        DataSource.STAFFprint.Value = STAFFprint.Value
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End STAFF UpdateRow Method

'STAFF DeleteRow Method 02-1BC26772
    Function DeleteRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeDelete", Me)
        If NOT DeleteAllowed Then DeleteRow = False : Exit Function
        DataSource.Delete(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterDelete", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Delete Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        DeleteRow = (Errors.Count = 0)
    End Function
'End STAFF DeleteRow Method

'STAFF Show Method 02-4884C95D
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "STAFF" & IIf(EditMode, ":↙
    Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(STAFFname, STAFFid, STAFFlogin, STAFFpwd, STAFFpost,       ↙
    STAFFstatus, STAFFsysadmin, STAFFadmin, STAFFprint, STAFFtimestamp,      ↙
    Button_Insert, Button_Update, Button_Delete))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record STAFF", DataSource.Connection.      ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        STAFFname.Value = ucase(Recordset.Fields(            ↙
    "STAFFname"))
```

```
                        STAFFid.Value = Recordset.Fields("STAFFid")
                        STAFFlogin.Value = Recordset.Fields("STAFFlogin")
                        STAFFpwd.Value = Recordset.Fields("STAFFpwd")
                        STAFFpost.Value = ucase(Recordset.Fields(
    "STAFFpost"))
                        STAFFstatus.Value = ucase(Recordset.Fields(
    "STAFFstatus"))
                        STAFFsysadmin.Value = Recordset.Fields(
    "STAFFsysadmin")
                        STAFFadmin.Value = Recordset.Fields("STAFFadmin")
                        STAFFprint.Value = Recordset.Fields("STAFFprint")
                    End If
                    STAFFtimestamp.Value = Recordset.Fields(
    "STAFFtimestamp")
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Insert.Visible = NOT EditMode AND InsertAllowed
        Button_Update.Visible = EditMode AND UpdateAllowed
        Button_Delete.Visible = EditMode AND DeleteAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End STAFF Show Method

End Class 'End STAFF Class @2-A61BA892

Class clsSTAFFDataSource 'STAFFDataSource Class @2-B7EF95AE

'DataSource Variables @2-CE20F8C9
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STAFFname
    Public STAFFid
    Public STAFFlogin
    Public STAFFpwd
    Public STAFFpost
    Public STAFFstatus
    Public STAFFsysadmin
    Public STAFFadmin
```

```
    Public STAFFprint
    Public STAFFtimestamp
'End DataSource Variables

'DataSource Class_Initialize Event 62-E45D1FF3
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STAFFname = CCCreateField("STAFFname", "STAFFname", ccsText,  ↙
    Empty, Recordset)
        Set STAFFid = CCCreateField("STAFFid", "STAFFid", ccsText, Empty, ↙
    Recordset)
        Set STAFFlogin = CCCreateField("STAFFlogin", "STAFFlogin", ccsText↙
    , Empty, Recordset)
        Set STAFFpwd = CCCreateField("STAFFpwd", "STAFFpwd", ccsText,     ↙
    Empty, Recordset)
        Set STAFFpost = CCCreateField("STAFFpost", "STAFFpost", ccsText,  ↙
    Empty, Recordset)
        Set STAFFstatus = CCCreateField("STAFFstatus", "STAFFstatus",     ↙
    ccsText, Empty, Recordset)
        Set STAFFsysadmin = CCCreateField("STAFFsysadmin", "STAFFsysadmin"↙
    , ccsBoolean, Array(1, 0, Empty), Recordset)
        Set STAFFadmin = CCCreateField("STAFFadmin", "STAFFadmin",        ↙
    ccsBoolean, Array(1, 0, Empty), Recordset)
        Set STAFFprint = CCCreateField("STAFFprint", "STAFFprint",        ↙
    ccsBoolean, Array(1, 0, Empty), Recordset)
        Set STAFFtimestamp = CCCreateField("STAFFtimestamp",              ↙
    "STAFFtimestamp", ccsDate, Array("yyyy", "-", "mm", "-", "dd", " ",   ↙
    "HH", ":", "nn", ":", "ss"), Recordset)
        Fields.AddFields Array(STAFFname, STAFFid, STAFFlogin, STAFFpwd,  ↙
    STAFFpost, STAFFstatus, STAFFsysadmin, STAFFadmin, STAFFprint,        ↙
    STAFFtimestamp)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM STAFF"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM STAFF"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method 62-1B2F0633
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlSTAFFid", ccsText, Empty, Empty, Empty,  ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STAFFid", .       ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
```

```
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 62-DF951BED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event 62-41B4B0E3
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Delete Method 62-97BD2E6B
    Sub Delete(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildDelete", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Delete operation. There↙
     are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "DELETE FROM STAFF" & IIf(Len(Where) > 0, " WHERE " &    ↙
    Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteDelete",    ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteDelete", Me)
    End Sub
'End Delete Method
```

```
'Update Method 62-E5FF530E
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.      ↙
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE STAFF SET " & _
            "STAFFname=" & Connection.ToSQL(STAFFname.SQLText, STAFFname. ↙
    DataType) & ", " & _
            "STAFFid=" & Connection.ToSQL(STAFFid.SQLText, STAFFid.       ↙
    DataType) & ", " & _
            "STAFFlogin=" & Connection.ToSQL(STAFFlogin.SQLText,          ↙
    STAFFlogin.DataType) & ", " & _
            "STAFFpwd=" & Connection.ToSQL(STAFFpwd.SQLText, STAFFpwd.    ↙
    DataType) & ", " & _
            "STAFFpost=" & Connection.ToSQL(STAFFpost.SQLText, STAFFpost. ↙
    DataType) & ", " & _
            "STAFFstatus=" & Connection.ToSQL(STAFFstatus.SQLText,        ↙
    STAFFstatus.DataType) & ", " & _
            "STAFFsysadmin=" & Connection.ToSQL(STAFFsysadmin.SQLText,    ↙
    STAFFsysadmin.DataType) & ", " & _
            "STAFFadmin=" & Connection.ToSQL(STAFFadmin.SQLText,          ↙
    STAFFadmin.DataType) & ", " & _
            "STAFFprint=" & Connection.ToSQL(STAFFprint.SQLText,          ↙
    STAFFprint.DataType) & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",   ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

'Insert Method 62-A95381BA
    Sub Insert(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildInsert", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        Cmd.SQL = "INSERT INTO STAFF(" & _
            "STAFFname, " & _
            "STAFFid, " & _
            "STAFFlogin, " & _
            "STAFFpwd, " & _
            "STAFFpost, " & _
            "STAFFstatus, " & _
            "STAFFsysadmin, " & _
            "STAFFadmin, " & _
            "STAFFprint, " & _
            "STAFFtimestamp" & _
        ") VALUES (" & _
            Connection.ToSQL(STAFFname.SQLText, STAFFname.DataType) & ", "↙
     & _
            Connection.ToSQL(STAFFid.SQLText, STAFFid.DataType) & ", " & _
            Connection.ToSQL(STAFFlogin.SQLText, STAFFlogin.DataType) & ",↙
     " & _
            Connection.ToSQL(STAFFpwd.SQLText, STAFFpwd.DataType) & ", " &↙
     _
            Connection.ToSQL(STAFFpost.SQLText, STAFFpost.DataType) & ", "↙
```

```
     & _
            Connection.ToSQL(STAFFstatus.SQLText, STAFFstatus.DataType) & ↙
    ", " & _
            Connection.ToSQL(STAFFsysadmin.SQLText, STAFFsysadmin.        ↙
    DataType) & ", " & _
            Connection.ToSQL(STAFFadmin.SQLText, STAFFadmin.DataType) & ",↙
     " & _
            Connection.ToSQL(STAFFprint.SQLText, STAFFprint.DataType) & ",↙
     " & _
            "CURRENT_TIMESTAMP " & _
        ")"
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteInsert",   ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteInsert", Me)
    End Sub
'End Insert Method

End Class 'End STAFFDataSource Class @2-A61BA892


%>
```

```
<%
'Include Common Files @1-D6EDE132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-6BC89299
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim NewRecord1
Dim STAFF

Redirect = ""
TemplateFileName = "ADMIN_StaffList.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "ADMIN_StaffList.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-912E7083
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set NewRecord1 = new clsRecordNewRecord1
Set STAFF = New clsGridSTAFF
STAFF.Initialize DBConnection1

' Events
%>
<!-- #INCLUDE FILE="ADMIN_StaffList_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-3E59D8AB
NewRecord1.Operation
'End Execute Components

'Go to destination page @1-6D35F4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
```

```
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-AE40C715
NewRecord1.Show Tpl
STAFF.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-60A9CC17
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set NewRecord1 = Nothing
    Set STAFF = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordNewRecord1 'NewRecord1 Class @21-11C1D1C9

'NewRecord1 Variables @21-22BCD617

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button1
    Dim Button2
    Dim Button3
'End NewRecord1 Variables

'NewRecord1 Class_Initialize Event @21-71D98CF6
    Private Sub Class_Initialize()
```

```
        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "NewRecord1")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "NewRecord1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button1 = CCCreateButton("Button1")
            Set Button2 = CCCreateButton("Button2")
            Set Button3 = CCCreateButton("Button3")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End NewRecord1 Class_Initialize Event

'NewRecord1 Class_Terminate Event @21-32BB4709
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End NewRecord1 Class_Terminate Event

'NewRecord1 Validate Method @21-7AE51640
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End NewRecord1 Validate Method

'NewRecord1 Operation Method @21-75363A7B
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button1"
            If Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            ElseIf Not IsEmpty(CCGetParam("Button2", Empty)) Then
                PressedButton = "Button2"
            ElseIf Not IsEmpty(CCGetParam("Button3", Empty)) Then
                PressedButton = "Button3"
            End If
        End If
        Redirect = FileName & "?" & CCGetQueryString("QueryString", Array ↙
    ("ccsForm", "Button1", "Button2", "Button3"))
        If Validate() Then
            If PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_RegisterStaff.asp?" &                ↙
    CCGetQueryString("QueryString", Array("ccsForm", "STAFFid"))

                End If
            ElseIf PressedButton = "Button2" Then
                If NOT Button2.OnClick() Then
                    Redirect = ""
```

```
                Else
                    Redirect = "DataEntry.asp?" & CCGetQueryString(
    "QueryString", Array("ccsForm", "STAFFid"))

                End If
            ElseIf PressedButton = "Button3" Then
                If NOT Button3.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_Menu.asp?" & CCGetQueryString(
    "QueryString", Array("ccsForm", "STAFFid"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End NewRecord1 Operation Method

'NewRecord1 Show Method 621-316F51FA
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.
    ServerVariables("QUERY_STRING"), "ccsForm", "NewRecord1" & IIf
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,
    ccsParseOverwrite, _
            Array(Button1, Button2, Button3))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End NewRecord1 Show Method

End Class 'End NewRecord1 Class 621-A61BA892

Class clsGridSTAFF 'STAFF Class 62-810C77A7

'STAFF Variables 62-9B9CA78B

    ' Public variables
    Public ComponentName, CCSEvents
    Public Visible, Errors
    Public DataSource, PageSize
    Public PageNumber
    Public Command
    Public TemplateBlock
    Public ActiveSorter, SortingDirection
    Public Recordset
```

```
    Private CCSEventResult

    ' Grid Controls
    Public StaticControls, RowControls
    Dim STAFF_TotalRecords
    Dim Sorter_STAFFid
    Dim Sorter_STAFFname
    Dim Sorter_STAFFlogin
    Dim Sorter_STAFFstatus
    Dim STAFFid
    Dim STAFFname
    Dim STAFFlogin
    Dim STAFFstatus
    Dim Navigator
'End STAFF Variables

'STAFF Class_Initialize Event 62-0CF21899
    Private Sub Class_Initialize()
        ComponentName = "STAFF"
        Visible = True
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Errors = New clsErrors
        Set DataSource = New clsSTAFFDataSource
        Set Command = New clsCommand
        PageSize = CCGetParam(ComponentName & "PageSize", Empty)
        If IsNumeric(PageSize) And Len(PageSize) > 0 Then
            If PageSize=0 Then Errors.AddError("(CCS06) Invalid page size.↙
    ")
            If PageSize > 15 Then PageSize = 15
        End If
        If NOT IsNumeric(PageSize) OR IsEmpty(PageSize) Then _
            PageSize = 15 _
        Else _
            PageSize = CInt(PageSize)
        PageNumber = CInt(CCGetParam(ComponentName & "Page", 1))
        ActiveSorter = CCGetParam("STAFFOrder", Empty)
        SortingDirection = CCGetParam("STAFFDir", Empty)
        If NOT(SortingDirection = "ASC" OR SortingDirection = "DESC") Then↙

     _
            SortingDirection = Empty

        Set STAFFid = CCCreateControl(ccsLink, "STAFFid", "STAFFid",      ↙
    ccsText, Empty, CCGetRequestParam("STAFFid", ccsGet))
        Set STAFFname = CCCreateControl(ccsLabel, "STAFFname", "STAFFname"↙
    , ccsText, Empty, CCGetRequestParam("STAFFname", ccsGet))
        Set STAFFlogin = CCCreateControl(ccsLabel, "STAFFlogin",          ↙
    "STAFFlogin", ccsText, Empty, CCGetRequestParam("STAFFlogin", ccsGet))
        Set STAFFstatus = CCCreateControl(ccsLabel, "STAFFstatus",        ↙
    "STAFFstatus", ccsText, Empty, CCGetRequestParam("STAFFstatus",       ↙
    ccsGet))
        Set STAFF_TotalRecords = CCCreateControl(ccsLabel,                ↙
    "STAFF_TotalRecords", "STAFF_TotalRecords", ccsText, Empty,           ↙
    CCGetRequestParam("STAFF_TotalRecords", ccsGet))
        Set Sorter_STAFFid = CCCreateSorter("Sorter_STAFFid", Me,         ↙
    FileName)
        Set Sorter_STAFFname = CCCreateSorter("Sorter_STAFFname", Me,     ↙
    FileName)
        Set Sorter_STAFFlogin = CCCreateSorter("Sorter_STAFFlogin", Me,   ↙
    FileName)
        Set Sorter_STAFFstatus = CCCreateSorter("Sorter_STAFFstatus", Me, ↙
    FileName)
        Set Navigator = CCCreateNavigator(ComponentName, "Navigator",     ↙
    FileName, 10, tpSimple)
    End Sub
'End STAFF Class_Initialize Event

'STAFF Initialize Method 62-2AEA3975
    Sub Initialize(objConnection)
```

```
        If NOT Visible Then Exit Sub

        Set DataSource.Connection = objConnection
        DataSource.PageSize = PageSize
        DataSource.SetOrder ActiveSorter, SortingDirection
        DataSource.AbsolutePage = PageNumber
    End Sub
'End STAFF Initialise Method

'STAFF Class_Terminate Event 62-E1528F98
    Private Sub Class_Terminate()
        Set DataSource = Nothing
        Set Command = Nothing
        Set Errors = Nothing
    End Sub
'End STAFF Class_Terminate Event

'STAFF Show Method 62-74048645
    Sub Show(Tpl)
        If NOT Visible Then Exit Sub

        Dim RecordCounter, ShownRecords
        Dim RowBlock


        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)

        Set TemplateBlock = Tpl.Block("Grid " & ComponentName)
        Set RowBlock = TemplateBlock.Block("Row")
        Set StaticControls = CCCreateCollection(TemplateBlock, Null,        ↙
    ccsParseOverwrite, _
            Array(STAFF_TotalRecords, Sorter_STAFFid, Sorter_STAFFname,     ↙
    Sorter_STAFFlogin, Sorter_STAFFstatus, Navigator))
        Set RowControls = CCCreateCollection(RowBlock, Null,                ↙
    ccsParseAccumulate, _
            Array(STAFFid, STAFFname, STAFFlogin, STAFFstatus))

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If NOT Visible Then Exit Sub

        Errors.AddErrors DataSource.Errors
        If Errors.Count > 0 Then
            TemplateBlock.HTML = CCFormatError("Grid STAFF", Errors)
        Else

            ' Show NoRecords block if no records are found
            If Recordset.EOF Then
                TemplateBlock.Block("NoRecords").Parse ccsParseOverwrite
            End If
            While NOT Recordset.EOF AND ShownRecords < PageSize
                STAFFid.Value = Recordset.Fields("STAFFid")
                STAFFid.Parameters = CCGetQueryString("QueryString", Array↙
    ("ccsForm"))
                STAFFid.Parameters = CCAddParam(STAFFid.Parameters,         ↙
    "STAFFid", Recordset.Fields("STAFFid_param1"))
                STAFFid.Page = "ADMIN_RegisterStaff.asp"
                STAFFname.Value = Recordset.Fields("STAFFname")
                STAFFlogin.Value = Recordset.Fields("STAFFlogin")
                STAFFstatus.Value = Recordset.Fields("STAFFstatus")
                CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShowRow", ↙
    Me)
                RowControls.Show
                Recordset.MoveNext
                ShownRecords = ShownRecords + 1
            Wend

            Navigator.SetDataSource Recordset
```

```
            StaticControls.Show
        End If

    End Sub
'End STAFF Show Method

End Class 'End STAFF Class 62-A61BA892

Class clsSTAFFDataSource 'STAFFDataSource Class 62-EC2F85AE

'DataSource Variables 62-B9DF2177
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STAFFid
    Public STAFFid_param1
    Public STAFFname
    Public STAFFlogin
    Public STAFFstatus
'End DataSource Variables

'DataSource Class_Initialize Event 62-3C89D46F
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STAFFid = CCCreateField("STAFFid", "STAFFid", ccsText, Empty, ↙
    Recordset)
        Set STAFFid_param1 = CCCreateField("STAFFid_param1", "STAFFid",   ↙
    ccsText, Empty, Recordset)
        Set STAFFname = CCCreateField("STAFFname", "STAFFname", ccsText,  ↙
    Empty, Recordset)
        Set STAFFlogin = CCCreateField("STAFFlogin", "STAFFlogin", ccsText↙
    , Empty, Recordset)
        Set STAFFstatus = CCCreateField("STAFFstatus", "STAFFstatus",     ↙
    ccsText, Empty, Recordset)
        Fields.AddFields Array(STAFFid, STAFFid_param1, STAFFname,        ↙
    STAFFlogin, STAFFstatus)
        Orders = Array( _
            Array("Sorter_STAFFid", "STAFFid", ""), _
            Array("Sorter_STAFFname", "STAFFname", ""), _
            Array("Sorter_STAFFlogin", "STAFFlogin", ""), _
            Array("Sorter_STAFFstatus", "STAFFstatus", ""))

        SQL = "SELECT *  " & _
        "FROM STAFF"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM STAFF"
        Where = ""
        Order = "STAFFname"
    End Sub
'End DataSource Class_Initialize Event
```

```
'SetOrder Method @2-68FC9576
    Sub SetOrder(Column, Direction)
        Order = Recordset.GetOrder(Order, Column, Direction, Orders)
    End Sub
'End SetOrder Method

'BuildTableWhere Method @2-98E5A92F
    Public Sub BuildTableWhere()
    End Sub
'End BuildTableWhere Method

'Open Method @2-D03E3B9E
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.CountSQL = CountSQL
        Cmd.Where = Where
        Cmd.OrderBy = Order
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event @2-41B4B062
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End STAFFDataSource Class @2-A61BA892

%>
```

```
<%
'BindEvents Method 61-7E19F76D
Sub BindEvents()
    Set STAFF.STAFF_TotalRecords.CCSEvents("BeforeShow") = GetRef(
    "STAFF_STAFF_TotalRecords_BeforeShow")
End Sub
'End BindEvents Method

Function STAFF_STAFF_TotalRecords_BeforeShow()
    'STAFF_STAFF_TotalRecords_BeforeShow 63-8BD98ADD

'Retrieve number of records 64-11400375
    STAFF.STAFF_TotalRecords.Value = STAFF.DataSource.Recordset.
    RecordCount
'End Retrieve number of records

End Function 'Close STAFF_STAFF_TotalRecords_BeforeShow 63-54C34B28


%>
```

```
<%
'Include Common Files @1-C6EDE132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-39938633
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim NewRecord1
Dim Student
Dim Student1Search
Dim Student1

Redirect = ""
TemplateFileName = "DataEntry.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "DataEntry.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-FB2A654E
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set NewRecord1 = new clsRecordNewRecord1
Set Student = new clsRecordStudent
Set Student1Search = new clsRecordStudent1Search
Set Student1 = New clsGridStudent1
Student.Initialize DBConnection1
Student1.Initialize DBConnection1

' Events
%>
<!-- #INCLUDE FILE="DataEntry_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-A6452828
NewRecord1.Operation
Student.Operation
```

```
Student1Search.Operation
'End Execute Components

'Go to destination page @1-6235F4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-DC06BEE2
NewRecord1.Show Tpl
Student.Show Tpl
Student1Search.Show Tpl
Student1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-6C9E9182
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set NewRecord1 = Nothing
    Set Student = Nothing
    Set Student1Search = Nothing
    Set Student1 = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordNewRecord1 'NewRecord1 Class @10-11C13106

'NewRecord1 Variables @10-F0412140

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
```

```
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button1
    Dim Button2
'End NewRecord1 Variables

'NewRecord1 Class_Initialize Event @10-C37E0C2F
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "NewRecord1")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "NewRecord1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button1 = CCCreateButton("Button1")
            Set Button2 = CCCreateButton("Button2")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End NewRecord1 Class_Initialize Event

'NewRecord1 Class_Terminate Event @10-32B64709
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End NewRecord1 Class_Terminate Event

'NewRecord1 Validate Method @10-7AE52640
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End NewRecord1 Validate Method

'NewRecord1 Operation Method @10-AD21F4BE
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button1"
            If Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            ElseIf Not IsEmpty(CCGetParam("Button2", Empty)) Then
                PressedButton = "Button2"
            End If
        End If
        Redirect = FileName & "?" & CCGetQueryString("QueryString", Array ↙
    ("ccsForm", "Button1", "Button2"))
        If Validate() Then
            If PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
```

```
                    Redirect = "ADMIN_StaffList.asp?" & CCGetQueryString( ↙
    "QueryString", Array("ccsForm", "STDid"))

                End If
            ElseIf PressedButton = "Button2" Then
                If NOT Button2.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "ADMIN_Menu.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm", "STDid"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End NewRecord1 Operation Method

'NewRecord1 Show Method @10-666DA4C9
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.             ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "NewRecord1" & IIf        ↙
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,            ↙
    ccsParseOverwrite, _
            Array(Button1, Button2))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End NewRecord1 Show Method

End Class 'End NewRecord1 Class @10-A61BA4C9

Class clsRecordStudent 'Student Class @1-44D07E1F

'Student Variables @2-B47100E5

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock
```

```
    Public CCSEvents
    Private CCSEventResult

    Public InsertAllowed
    Public UpdateAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim STDid
    Dim chkSTDactive
    Dim Button_Insert
    Dim Button_Update
    Dim Button_Delete
    Dim Button1
'End Student Variables

'Student Class_Initialize Event 62-9B7967DA
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsStudentDataSource
        Set Command = New clsCommand
        InsertAllowed = True
        UpdateAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Student")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Student"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set STDid = CCCreateControl(ccsTextBox, "STDid", "STDid",      ↙
    ccsText, Empty, CCGetRequestParam("STDid", Method))
            STDid.Required = True
            Set chkSTDactive = CCCreateControl(ccsCheckBox, "chkSTDactive"↙
    , "chkSTDactive", ccsBoolean, Array("True", "False", Empty),           ↙
    CCGetRequestParam("chkSTDactive", Method))
            chkSTDactive.CheckedValue = 1
            chkSTDactive.UncheckedValue = 0
            Set Button_Insert = CCCreateButton("Button_Insert")
            Set Button_Update = CCCreateButton("Button_Update")
            Set Button_Delete = CCCreateButton("Button_Delete")
            Set Button1 = CCCreateButton("Button1")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STDid, chkSTDactive)
            If Not FormSubmitted Then
                If IsEmpty(chkSTDactive.State) Then _
                    chkSTDactive.State = IIf(0 = chkSTDactive.CheckedValue↙
    , ccsChecked, ccsUnchecked)
            End If
        End If
    End Sub
'End Student Class_Initialize Event

'Student Initialize Method 62-2179AA22
    Sub Initialize(objConnection)
```

```
        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlSTDid") = CCGetRequestParam("STDid", ccsGET)
        End With
    End Sub
'End Student Initialize Method

'Student Class_Terminate Event @2-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Student Class_Terminate Event

'Student Validate Method @2-7AE5264D
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Student Validate Method

'Student Operation Method @2-407B9A5D
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "Button_Insert"↙
    )
            If Not IsEmpty(CCGetParam("Button_Insert", Empty)) Then
                PressedButton = "Button_Insert"
            ElseIf Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            ElseIf Not IsEmpty(CCGetParam("Button_Delete", Empty)) Then
                PressedButton = "Button_Delete"
            ElseIf Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            End If
        End If
        Redirect = "StudentList.asp?" & CCGetQueryString("QueryString",    ↙
    Array("ccsForm", "Button_Insert", "Button_Update", "Button_Delete",    ↙
    "Button1"))
        If PressedButton = "Button_Delete" Then
            If NOT Button_Delete.OnClick Then
                Redirect = ""
            Else
                Redirect = "DataEntry.asp?" & CCGetQueryString(            ↙
    "QueryString", Array("ccsForm", "STDid", "s_STDid"))
            End If
        ElseIf PressedButton = "Button1" Then
            If NOT Button1.OnClick Then
                Redirect = ""
            Else
                Redirect = "DataEntry.asp?" & CCGetQueryString(            ↙
    "QueryString", Array("ccsForm", "STDid", "s_STDid"))
            End If
        ElseIf Validate() Then
            If PressedButton = "Button_Insert" Then
                If NOT Button_Insert.OnClick() OR NOT InsertRow() Then
                    Redirect = ""
                Else
                    Redirect = "DataEntry.asp?" & CCGetQueryString(        ↙
    "QueryString", Array("ccsForm", "STDid", "s_STDid"))

                End If
```

```
            ElseIf PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
                    Redirect = "DataEntry.asp?" & CCGetQueryString(
    "QueryString", Array("ccsForm", "STDid", "s_STDid"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Student Operation Method

'Student InsertRow Method 62-F8EFFC20
    Function InsertRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeInsert", Me)
        If NOT InsertAllowed Then InsertRow = False : Exit Function
        DataSource.STDid.Value = STDid.Value
        'DataSource.STDname.Value = STDname.Value
        'DataSource.STDpwd.Value = STDid.Value '<AE
        DataSource.Insert(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInsert", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Insert Operation"
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        InsertRow = (Errors.Count = 0)
    End Function
'End Student InsertRow Method

'Student UpdateRow Method 62-2F071735
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.STDid.Value = STDid.Value
        'DataSource.STDname.Value = STDname.Value
        'DataSource.STDpwd.Value = STDid.Value '<AE
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End Student UpdateRow Method

'Student Show Method 62-0D3BB8F8
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
```

```
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.
    ServerVariables("QUERY_STRING"), "ccsForm", "Student" & IIf(EditMode,
    ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,
    ccsParseOverwrite, _
            Array(STDid, chkSTDactive, Button_Insert, Button_Update,
    Button_Delete, Button1))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record Student", DataSource.Connection.
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        STDid.Value = Recordset.Fields("STDid")
                        chkSTDactive.Value = Recordset.Fields(
    "chkSTDactive")
                    End If
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Insert.Visible = NOT EditMode AND InsertAllowed
        Button_Update.Visible = EditMode AND UpdateAllowed
        Button_Delete.Visible = False

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Student Show Method

End Class 'End Student Class 62-A61BA892

Class clsStudentDataSource 'StudentDataSource Class 62-2F9B50BE

'DataSource Variables 62-E92936AE
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult
```

```
    ' Datasource fields
    Public STDid
    Public STDname
    Public STDpwd '<%E
    Public chkSTDactive
'End DataSource Variables

'DataSource Class_Initialize Event 02-7F75932F
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,
    Recordset)
        Set chkSTDactive = CCCreateField("chkSTDactive", "STDactive",
    ccsBoolean, Array(1, 0, Empty), Recordset)
        Fields.AddFields Array(STDid, chkSTDactive)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method 02-7E709E1C
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlSTDid", ccsText, Empty, Empty, Empty,
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STDid", .
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 02-2F951BED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
```

```
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event @2-41B4B08D
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Update Method @2-5BA478A7
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.       ↙
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE Student SET " & _
            "STDid=" & Connection.ToSQL(STDid.SQLText, STDid.DataType) & "↙
    , " & _
            "STDpwd=" & Connection.ToSQL(STDid.SQLtext, STDid.DataType) & ↙
    ", " & _
            "STDactive=" & Student.chkSTDactive.value & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",    ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

'Insert Method @2-DB8E050F
    Sub Insert(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildInsert", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        Cmd.SQL = "INSERT INTO Student(" & _
            "STDid, " & _
            "STDpwd, " & _
```

```
            "STDactive" & _
        ") VALUES (" & _
            Connection.ToSQL(STDid.SQLText, STDid.DataType) & ", " & _
            Connection.ToSQL(STDid.SQLText, STDid.DataType) & ", 1" & _
        ")"
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteInsert",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteInsert", Me)
    End Sub
'End Insert Method

End Class 'End StudentDataSource Class @2-A61BA692

Class clsRecordStudent1Search 'Student1Search Class @43-7A82DD12

'Student1Search Variables @43-DAE7BC26

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim s_STDid
    Dim Button_DoSearch
'End Student1Search Variables

'Student1Search Class_Initialize Event @43-6A792192
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Student1Search")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Student1Search"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set s_STDid = CCCreateControl(ccsTextBox, "s_STDid", "s_STDid"
    , ccsText, Empty, CCGetRequestParam("s_STDid", Method))
            Set Button_DoSearch = CCCreateButton("Button_DoSearch")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(s_STDid)
        End If
    End Sub
```

```
'End Student1Search Class_Initialize Event

'Student1Search Class_Terminate Event 643-31884709
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Student1Search Class_Terminate Event

'Student1Search Validate Method 643-7AE52643
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Student1Search Validate Method

'Student1Search Operation Method 643-69836CB6
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoSearch"
            If Not IsEmpty(CCGetParam("Button_DoSearch", Empty)) Then
                PressedButton = "Button_DoSearch"
            End If
        End If
        Redirect = "DataEntry.asp"
        If Validate() Then
            If PressedButton = "Button_DoSearch" Then
                If NOT Button_DoSearch.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "DataEntry.asp?" & CCGetQueryString("Form",↙
     Array("Button_DoSearch"))
                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Student1Search Operation Method

'Student1Search Show Method 643-EC0FB1C2
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.            ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Student1Search" & IIf    ↙
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,           ↙
    ccsParseOverwrite, _
            Array(s_STDid, Button_DoSearch))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
```

```
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Student1Search Show Method

End Class 'End Student1Search Class @43-A61BA892

Class clsGridStudent1 'Student1 Class @18-4BAA6B1D

'Student1 Variables @18-7A4BB822

    ' Public variables
    Public ComponentName, CCSEvents
    Public Visible, Errors
    Public DataSource, PageSize
    Public PageNumber
    Public Command
    Public TemplateBlock
    Public ActiveSorter, SortingDirection
    Public Recordset

    Private CCSEventResult

    ' Grid Controls
    Public StaticControls, RowControls
    Dim Student1_TotalRecords
    Dim Sorter_STDid
    Dim Sorter_STDstampdate
    Dim STDid
    Dim STDstampdate
    Dim CheckBox1
    Dim Navigator
'End Student1 Variables

'Student1 Class_Initialize Event @18-07C11A95
    Private Sub Class_Initialize()
        ComponentName = "Student1"
        Visible = True
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Errors = New clsErrors
        Set DataSource = New clsStudent1DataSource
        Set Command = New clsCommand
        PageSize = CCGetParam(ComponentName & "PageSize", Empty)
        If IsNumeric(PageSize) And Len(PageSize) > 0 Then
            If PageSize=0 Then Errors.AddError("(CCS06) Invalid page size.↙
    ")
            If PageSize > 50 Then PageSize = 50
        End If
        If NOT IsNumeric(PageSize) OR IsEmpty(PageSize) Then _
            PageSize = 50 _
        Else _
            PageSize = CInt(PageSize)
        PageNumber = CInt(CCGetParam(ComponentName & "Page", 1))
        ActiveSorter = CCGetParam("Student1Order", Empty)
        SortingDirection = CCGetParam("Student1Dir", Empty)
        If NOT(SortingDirection = "ASC" OR SortingDirection = "DESC") Then↙

     _
            SortingDirection = Empty

        Set STDid = CCCreateControl(ccsLink, "STDid", "STDid", ccsText,    ↙
    Empty, CCGetRequestParam("STDid", ccsGet))
        Set STDstampdate = CCCreateControl(ccsLabel, "STDstampdate",       ↙
    "STDstampdate", ccsDate, Array("dd", "/", "mmm", "/", "yyyy"),         ↙
    CCGetRequestParam("STDstampdate", ccsGet))
        Set CheckBox1 = CCCreateControl(ccsCheckBox, "CheckBox1",          ↙
    "CheckBox1", ccsBoolean, Array("True", "False", Empty),                ↙
```

```
    CCGetRequestParam("CheckBox1", ccsGet))
        CheckBox1.CheckedValue = True
        CheckBox1.UncheckedValue = False
        Set Student1_TotalRecords = CCCreateControl(ccsLabel,
    "Student1_TotalRecords", "Student1_TotalRecords", ccsText, Empty,
    CCGetRequestParam("Student1_TotalRecords", ccsGet))
        Set Sorter_STDid = CCCreateSorter("Sorter_STDid", Me, FileName)
        Set Sorter_STDstampdate = CCCreateSorter("Sorter_STDstampdate", Me
    , FileName)
        Set Navigator = CCCreateNavigator(ComponentName, "Navigator",
    FileName, 10, tpSimple)
    End Sub
'End Student1 Class_Initialize Event

'Student1 Initialize Method {18-2AEA3975
    Sub Initialize(objConnection)
        If NOT Visible Then Exit Sub

        Set DataSource.Connection = objConnection
        DataSource.PageSize = PageSize
        DataSource.SetOrder ActiveSorter, SortingDirection
        DataSource.AbsolutePage = PageNumber
    End Sub
'End Student1 Initialize Method

'Student1 Class_Terminate Event {18-E1528F98
    Private Sub Class_Terminate()
        Set DataSource = Nothing
        Set Command = Nothing
        Set Errors = Nothing
    End Sub
'End Student1 Class_Terminate Event

'Student1 Show Method {18-C9CACE71
    Sub Show(Tpl)
        If NOT Visible Then Exit Sub

        Dim RecordCounter, ShownRecords
        Dim RowBlock

        With DataSource
            .Parameters("urls_STDid") = CCGetRequestParam("s_STDid",
    ccsGET)
            .Parameters("urls_STDactive") = CCGetRequestParam("s_STDactive
    ", ccsGET)
        End With

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)

        Set TemplateBlock = Tpl.Block("Grid " & ComponentName)
        Set RowBlock = TemplateBlock.Block("Row")
        Set StaticControls = CCCreateCollection(TemplateBlock, Null,
    ccsParseOverwrite, _
            Array(Student1_TotalRecords, Sorter_STDid, Sorter_STDstampdate
    , Navigator))
        Set RowControls = CCCreateCollection(RowBlock, Null,
    ccsParseAccumulate, _
            Array(STDid, STDstampdate, CheckBox1))

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If NOT Visible Then Exit Sub

        Errors.AddErrors DataSource.Errors
        If Errors.Count > 0 Then
            TemplateBlock.HTML = CCFormatError("Grid Student1", Errors)
        Else
```

```
            ' Show NoRecords block if no records are found
            If Recordset.EOF Then
                TemplateBlock.Block("NoRecords").Parse ccsParseOverwrite
            End If
            While NOT Recordset.EOF AND ShownRecords < PageSize
                STDid.Value = Recordset.Fields("STDid")
                STDid.Parameters = CCGetQueryString("QueryString", Array( ↙
    "ccsForm"))
                STDid.Parameters = CCAddParam(STDid.Parameters, "STDid",  ↙
    Recordset.Fields("STDid_param1"))
                STDid.Page = "DataEntry.asp"
                STDstampdate.Value = Recordset.Fields("STDstampdate")
                CheckBox1.Value = Recordset.Fields("CheckBox1")
                CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShowRow", ↙
    Me)
                RowControls.Show
                Recordset.MoveNext
                ShownRecords = ShownRecords + 1
            Wend

            Navigator.SetDataSource Recordset
            StaticControls.Show
        End If

    End Sub
'End Student1 Show Method

End Class 'End Student1 Class @18-A61BA892

Class clsStudent1DataSource 'Student1DataSource Class @18-879CDB23

'DataSource Variables @18-AC6BDAAF
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STDid
    Public STDid_param1
    Public STDstampdate
    Public CheckBox1
'End DataSource Variables

'DataSource Class_Initialize Event @18-1D46F517
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,       ↙
    Recordset)
        Set STDid_param1 = CCCreateField("STDid_param1", "STDid", ccsText,↙
     Empty, Recordset)
        Set STDstampdate = CCCreateField("STDstampdate", "STDstampdate",  ↙
    ccsDate, Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":"↙
```

```
    , "ss"), Recordset)
        Set CheckBox1 = CCCreateField("CheckBox1", "STDactive", ccsBoolean↙
    , Array(1, 0, Empty), Recordset)
        Fields.AddFields Array(STDid, STDid_param1, STDstampdate,          ↙
    CheckBox1)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing
        Orders = Array( _
            Array("Sorter_STDid", "STDid", ""), _
            Array("Sorter_STDstampdate", "STDstampdate", ""))

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
        Order = "STDid"
    End Sub
'End DataSource Class_Initialize Event

'SetOrder Method 618-68FC9576
    Sub SetOrder(Column, Direction)
        Order = Recordset.GetOrder(Order, Column, Direction, Orders)
    End Sub
'End SetOrder Method

'BuildTableWhere Method 618-3A01178A
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urls_STDid", ccsText, Empty, Empty, Empty,  ↙
    False
            .AddParameter 2, "urls_STDactive", ccsBoolean, Array("True",  ↙
    "False", Empty), Array(1, 0, Empty), Empty, False
            .Criterion(1) = .Operation(opContains, False, "STDid", .      ↙
    getParamByID(1))
            .Criterion(2) = .Operation(opEqual, False, "STDactive", .     ↙
    getParamByID(2))
            .AssembledWhere = .opAND(False, .Criterion(1), .Criterion(2))
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 618-F1D016EC
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
```

```
        Cmd.SQL = SQL
        Cmd.CountSQL = CountSQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event 618-41B4B08D
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End Student1DataSource Class 618-A61BA892


%>
```

```
<%
'BindEvents Method @1-F700F72B
Sub BindEvents()
    Set Student1.Student1_TotalRecords.CCSEvents("BeforeShow") = GetRef(
    "Student1_Student1_TotalRecords_BeforeShow")
End Sub
'End BindEvents Method

Function Student1_Student1_TotalRecords_BeforeShow()
    'Student1_Student1_TotalRecords_BeforeShow @19-B964F650

'Retrieve number of records @20-24E81A3A
    Student1.Student1_TotalRecords.Value = Student1.DataSource.Recordset.
    RecordCount
'End Retrieve number of records

End Function 'Close Student1_Student1_TotalRecords_BeforeShow @19-54C34B28


%>
```

```
<%
'Include Common Files @1-2BCCF1A2
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Server.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-05374323
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Page controls
Dim Login

Redirect = ""
TemplateFileName = "STAFF_Login.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STAFF_Login.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-A0177FAE

' Controls
Set Login = new clsRecordLogin

' Events
%>
<!-- #INCLUDE FILE="STAFF_Login_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-6AA0CED0
Login.Operation
'End Execute Components

'Go to destination page @1-63B5B4F0
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-015DE10E
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-C559F43E
Login.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-DE8F79E6
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    Set Login = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordLogin 'Login Class @37-C3C09674

'Login Variables @37-55290F3F

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim login
    Dim password
    Dim Button_DoLogin
'End Login Variables

'Login Class_Initialize Event @37-AB01F778
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Login")
            End If
            If UBound(OperationMode) > 0 Then
```

```
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Login"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set login = CCCreateControl(ccsTextBox, "login", "login",
    ccsText, Empty, CCGetRequestParam("login", Method))
            login.Required = True
            Set password = CCCreateControl(ccsTextBox, "password",
    "password", ccsText, Empty, CCGetRequestParam("password", Method))
            password.Required = True
            Set Button_DoLogin = CCCreateButton("Button_DoLogin")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(login, password)
        End If
    End Sub
'End Login_Class_Initialize Event

'Login_Class_Terminate Event @37-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Login_Class_Terminate Event

'Login_Validate Method @37-7AE52643
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Login_Validate Method

'Login_Operation Method @37-F66A0CEE
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoLogin"
            If Not IsEmpty(CCGetParam("Button_DoLogin", Empty)) Then
                PressedButton = "Button_DoLogin"
            End If
        End If
        Redirect = FileName & ""
        If Validate() Then
            If PressedButton = "Button_DoLogin" Then
                If NOT Button_DoLogin.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_options.asp?" & CCGetQueryString(
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Login_Operation Method

'Login_Show Method @37-C4058633
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.
    ServerVariables("QUERY_STRING"), "ccsForm", "Login" & IIf(EditMode, ":
    Edit", ""))
```

```
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,            ↙
    ccsParseOverwrite, _
            Array(login, password, Button_DoLogin))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Login Show Method

End Class 'End Login Class 637-A61BA892


%>
```

```
<%
'BindEvents Method @1-2261B678
Sub BindEvents()
    Set Login.Button_DoLogin.CCSEvents("OnClick") = GetRef(
    "Login_Button_DoLogin_OnClick")
End Sub
'End BindEvents Method

Function Login_Button_DoLogin_OnClick() 'Login_Button_DoLogin_OnClick @38-
    ADB4691A

'Login @39-AA430BB1
    With Login
        If NOT CCLoginUser(.login.Value, .password.Value) Then
            .Errors.addError("Login or Password is incorrect.")
            Login_Button_DoLogin_OnClick = False
            .password.Value = ""
        Else
            If Not IsEmpty(CCGetParam("ret_link", Empty)) Then _
                Redirect = CCGetParam("ret_link", Empty)
            Login_Button_DoLogin_OnClick = True
        End If
    End With
'End Login

End Function 'Close Login_Button_DoLogin_OnClick @38-54C94B28


%>
```

```
<%
'Include Common Files {1-D6E2C1A}
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Scanner.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page {1-0251F124
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim NewRecord1

Redirect = ""
TemplateFileName = "STAFF_Logout.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STAFF_Logout.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects {1-B9D9A0B3
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set NewRecord1 = new clsRecordNewRecord1
NewRecord1.Initialize DBConnection1

' Events
%>
<!-- #INCLUDE FILE="STAFF_Logout_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components {1-9E5938AB
NewRecord1.Operation
'End Execute Components

'Go to destination page {1-6D3B04A
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page
```

```
'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-64642188
NewRecord1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-B3A76CC6
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set NewRecord1 = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordNewRecord1 'NewRecord1 Class @3-11C1D1C6

'NewRecord1 Variables @3-AFF52432

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public UpdateAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button_Update
'End NewRecord1 Variables

'NewRecord1 Class_Initialize Event @3-[illegible]
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsNewRecord1DataSource
```

```
        Set Command = New clsCommand
        UpdateAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "NewRecord1")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "NewRecord1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button_Update = CCCreateButton("Button_Update")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End NewRecord1 Class_Initialize Event

'NewRecord1 Initialize Method @3-36D27969
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
        End With
    End Sub
'End NewRecord1 Initialize Method

'Newkecord1 Class_Terminate Event @3-32B547C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End NewRecord1 Class_Terminate Event

'NewRecord1 Validate Method @3-7AE92E4D
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End NewRecord1 Validate Method

'NewRecord1 Operation Method @3-AF8CCB4F
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "Button_Update"↙
    )
            If Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            End If
        End If
        Redirect = FileName & "?" & CCGetQueryString("QueryString", Array ↙
    ("ccsForm", "Button_Update"))
        If Validate() Then
            If PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_Login.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm"))
```

```
                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End NewRecord1 Operation Method

'NewRecord1 UpdateRow Method 63-81504195
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End NewRecord1 UpdateRow Method

'NewRecord1 Show Method 63-C7411A06
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "NewRecord1" & IIf           ↙
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(Button_Update))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record NewRecord1", DataSource.            ↙
    Connection.LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                    End If
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
```

```
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End NewRecord1 Show Method

End Class 'End NewRecord1 Class @3-A61BA892

Class clsNewRecord1DataSource 'NewRecord1DataSource Class @3-C2F1314B

'DataSource Variables @3-1E4E7B43
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

'End DataSource Variables

'DataSource Class_Initialize Event @3-C0276E7E
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True

        SQL = "SELECT *   " & _
        "FROM STAFF"
        CountSQL = "SELECT COUNT(*)   " & _
        "FROM STAFF"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method @3-9EB5A92F
    Public Sub BuildTableWhere()
    End Sub
'End BuildTableWhere Method

'Open Method @3-B139FD2D
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.Where = Where
```

```
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {3-41B4B08D
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Update Method {3-44E42957
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE STAFF SET " & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

End Class 'End NewRecordsetDataSource Class {3-A61BA69E


%>
```

```
<%
'BindEvents Method @1-00A33F24
Sub BindEvents()
    Set CCSEvents("AfterInitialize") = GetRef("Page_AfterInitialize")
End Sub
'End BindEvents Method

Function Page_AfterInitialize() 'Page_AfterInitialize @1-1E51C15

'Logout @2-20558F71
    If NOT IsEmpty(CCGetParam("Logout", Empty)) Then
    CCLogoutUser
    Redirect = "STAFF_Logout.asp"
    End If
'End Logout

End Function 'Close Page_AfterInitialize @1-54C34B28


%>
```

```
<%
'Include Common Files @1-2BCBD132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Scanner.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-78E96A7A
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim STAFF

Redirect = ""
TemplateFileName = "STAFF_pwd.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STAFF_pwd.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-94100C1B2
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set STAFF = new clsRecordSTAFF
STAFF.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-2BC543BB
STAFF.Operation
'End Execute Components

'Go to destination page @1-6D35F4F0
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Templates

'Show Page @1-1C138EB0
STAFF.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-001CD4BE
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set STAFF = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordSTAFF 'STAFF Class @2-02A4242

'STAFF Variables @2-E3AC1B75

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public UpdateAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim STAFFid
    Dim STAFFname
    Dim STAFFlogin
    Dim STAFFpwd
    Dim Button_Update
    Dim Button1
'End STAFF Variables

'STAFF Class Initialize Event @2-60B43B1E
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsSTAFFDataSource
        Set Command = New clsCommand
```

```
        UpdateAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "STAFF")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "STAFF"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set STAFFid = CCCreateControl(ccsLabel, "STAFFid", "Id",
    ccsText, Empty, CCGetRequestParam("STAFFid", Method))
            Set STAFFname = CCCreateControl(ccsLabel, "STAFFname", "Name",
     ccsText, Empty, CCGetRequestParam("STAFFname", Method))
            Set STAFFlogin = CCCreateControl(ccsLabel, "STAFFlogin",
    "Login", ccsText, Empty, CCGetRequestParam("STAFFlogin", Method))
            Set STAFFpwd = CCCreateControl(ccsTextBox, "STAFFpwd", "Pwd",
    ccsText, Empty, CCGetRequestParam("STAFFpwd", Method))
            Set Button_Update = CCCreateButton("Button_Update")
            Set Button1 = CCCreateButton("Button1")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STAFFpwd)
        End If
    End Sub
'End STAFF Class_Initialize Event

'STAFF Initialize Method {2-0EB95DAE
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("sesUserID") = Session("UserID")
        End With
    End Sub
'End STAFF Initialize Method

'STAFF Class_Terminate Event {2-32B84709
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End STAFF Class_Terminate Event

'STAFF Validate Method {2-7AE5264C
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End STAFF Validate Method

'STAFF Operation Method {2-11C68263
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "Button1")
            If Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            ElseIf Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
```

```
            End If
        End If
        Redirect = "STAFF_pwd.asp?" & CCGetQueryString("QueryString",      ↙
    Array("ccsForm", "Button_Update", "Button1"))
        If Validate() Then
            If PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_options.asp?" & CCGetQueryString(    ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_options.asp?" & CCGetQueryString(    ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End STAFF Operation Method

'STAFF UpdateRow Method 62-1870092E
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.STAFFpwd.Value = STAFFpwd.Value
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End STAFF UpdateRow Method

'STAFF Show Method 62-10FA1566
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.              ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "STAFF" & IIf(EditMode, ":↙
    Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,             ↙
    ccsParseOverwrite, _
```

```
            Array(STAFFid, STAFFname, STAFFlogin, STAFFpwd, Button_Update,↙
     Button1))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record STAFF", DataSource.Connection.   ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        STAFFpwd.Value = Recordset.Fields("STAFFpwd")
                    End If
                    STAFFid.Value = Recordset.Fields("STAFFid")
                    STAFFname.Value = Recordset.Fields("STAFFname")
                    STAFFlogin.Value = Recordset.Fields("STAFFlogin")
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Update.Visible = EditMode AND UpdateAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End STAFF Show Method

End Class 'End STAFF Class @2-A61BA892

Class clsSTAFFDataSource 'STAFFDataSource Class @1-C7[illegible]

'DataSource Variables @1-70A06A6B
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STAFFid
    Public STAFFname
    Public STAFFlogin
    Public STAFFpwd
'End DataSource Variables

'DataSource Class_Initialize Event @2-2663B06B
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
```

```
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STAFFid = CCCreateField("STAFFid", "STAFFid", ccsText, Empty, ↙
    Recordset)
        Set STAFFname = CCCreateField("STAFFname", "STAFFname", ccsText,  ↙
    Empty, Recordset)
        Set STAFFlogin = CCCreateField("STAFFlogin", "STAFFlogin", ccsText↙
    , Empty, Recordset)
        Set STAFFpwd = CCCreateField("STAFFpwd", "STAFFpwd", ccsText,     ↙
    Empty, Recordset)
        Fields.AddFields Array(STAFFid, STAFFname, STAFFlogin, STAFFpwd)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM STAFF"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM STAFF"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method {2-4BE08281
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "sesUserID", ccsText, Empty, Empty, Empty,   ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STAFFid", .       ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method {2-DF951BED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
```

```
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event 02-41B4B0E0
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Update Method 30-E1C7EBFF
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE STAFF SET " & _
            "STAFFpwd=" & Connection.ToSQL(STAFFpwd.SQLText, STAFFpwd.
    DataType) & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

End Class 'End STAFFDataSource Class 30-A61BA5W1


%>
```

```
<%
'Include Common Files @1-D6BDF132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-9E6886A7
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Page controls
Dim Label_username
Dim NewRecord1

Redirect = ""
TemplateFileName = "STAFF_options.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STAFF_options.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-F46DF6FB

' Controls
Set Label_username = CCCreateControl(ccsLabel, "Label_username",          ↙
    "Label_username", ccsText, Empty, CCGetRequestParam("Label_username", ↙
    ccsGet))
Set NewRecord1 = new clsRecordNewRecord1
Label_username.Value = Session("UserLogin")

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-3E593CAB
NewRecord1.Operation
'End Execute Components

'Go to destination page @1-6D35F4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
```

```
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Set Parse HTML Template

'Show Page 01-1ADF2D8E
Label_username.Show Tpl
NewRecord1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page 01-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub 01-AB0AA71B
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    Set NewRecord1 = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordNewRecord1 'NewRecord1 Class 012-11C1D106

'NewRecord1 Variables 012-EC451D98

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button1
    Dim Button4
    Dim Button2
    Dim Button3
'End NewRecord1 Variables

'NewRecord1 Class_Initialize Event 012-2C14EFE1
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "NewRecord1")
            End If
```

```
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "NewRecord1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button1 = CCCreateButton("Button1")
            Set Button4 = CCCreateButton("Button4")
            Set Button2 = CCCreateButton("Button2")
            Set Button3 = CCCreateButton("Button3")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End NewRecord1 Class_Initialize Event

'NewRecord1 Class_Terminate Event 612-32B84709
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End NewRecord1 Class_Terminate Event

'NewRecord1 Validate Method 612-7AE52643
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End NewRecord1 Validate Method

'NewRecord1 Operation Method 612-71902305
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button1"
            If Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            ElseIf Not IsEmpty(CCGetParam("Button4", Empty)) Then
                PressedButton = "Button4"
            ElseIf Not IsEmpty(CCGetParam("Button2", Empty)) Then
                PressedButton = "Button2"
            ElseIf Not IsEmpty(CCGetParam("Button3", Empty)) Then
                PressedButton = "Button3"
            End If
        End If
        Redirect = FileName & "?" & CCGetQueryString("QueryString", Array ↙
    ("ccsForm", "Button1", "Button4", "Button2", "Button3"))
        If Validate() Then
            If PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_List.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button4" Then
                If NOT Button4.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_StdList.asp?" & CCGetQueryString(   ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button2" Then
                If NOT Button2.OnClick() Then
                    Redirect = ""
                Else
```

```
                    Redirect = "STAFF_pwd.asp?" & CCGetQueryString(
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button3" Then
                If NOT Button3.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_Logout.asp?" & CCGetQueryString(
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End NewRecord1 Operation Method

'NewRecord1 Show Method @12-74B72E7D
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.
    ServerVariables("QUERY_STRING"), "ccsForm", "NewRecord1" & IIf
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,
    ccsParseOverwrite, _
            Array(Button1, Button4, Button2, Button3))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End NewRecord1 Show Method

End Class 'End NewRecord1 Class @12-A61BA892


%>
```

```
<%
'Include Common Files @1-3BEDD1A5
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-5B105822
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim NewRecord1
Dim StudentSearch
Dim Student

Redirect = ""
TemplateFileName = "STAFF_StdList.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STAFF_StdList.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-85A3F3D5
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set NewRecord1 = new clsRecordNewRecord1
Set StudentSearch = new clsRecordStudentSearch
Set Student = New clsGridStudent
Student.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-0190E3F9
NewRecord1.Operation
StudentSearch.Operation
'End Execute Components

'Go to destination page @1-6235F4F3
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page
```

```
'Initialize HTML Template 01-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page 01-3263165F
NewRecord1.Show Tpl
StudentSearch.Show Tpl
Student.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page 01-CB810362
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub 01-026FE4B1
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set NewRecord1 = Nothing
    Set StudentSearch = Nothing
    Set Student = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordNewRecord1 'NewRecord1 Class 024-11C10126

'NewRecord1 Variables 024-F0AA16FB

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button2
    Dim Button1
'End NewRecord1 Variables

'NewRecord1 Class Initialize Event 024-10A932B1
    Private Sub Class_Initialize()

        Visible = True
```

```
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "NewRecord1")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "NewRecord1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button2 = CCCreateButton("Button2")
            Set Button1 = CCCreateButton("Button1")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End NewRecord1 Class_Initialize Event

'NewRecord1 Class_Terminate Event 624-32B84729
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End NewRecord1 Class_Terminate Event

'NewRecord1 Validate Method 624-7AE52643
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End NewRecord1 Validate Method

'NewRecord1 Operation Method 624-F9AB1009
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button2"
            If Not IsEmpty(CCGetParam("Button2", Empty)) Then
                PressedButton = "Button2"
            ElseIf Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            End If
        End If
        Redirect = FileName & "?" & CCGetQueryString("QueryString", Array ↙
    ("ccsForm", "Button2", "Button1"))
        If Validate() Then
            If PressedButton = "Button2" Then
                If NOT Button2.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_List.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_options.asp?" & CCGetQueryString(   ↙
    "QueryString", Array("ccsForm"))

                End If
```

```
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End NewRecord1 Operation Method

'NewRecord1 Show Method 624-9DA3202F
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "NewRecord1" & IIf          ↙
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,              ↙
    ccsParseOverwrite, _
            Array(Button2, Button1))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End NewRecord1 Show Method

End Class 'End NewRecord1 Class 624-A91BA89C

Class clsRecordStudentSearch 'StudentSearch Class 63-C9B95954

'StudentSearch Variables 63-1B9CEF23

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim s_STDid
    Dim s_STDname
    Dim s_STDadmissionNum
```

```
    Dim s_AdmYear
    Dim Button_DoSearch
'End StudentSearch Variables

'StudentSearch Class_Initialize Event {3-2884AA3D}
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "StudentSearch")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "StudentSearch"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set s_STDid = CCCreateControl(ccsTextBox, "s_STDid", "s_STDid"↙
    , ccsText, Empty, CCGetRequestParam("s_STDid", Method))
            Set s_STDname = CCCreateControl(ccsTextBox, "s_STDname",      ↙
    "s_STDname", ccsText, Empty, CCGetRequestParam("s_STDname", Method))
            Set s_STDadmissionNum = CCCreateControl(ccsTextBox,           ↙
    "s_STDadmissionNum", "s_STDadmissionNum", ccsFloat, Empty,            ↙
    CCGetRequestParam("s_STDadmissionNum", Method))
            Set s_AdmYear = CCCreateControl(ccsTextBox, "s_AdmYear",      ↙
    "s_AdmYear", ccsInteger, Empty, CCGetRequestParam("s_AdmYear",        ↙
    Method))
            Set Button_DoSearch = CCCreateButton("Button_DoSearch")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(s_STDid, s_STDname,      ↙
    s_STDadmissionNum, s_AdmYear)
        End If
    End Sub
'End StudentSearch Class_Initialize Event

'StudentSearch Class_Terminate Event {3-2DB4F7D9}
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End StudentSearch Class_Terminate Event

'StudentSearch Validate Method {3-7AE51E42}
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End StudentSearch Validate Method

'StudentSearch Operation Method {3-6EF96A61}
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoSearch"
            If Not IsEmpty(CCGetParam("Button_DoSearch", Empty)) Then
                PressedButton = "Button_DoSearch"
            End If
        End If
        Redirect = "STAFF_StdList.asp"
        If Validate() Then
```

```
            If PressedButton = "Button_DoSearch" Then
                If NOT Button_DoSearch.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_StdList.asp?" & CCGetQueryString(    ↙
    "Form", Array("Button_DoSearch"))
                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End StudentSearch Operation Method

'StudentSearch Show Method 63-51AAAF4A
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.              ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "StudentSearch" & IIf      ↙
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,             ↙
    ccsParseOverwrite, _
            Array(s_STDid, s_STDname, s_STDadmissionNum, s_AdmYear,        ↙
    Button_DoSearch))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End StudentSearch Show Method

End Class 'End StudentSearch Class 63-A61BA498

Class clsGridStudent 'Student Class 62-A1C52A32

'Student Variables 62-75022080

    ' Public variables
    Public ComponentName, CCSEvents
    Public Visible, Errors
    Public DataSource, PageSize
    Public PageNumber
    Public Command
    Public TemplateBlock
    Public ActiveSorter, SortingDirection
    Public Recordset

    Private CCSEventResult

    ' Grid Controls
    Public StaticControls, RowControls
    Dim Sorter_STDadmissionNum
```

```
    Dim Sorter_STDid
    Dim Sorter_STDname
    Dim Sorter_STDsex
    Dim STDadmissionNum
    Dim Label1
    Dim STDid
    Dim STDname
    Dim STDsex
    Dim Navigator
'End Student Variables

'Student Class_Initialize Event 62-9AE64359
    Private Sub Class_Initialize()
        ComponentName = "Student"
        Visible = True
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Errors = New clsErrors
        Set DataSource = New clsStudentDataSource
        Set Command = New clsCommand
        PageSize = CCGetParam(ComponentName & "PageSize", Empty)
        If IsNumeric(PageSize) And Len(PageSize) > 0 Then
            If PageSize=0 Then Errors.AddError("(CCS06) Invalid page size.↙
    ")
            If PageSize > 50 Then PageSize = 50
        End If
        If NOT IsNumeric(PageSize) OR IsEmpty(PageSize) Then _
            PageSize = 50 _
        Else _
            PageSize = CInt(PageSize)
        PageNumber = CInt(CCGetParam(ComponentName & "Page", 1))
        ActiveSorter = CCGetParam("StudentOrder", Empty)
        SortingDirection = CCGetParam("StudentDir", Empty)
        If NOT(SortingDirection = "ASC" OR SortingDirection = "DESC") Then↙

     _
            SortingDirection = Empty

        Set STDadmissionNum = CCCreateControl(ccsLabel, "STDadmissionNum",↙
     "STDadmissionNum", ccsFloat, Array(True, False, "", False, "", Array ↙
    ("0", "0", "0", "0", "0", "0"), Empty, 1, True, "", False),          ↙
    CCGetRequestParam("STDadmissionNum", ccsGet))
        Set Label1 = CCCreateControl(ccsLabel, "Label1", "Label1", ccsDate↙
    , Array("yy"), CCGetRequestParam("Label1", ccsGet))
        Set STDid = CCCreateControl(ccsLink, "STDid", "STDid", ccsText,   ↙
    Empty, CCGetRequestParam("STDid", ccsGet))
        Set STDname = CCCreateControl(ccsLabel, "STDname", "STDname",     ↙
    ccsText, Empty, CCGetRequestParam("STDname", ccsGet))
        Set STDsex = CCCreateControl(ccsLabel, "STDsex", "STDsex", ccsText↙
    , Empty, CCGetRequestParam("STDsex", ccsGet))
        Set Sorter_STDadmissionNum = CCCreateSorter(                      ↙
    "Sorter_STDadmissionNum", Me, FileName)
        Set Sorter_STDid = CCCreateSorter("Sorter_STDid", Me, FileName)
        Set Sorter_STDname = CCCreateSorter("Sorter_STDname", Me,         ↙
    FileName)
        Set Sorter_STDsex = CCCreateSorter("Sorter_STDsex", Me, FileName)
        Set Navigator = CCCreateNavigator(ComponentName, "Navigator",     ↙
    FileName, 10, tpSimple)
    End Sub
'End Student Class_Initialize Event

'Student Initialize Method 62-2AEA3975
    Sub Initialize(objConnection)
        If NOT Visible Then Exit Sub

        Set DataSource.Connection = objConnection
        DataSource.PageSize = PageSize
        DataSource.SetOrder ActiveSorter, SortingDirection
        DataSource.AbsolutePage = PageNumber
    End Sub
```

```
'End Student Initialize Method

'Student Class_Terminate Event @2-E1568F98
    Private Sub Class_Terminate()
        Set DataSource = Nothing
        Set Command = Nothing
        Set Errors = Nothing
    End Sub
'End Student Class_Terminate Event

'Student Show Method @2-2B4CC515
    Sub Show(Tpl)
        If NOT Visible Then Exit Sub

        Dim RecordCounter, ShownRecords
        Dim RowBlock

        With DataSource
            .Parameters("urls_STDid") = CCGetRequestParam("s_STDid",        ↙
    ccsGET)
            .Parameters("urls_STDname") = CCGetRequestParam("s_STDname",    ↙
    ccsGET)
            .Parameters("urls_STDadmissionNum") = CCGetRequestParam(        ↙
    "s_STDadmissionNum", ccsGET)
            .Parameters("urls_AdmYear") = CCGetRequestParam("s_AdmYear",    ↙
    ccsGET)
        End With

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)

        Set TemplateBlock = Tpl.Block("Grid " & ComponentName)
        Set RowBlock = TemplateBlock.Block("Row")
        Set StaticControls = CCCreateCollection(TemplateBlock, Null,        ↙
    ccsParseOverwrite, _
            Array(Sorter_STDadmissionNum, Sorter_STDid, Sorter_STDname,     ↙
    Sorter_STDsex, Navigator))
        Set RowControls = CCCreateCollection(RowBlock, Null,                ↙
    ccsParseAccumulate, _
            Array(STDadmissionNum, Label1, STDid, STDname, STDsex))

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If NOT Visible Then Exit Sub

        Errors.AddErrors DataSource.Errors
        If Errors.Count > 0 Then
            TemplateBlock.HTML = CCFormatError("Grid Student", Errors)
        Else

            ' Show NoRecords block if no records are found
            If Recordset.EOF Then
                TemplateBlock.Block("NoRecords").Parse ccsParseOverwrite
            End If
            While NOT Recordset.EOF AND ShownRecords < PageSize
                STDadmissionNum.Value = Recordset.Fields("STDadmissionNum"↙
    )
                Label1.Value = Recordset.Fields("Label1")
                STDid.Value = Recordset.Fields("STDid")
                STDid.Parameters = CCGetQueryString("QueryString", Array( ↙
    "ccsForm"))
                STDid.Parameters = CCAddParam(STDid.Parameters, "STDid",  ↙
    Recordset.Fields("STDid_param1"))
                STDid.Page = "STAFF_StdRecords.asp"
                STDname.Value = Recordset.Fields("STDname")
                STDsex.Value = Recordset.Fields("STDsex")
                CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShowRow", ↙
    Me)
                RowControls.Show
```

```
                Recordset.MoveNext
                ShownRecords = ShownRecords + 1
            Wend
            Navigator.SetDataSource Recordset
            StaticControls.Show
        End If

    End Sub
'End Student Show Method

End Class 'End Student Class 02-A61BA092

Class clsStudentDataSource 'StudentDataSource Class 02-2F9B50BE

'DataSource Variables 02-A4497640
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource Fields
    Public STDadmissionNum
    Public Label1
    Public STDid
    Public STDid_param1
    Public STDname
    Public STDsex
'End DataSource Variables

'DataSource Class_Initialize Event 02-6F85AC6D
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDadmissionNum = CCCreateField("STDadmissionNum",                ↙
    "STDadmissionNum", ccsFloat, Empty, Recordset)
        Set Label1 = CCCreateField("Label1", "STDstampdate", ccsDate,         ↙
    Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"), ↙
    Recordset)
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,           ↙
    Recordset)
        Set STDid_param1 = CCCreateField("STDid_param1", "STDid", ccsText,↙
     Empty, Recordset)
        Set STDname = CCCreateField("STDname", "STDname", ccsText, Empty, ↙
    Recordset)
        Set STDsex = CCCreateField("STDsex", "Type", ccsText, Empty,          ↙
    Recordset)
        Fields.AddFields Array(STDadmissionNum, Label1, STDid,                ↙
    STDid_param1, STDname, STDsex)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing
        Orders = Array( _
            Array("Sorter_STDadmissionNum", "STDadmissionNum", ""), _
            Array("Sorter_STDid", "STDid", ""), _
            Array("Sorter_STDname", "STDname", ""), _
```

```
            Array("Sorter_STDsex", "STDsex", ""))

        SQL = "SELECT *  " & _
        "FROM lstRegisteredStd"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM lstRegisteredStd"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'SetOrder Method 62-6BFC9576
    Sub SetOrder(Column, Direction)
        Order = Recordset.GetOrder(Order, Column, Direction, Orders)
    End Sub
'End SetOrder Method

'BuildTableWhere Method 62-7E3ED753
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urls_STDid", ccsText, Empty, Empty, Empty,  ↙
    False
            .AddParameter 2, "urls_STDname", ccsText, Empty, Empty, Empty,↙
     False
            .AddParameter 3, "urls_STDadmissionNum", ccsFloat, Empty,     ↙
    Empty, Empty, False
            .AddParameter 4, "urls_AdmYear", ccsInteger, Empty, Empty,    ↙
    Empty, False
            .Criterion(1) = .Operation(opContains, False, "STDid", .      ↙
    getParamByID(1))
            .Criterion(2) = .Operation(opContains, False, "STDname", .    ↙
    getParamByID(2))
            .Criterion(3) = .Operation(opEqual, False, "STDadmissionNum", ↙
    .getParamByID(3))
            .Criterion(4) = .Operation(opEqual, False, "AdmYear", .       ↙
    getParamByID(4))
            .AssembledWhere = .opAND(False, .opAND(False, .opAND(False, . ↙
    Criterion(1), .Criterion(2)), .Criterion(3)), .Criterion(4))
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 62-F13016E1
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
```

```
        Cmd.SQL = SQL
        Cmd.CountSQL = CountSQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {2-4184B08D
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End StudentDataSource Class {2-A61BA692


%>
```

```
<%
'Include Common Files @1-3BED0132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-CU&B7291
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim CLASSSTD
Dim Student

Redirect = ""
TemplateFileName = "STAFF_StdRecords.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STAFF_StdRecords.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-D41460BA
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set CLASSSTD = new clsRecordCLASSSTD
Set Student = new clsRecordStudent
CLASSSTD.Initialize DBConnection1
Student.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-A6CE9410
CLASSSTD.Operation
Student.Operation
'End Execute Components

'Go to destination page @1-6D35F4F2
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0159C18E
```

```
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-8CE13022
CLASSSTD.Show Tpl
Student.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210352
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-5114E337
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set CLASSSTD = Nothing
    Set Student = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordCLASSSTD 'CLASSSTD Class @9-1955B9A6

'CLASSSTD Variables @9-4F9D94A0

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public InsertAllowed
    Public UpdateAllowed
    Public DeleteAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button1
    Dim Button2
    Dim Button3
'End CLASSSTD Variables

'CLASSSTD Class_Initialize Event @9-E1349E6F
    Private Sub Class_Initialize()
```

```
        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsCLASSSTDDataSource
        Set Command = New clsCommand
        InsertAllowed = True
        UpdateAllowed = True
        DeleteAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "CLASSSTD")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "CLASSSTD"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button1 = CCCreateButton("Button1")
            Set Button2 = CCCreateButton("Button2")
            Set Button3 = CCCreateButton("Button3")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End CLASSSTD Class_Initialize Event

'CLASSSTD Initialize Method @9-476B1952
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlCLASSid") = CCGetRequestParam("CLASSid",
    ccsGET)
        End With
    End Sub
'End CLASSSTD Initialize Method

'CLASSSTD Class_Terminate Event @9-32B64709
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End CLASSSTD Class_Terminate Event

'CLASSSTD Validate Method @9-7AE52642
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End CLASSSTD Validate Method

'CLASSSTD Operation Method @9-27111023
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button1", "Button1")
            If Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            ElseIf Not IsEmpty(CCGetParam("Button2", Empty)) Then
                PressedButton = "Button2"
```

```
            ElseIf Not IsEmpty(CCGetParam("Button3", Empty)) Then
                PressedButton = "Button3"
            End If
        End If
        Redirect = "STAFF_StdRecords.asp?" & CCGetQueryString("QueryString↙
    ", Array("ccsForm", "Button1", "Button2", "Button3"))
        If Validate() Then
            If PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_List.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm", "STDid"))

                End If
            ElseIf PressedButton = "Button2" Then
                If NOT Button2.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_StdList.asp?" & CCGetQueryString(   ↙
    "QueryString", Array("ccsForm", "STDid"))

                End If
            ElseIf PressedButton = "Button3" Then
                If NOT Button3.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_options.asp?" & CCGetQueryString(   ↙
    "QueryString", Array("ccsForm", "STDid"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End CLASSSTL Operation Method

'CLASSSTL InsertRow Method 69-C93E1C3D
    Function InsertRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeInsert", Me)
        If NOT InsertAllowed Then InsertRow = False : Exit Function
        DataSource.Insert(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInsert", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Insert Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        InsertRow = (Errors.Count = 0)
    End Function
'End CLASSSTL InsertRow Method

'CLASSSTL UpdateRow Method 69-A1504176
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
```

```
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End CLASSSTD UpdateRow Method

'CLASSSTD DeleteRow Method 69-1BC26772
    Function DeleteRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeDelete", Me)
        If NOT DeleteAllowed Then DeleteRow = False : Exit Function
        DataSource.Delete(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterDelete", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Delete Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        DeleteRow = (Errors.Count = 0)
    End Function
'End CLASSSTD DeleteRow Method

'CLASSSTD Show Method 69-675350BB
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "CLASSSTD" & IIf(EditMode,↙
     ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(Button1, Button2, Button3))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record CLASSSTD", DataSource.Connection.↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                    End If
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
```

```
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End CLASSSTD Show Method

End Class 'End CLASSSTD Class 69-A61BA892

Class clsCLASSSTDDataSource 'CLASSSTDDataSource Class 69-80976397

'DataSource Variables 69-184E7B43
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

'End DataSource Variables

'DataSource Class_Initialize Event 69-20C21800
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM CLASSSTD"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM CLASSSTD"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method 69-80CB531B
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlCLASSid", ccsFloat, Empty, Empty, Empty, ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "CLASSid", .        ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
```

```
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 69-DF951BED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event 69-41D4B0ED
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Delete Method 69-7FB11504
    Sub Delete(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildDelete", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Delete operation. There
     are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "DELETE FROM CLASSSTD" & IIf(Len(Where) > 0, " WHERE " &
     Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteDelete",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteDelete", Me)
```

```
    End Sub
'End Delete Method

'Update Method (9-FB6291F0
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE CLASSSTD SET " & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

'Insert Method (9-E90B0AEE
    Sub Insert(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildInsert", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        Cmd.SQL = "INSERT INTO CLASSSTD(" & _
        ") VALUES (" & _
        ")"
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteInsert",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteInsert", Me)
    End Sub
'End Insert Method

End Class 'End CLASSSTDDataSource Class (9-4518AB92

Class clsRecordStudent 'Student Class (18-44207813

'Student Variables (18-30005E37

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public UpdateAllowed
    Public DeleteAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls
```

```
    ' Class variables
    Dim STDid
    Dim STDname
    Dim STDadmissionNum
    Dim Label1
    Dim STDsex
    Dim STDsexDataSource
    Dim STDreligion
    Dim STDrace
    Dim STDbirthcert
    Dim STDdob
    Dim DatePicker_STDdob
    Dim STDpob
    Dim STDiniclass
    Dim STDfathername
    Dim STDfatheradd
    Dim STDguardianname
    Dim STDguardianadd
    Dim STDguardiantype
    Dim STDstampdate
    Dim Button_Update
    Dim Button_Delete
'End Student Variables

'Student Class_Initialize Event @18-022FECA3
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsStudentDataSource
        Set Command = New clsCommand
        UpdateAllowed = True
        DeleteAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Student")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Student"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set STDid = CCCreateControl(ccsLabel, "STDid", "STDid",
    ccsText, Empty, CCGetRequestParam("STDid", Method))
            Set STDname = CCCreateControl(ccsTextBox, "STDname", "STDname"
    , ccsText, Empty, CCGetRequestParam("STDname", Method))
            Set STDadmissionNum = CCCreateControl(ccsLabel,
    "STDadmissionNum", "STDadmission Num", ccsFloat, Array(True, False, ""
    , False, "", Array("0", "0", "0", "0", "0", "0"), Empty, 1, True, "",
    False), CCGetRequestParam("STDadmissionNum", Method))
            Set Label1 = CCCreateControl(ccsLabel, "Label1", "Label1",
    ccsDate, Array("yy"), CCGetRequestParam("Label1", Method))
            Set STDsexDataSource = CCCreateDataSource(dsTable,
    DBConnection1, Array("SELECT *  " & _
"FROM Type", "", ""))
            Set STDsex = CCCreateList(ccsListBox, "STDsex", "STDsex",
    ccsText, CCGetRequestParam("STDsex", Method), STDsexDataSource)
            Set STDreligion = CCCreateControl(ccsTextBox, "STDreligion",
    "STDreligion", ccsText, Empty, CCGetRequestParam("STDreligion",
    Method))
            Set STDrace = CCCreateControl(ccsTextBox, "STDrace", "STDrace"
    , ccsText, Empty, CCGetRequestParam("STDrace", Method))
```

```
            Set STDbirthcert = CCCreateControl(ccsTextBox, "STDbirthcert",↙
     "STDbirthcert", ccsText, Empty, CCGetRequestParam("STDbirthcert",    ↙
    Method))
            Set STDdob = CCCreateControl(ccsTextBox, "STDdob", "STDdob",  ↙
    ccsDate, Array("dd", "/", "mm", "/", "yyyy"), CCGetRequestParam(      ↙
    "STDdob", Method))
            Set DatePicker_STDdob = CCCreateDatePicker("DatePicker_STDdob"↙
    , "Student", "STDdob")
            Set STDpob = CCCreateControl(ccsTextBox, "STDpob", "STDpob",  ↙
    ccsText, Empty, CCGetRequestParam("STDpob", Method))
            Set STDiniclass = CCCreateControl(ccsTextBox, "STDiniclass",  ↙
    "STDiniclass", ccsText, Empty, CCGetRequestParam("STDiniclass",       ↙
    Method))
            Set STDfathername = CCCreateControl(ccsTextBox, "STDfathername↙
    ", "STDfathername", ccsText, Empty, CCGetRequestParam("STDfathername",↙
     Method))
            Set STDfatheradd = CCCreateControl(ccsTextArea, "STDfatheradd"↙
    , "STDfatheradd", ccsText, Empty, CCGetRequestParam("STDfatheradd",   ↙
    Method))
            Set STDguardianname = CCCreateControl(ccsTextBox,             ↙
    "STDguardianname", "STDguardianname", ccsText, Empty,                 ↙
    CCGetRequestParam("STDguardianname", Method))
            Set STDguardianadd = CCCreateControl(ccsTextArea,             ↙
    "STDguardianadd", "STDguardianadd", ccsText, Empty, CCGetRequestParam ↙
    ("STDguardianadd", Method))
            Set STDguardiantype = CCCreateControl(ccsTextBox,             ↙
    "STDguardiantype", "STDguardiantype", ccsText, Empty,                 ↙
    CCGetRequestParam("STDguardiantype", Method))
            Set STDstampdate = CCCreateControl(ccsLabel, "STDstampdate",  ↙
    "STDstampdate", ccsDate, Array("dd", "/", "mm", "/", "yyyy"),         ↙
    CCGetRequestParam("STDstampdate", Method))
            Set Button_Update = CCCreateButton("Button_Update")
            Set Button_Delete = CCCreateButton("Button_Delete")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STDname, STDsex,         ↙
    STDreligion, STDrace, STDbirthcert, STDdob, STDpob, STDiniclass,      ↙
    STDfathername, STDfatheradd, STDguardianname, STDguardianadd,         ↙
    STDguardiantype)
        End If
    End Sub
'End Student_Class_Initialize Event

'Student Initialize Method {1E-2179AA2}
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlSTDid") = CCGetRequestParam("STDid", ccsGET)
        End With
    End Sub
'End Student Initialize Method

'Student Class_Terminate Event {1E-32B647C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Student Class_Terminate Event

'Student Validate Method {1E-7AE52864}
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
```

```
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterDelete", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Delete Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        DeleteRow = (Errors.Count = 0)
    End Function
'End Student DeleteRow Method

'Student Show Method {16-DFB5AA5
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Student" & IIf(EditMode, ↙
    ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(STDid, STDname, STDadmissionNum, Label1, STDsex,           ↙
    STDreligion, STDrace, STDbirthcert, STDdob, DatePicker_STDdob, STDpob,↙
     STDiniclass, STDfathername, STDfatheradd, STDguardianname,              ↙
    STDguardianadd, STDguardiantype, STDstampdate, Button_Update,            ↙
    Button_Delete))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record Student", DataSource.Connection. ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        STDname.Value = Recordset.Fields("STDname")
                        STDsex.Value = Recordset.Fields("STDsex")
                        STDreligion.Value = Recordset.Fields("STDreligion"↙
    )
                        STDrace.Value = Recordset.Fields("STDrace")
                        STDbirthcert.Value = Recordset.Fields(               ↙
    "STDbirthcert")
                        STDdob.Value = Recordset.Fields("STDdob")
                        STDpob.Value = Recordset.Fields("STDpob")
                        STDiniclass.Value = Recordset.Fields("STDiniclass"↙
    )
                        STDfathername.Value = Recordset.Fields(              ↙
    "STDfathername")
                        STDfatheradd.Value = Recordset.Fields(               ↙
    "STDfatheradd")
                        STDguardianname.Value = Recordset.Fields(            ↙
    "STDguardianname")
                        STDguardianadd.Value = Recordset.Fields(             ↙
    "STDguardianadd")
                        STDguardiantype.Value = Recordset.Fields(            ↙
    "STDguardiantype")
                    End If
```

```
                    STDid.Value = Recordset.Fields("STDid")
                    STDadmissionNum.Value = Recordset.Fields(
    "STDadmissionNum")
                    Label1.Value = Recordset.Fields("Label1")
                    STDstampdate.Value = Recordset.Fields("STDstampdate")
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Update.Visible = EditMode AND UpdateAllowed
        Button_Delete.Visible = EditMode AND DeleteAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Student Show Method

End Class 'End Student Class 618-A61BAE92

Class clsStudentDataSource 'StudentDataSource Class 618-1F0C0BE

'DataSource Variables 618-97917451
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STDid
    Public STDname
    Public STDadmissionNum
    Public Label1
    Public STDsex
    Public STDreligion
    Public STDrace
    Public STDbirthcert
    Public STDdob
    Public STDpob
    Public STDiniclass
    Public STDfathername
    Public STDfatheradd
    Public STDguardianname
    Public STDguardianadd
    Public STDguardiantype
    Public STDstampdate
'End DataSource Variables
```

```
'DataSource Class_Initialize Event 018-0B6D2FBF6
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,        ↙
    Recordset)
        Set STDname = CCCreateField("STDname", "STDname", ccsText, Empty, ↙
    Recordset)
        Set STDadmissionNum = CCCreateField("STDadmissionNum",             ↙
    "STDadmissionNum", ccsFloat, Empty, Recordset)
        Set Label1 = CCCreateField("Label1", "STDstampdate", ccsDate,      ↙
    Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"), ↙
    Recordset)
        Set STDsex = CCCreateField("STDsex", "STDsex", ccsText, Empty,     ↙
    Recordset)
        Set STDreligion = CCCreateField("STDreligion", "STDreligion",      ↙
    ccsText, Empty, Recordset)
        Set STDrace = CCCreateField("STDrace", "STDrace", ccsText, Empty, ↙
    Recordset)
        Set STDbirthcert = CCCreateField("STDbirthcert", "STDbirthcert",   ↙
    ccsText, Empty, Recordset)
        Set STDdob = CCCreateField("STDdob", "STDdob", ccsDate, Array(     ↙
    "yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"),        ↙
    Recordset)
        Set STDpob = CCCreateField("STDpob", "STDpob", ccsText, Empty,     ↙
    Recordset)
        Set STDiniclass = CCCreateField("STDiniclass", "STDiniclass",      ↙
    ccsText, Empty, Recordset)
        Set STDfathername = CCCreateField("STDfathername", "STDfathername"↙
    , ccsText, Empty, Recordset)
        Set STDfatheradd = CCCreateField("STDfatheradd", "STDfatheradd",   ↙
    ccsText, Empty, Recordset)
        Set STDguardianname = CCCreateField("STDguardianname",             ↙
    "STDguardianname", ccsText, Empty, Recordset)
        Set STDguardianadd = CCCreateField("STDguardianadd",               ↙
    "STDguardianadd", ccsText, Empty, Recordset)
        Set STDguardiantype = CCCreateField("STDguardiantype",             ↙
    "STDguardiantype", ccsText, Empty, Recordset)
        Set STDstampdate = CCCreateField("STDstampdate", "STDstampdate",   ↙
    ccsDate, Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":"↙
    , "ss"), Recordset)
        Fields.AddFields Array(STDid, STDname, STDadmissionNum, Label1,    ↙
    STDsex, STDreligion, STDrace, STDbirthcert, STDdob, STDpob,            ↙
    STDiniclass, STDfathername, STDfatheradd, STDguardianname,             ↙
    STDguardianadd, STDguardiantype, STDstampdate)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method 018-7E709E1C
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
```

```
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlSTDid", ccsText, Empty, Empty, Empty,     ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STDid", .          ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method {18-DF9518EE
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {18-41B4B06E
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Delete Method {18-41150483
    Sub Delete(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildDelete", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
```

```
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Delete operation. There↙
     are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "DELETE FROM Student" & IIf(Len(Where) > 0, " WHERE " & ↙
    Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteDelete",    ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteDelete", Me)
    End Sub
'End Delete Method

'Update Method 618-C7A7C132
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.       ↙
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE Student SET " & _
            "STDname=" & Connection.ToSQL(STDname.SQLText, STDname.        ↙
    DataType) & ", " & _
            "STDsex=" & Connection.ToSQL(STDsex.SQLText, STDsex.DataType) ↙
    & ", " & _
            "STDreligion=" & Connection.ToSQL(STDreligion.SQLText,         ↙
    STDreligion.DataType) & ", " & _
            "STDrace=" & Connection.ToSQL(STDrace.SQLText, STDrace.        ↙
    DataType) & ", " & _
            "STDbirthcert=" & Connection.ToSQL(STDbirthcert.SQLText,       ↙
    STDbirthcert.DataType) & ", " & _
            "STDdob=" & Connection.ToSQL(STDdob.SQLText, STDdob.DataType) ↙
    & ", " & _
            "STDpob=" & Connection.ToSQL(STDpob.SQLText, STDpob.DataType) ↙
    & ", " & _
            "STDiniclass=" & Connection.ToSQL(STDiniclass.SQLText,         ↙
    STDiniclass.DataType) & ", " & _
            "STDfathername=" & Connection.ToSQL(STDfathername.SQLText,     ↙
    STDfathername.DataType) & ", " & _
            "STDfatheradd=" & Connection.ToSQL(STDfatheradd.SQLText,       ↙
    STDfatheradd.DataType) & ", " & _
            "STDguardianname=" & Connection.ToSQL(STDguardianname.SQLText,↙
     STDguardianname.DataType) & ", " & _
            "STDguardianadd=" & Connection.ToSQL(STDguardianadd.SQLText,   ↙
    STDguardianadd.DataType) & ", " & _
            "STDguardiantype=" & Connection.ToSQL(STDguardiantype.SQLText,↙
     STDguardiantype.DataType) & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",    ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

End Class 'End StudentDataSource Class 618-A61BA6B2


%>
```

```
<%
'Include Common Files @1-2B0E0118
%>
<!-- #INCLUDE FILE="./Common.asp"-->
<!-- #INCLUDE FILE="./Cache.asp" -->
<!-- #INCLUDE FILE="./Template.asp" -->
<!-- #INCLUDE FILE="./Sorter.asp" -->
<!-- #INCLUDE FILE="./Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-29CCF3C0
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Class1

Redirect = ""
TemplateFileName = "CLASS_StdEnroll.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "CLASS_StdEnroll.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-2256752E
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Class1 = new clsRecordClass1
Class1.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-6C275794
Class1.Operation
'End Execute Components

'Go to destination page @1-6C953457
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-3F479BE7
Class1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-4143E5C3
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Class1 = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordClass1 'Class1 Class @2-A9174E40

'Class1 Variables @2-2C5E1CA8

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim CLASSid
    Dim CLASSname
    Dim CLASSteacher
    Dim CLASSyear
'End Class1 Variables

'Class1 Class_Initialize Event @2-73577462
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsClass1DataSource
        Set Command = New clsCommand
        Dim Method
        Dim OperationMode
```

```
        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Class1")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Class1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set CLASSid = CCCreateControl(ccsLabel, "CLASSid", "Id",
    ccsFloat, Empty, CCGetRequestParam("CLASSid", Method))
            Set CLASSname = CCCreateControl(ccsLabel, "CLASSname", "Name",
     ccsText, Empty, CCGetRequestParam("CLASSname", Method))
            Set CLASSteacher = CCCreateControl(ccsLabel, "CLASSteacher",
    "Teacher", ccsText, Empty, CCGetRequestParam("CLASSteacher", Method))
            Set CLASSyear = CCCreateControl(ccsLabel, "CLASSyear", "Year",
     ccsDate, Array("dd", "/", "mm", "/", "yyyy"), CCGetRequestParam(
    "CLASSyear", Method))
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End Class1 Class_Initialize Event

'Class1 Initialize Method {2-476B2952
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlCLASSid") = CCGetRequestParam("CLASSid",
    ccsGET)
        End With
    End Sub
'End Class1 Initialize Method

'Class1 Class_Terminate Event {2-9BB4739
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Class1 Class_Terminate Event

'Class1 Validate Method {2-7AE52642
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Class1 Validate Method

'Class1 Operation Method {2-114793B4
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        Redirect = "CLASS_StdEnroll.asp?" & CCGetQueryString("QueryString"
    , Array("ccsForm"))
    End Sub
'End Class1 Operation Method

'Class1 Show Method {2-15B876B5
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
```

```
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Class1" & IIf(EditMode, "↙
    :Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(CLASSid, CLASSname, CLASSteacher, CLASSyear))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record Class1", DataSource.Connection.  ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                    End If
                    CLASSid.Value = Recordset.Fields("CLASSid")
                    CLASSname.Value = Recordset.Fields("CLASSname")
                    CLASSteacher.Value = Recordset.Fields("CLASSteacher")
                    CLASSyear.Value = Recordset.Fields("CLASSyear")
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Class1 Show Method

End Class 'End Class1 Class CF-A61BA8DC

Class clsClass1DataSource ''Class1DataSource Class CF-E4994DAA

'DataSource Variables CF-6511CD0
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
```

```
    Private CCSEventResult

    ' Datasource fields
    Public CLASSid
    Public CLASSname
    Public CLASSteacher
    Public CLASSyear
'End DataSource Variables

'DataSource Class_Initialize Event 62-AC7BE4D0
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set CLASSid = CCCreateField("CLASSid", "CLASSid", ccsFloat, Empty,↙
     Recordset)
        Set CLASSname = CCCreateField("CLASSname", "CLASSname", ccsText,  ↙
    Empty, Recordset)
        Set CLASSteacher = CCCreateField("CLASSteacher", "CLASSteacher",  ↙
    ccsText, Empty, Recordset)
        Set CLASSyear = CCCreateField("CLASSyear", "CLASSyear", ccsDate,  ↙
    Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"), ↙
    Recordset)
        Fields.AddFields Array(CLASSid, CLASSname, CLASSteacher,          ↙
    CLASSyear)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM Class"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Class"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method 62-E0CB531B
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlCLASSid", ccsFloat, Empty, Empty, Empty, ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "CLASSid", .       ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method
```

```
'Class Method {C-39951BE2}
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {E1-4164B0E2}
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End ClassDataSource Class {E1-A61BAA9E}


%>
```

```
<%
'BindEvents Method @1-6700572B
Sub BindEvents()
    Set Student1.Student1_TotalRecords.CCSEvents("BeforeShow") = GetRef(  ↙
    "Student1_Student1_TotalRecords_BeforeShow")
End Sub
'End BindEvents Method

Function Student1_Student1_TotalRecords_BeforeShow()                      ↙
    'Student1_Student1_TotalRecords_BeforeShow @30-D0647050

'Retrieve number of records @31-24EF1A3A
    Student1.Student1_TotalRecords.Value = Student1.DataSource.Recordset. ↙
    RecordCount
'End Retrieve number of records

End Function 'Close Student1_Student1_TotalRecords_BeforeShow @30-54C34B2E


%>
```

```
<%
'Include Common Files @1-D6E3E13C
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-EB6ACDCF
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Class1
Dim CLASSSTD
Dim Student1

Redirect = ""
TemplateFileName = "CLASS_DataEntry.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "CLASS_DataEntry.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-3257173F
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Class1 = new clsRecordClass1
Set CLASSSTD = new clsRecordCLASSSTD
Set Student1 = New clsGridStudent1
Class1.Initialize DBConnection1
CLASSSTD.Initialize DBConnection1
Student1.Initialize DBConnection1

' Events
%>
<!-- #INCLUDE FILE="CLASS_DataEntry_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-04CBE372
Class1.Operation
CLASSSTD.Operation
'End Execute Components
```

```
'Go to destination page 01-5B4B74E5
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template 01-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page 01-9810C267
Class1.Show Tpl
CLASSSTD.Show Tpl
Student1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page 01-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub 01-6D407E0E
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Class1 = Nothing
    Set CLASSSTD = Nothing
    Set Student1 = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordClass1 'Class1 Class 01-A9174E69

'Class1 Variables 01-6A962C05

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public InsertAllowed
    Public UpdateAllowed
    Public DeleteAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
```

```
    Public Controls

    ' Class variables
    Dim CLASSname
    Dim CLASSteacher
    Dim CLASSyear
    Dim DatePicker_CLASSyear
    Dim CLASSid
    Dim Button_Insert
    Dim Button_Update
    Dim Button_Delete
    Dim Button1
    Dim Button2
'End Class1 Variables

'Class1 Class_Initialize Event 02-E15E6B43
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsClass1DataSource
        Set Command = New clsCommand
        InsertAllowed = True
        UpdateAllowed = True
        DeleteAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Class1")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Class1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set CLASSname = CCCreateControl(ccsTextBox, "CLASSname", "Name↙
    ", ccsText, Empty, CCGetRequestParam("CLASSname", Method))
            Set CLASSteacher = CCCreateControl(ccsTextBox, "CLASSteacher",↙
     "Teacher", ccsText, Empty, CCGetRequestParam("CLASSteacher", Method))
            Set CLASSyear = CCCreateControl(ccsTextBox, "CLASSyear", "Year↙
    ", ccsDate, Array("mm", "/", "yyyy"), CCGetRequestParam("CLASSyear",  ↙
    Method))
            Set DatePicker_CLASSyear = CCCreateDatePicker(                ↙
    "DatePicker_CLASSyear", "Class1", "CLASSyear")
            Set CLASSid = CCCreateControl(ccsLabel, "CLASSid", "CLASSid", ↙
    ccsText, Empty, CCGetRequestParam("CLASSid", Method))
            Set Button_Insert = CCCreateButton("Button_Insert")
            Set Button_Update = CCCreateButton("Button_Update")
            Set Button_Delete = CCCreateButton("Button_Delete")
            Set Button1 = CCCreateButton("Button1")
            Set Button2 = CCCreateButton("Button2")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(CLASSname, CLASSteacher, ↙
    CLASSyear)
        End If
    End Sub
'End Class1 Class_Initialize Event

'Class1 Initialize Method 02-4706E9C2
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub
```

```
        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlCLASSid") = CCGetRequestParam("CLASSid",        ↙
    ccsGET)
        End With
    End Sub
'End Class1 Initialize Method

'Class1 Class_Terminate Event 02-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Class1 Class_Terminate Event

'Class1 Validate Method 02-7AE52540
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Class1 Validate Method

'Class1 Operation Method 02-62594B0F
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "Button_Insert"↙
    )
            If Not IsEmpty(CCGetParam("Button_Insert", Empty)) Then
                PressedButton = "Button_Insert"
            ElseIf Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            ElseIf Not IsEmpty(CCGetParam("Button_Delete", Empty)) Then
                PressedButton = "Button_Delete"
            ElseIf Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            ElseIf Not IsEmpty(CCGetParam("Button2", Empty)) Then
                PressedButton = "Button2"
            End If
        End If
        Redirect = "CLASS_DataEntry.asp?" & CCGetQueryString("QueryString"↙
    , Array("ccsForm", "Button_Insert", "Button_Update", "Button_Delete", ↙
    "Button1", "Button2"))
        If PressedButton = "Button_Delete" Then
            If NOT Button_Delete.OnClick OR NOT DeleteRow() Then
                Redirect = ""
            Else
                Redirect = "CLASS_List.asp?" & CCGetQueryString(            ↙
    "QueryString", Array("ccsForm"))
            End If
        ElseIf Validate() Then
            If PressedButton = "Button_Insert" Then
                If NOT Button_Insert.OnClick() OR NOT InsertRow() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_List.asp?" & CCGetQueryString(        ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_List.asp?" & CCGetQueryString(        ↙
    "QueryString", Array("ccsForm"))
```

```
                End If
            ElseIf PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_List.asp?" & CCGetQueryString(        ↙
    "QueryString", Array("ccsForm", "STDid", "CLASSid"))

                End If
            ElseIf PressedButton = "Button2" Then
                If NOT Button2.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_options.asp?" & CCGetQueryString(     ↙
    "QueryString", Array("ccsForm", "CLASSid", "STDid"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Class1 Operation Method

'Class1 InsertRow Method {1-B4D3644E
    Function InsertRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeInsert", Me)
        If NOT InsertAllowed Then InsertRow = False : Exit Function
        DataSource.CLASSname.Value = CLASSname.Value
        DataSource.CLASSteacher.Value = CLASSteacher.Value
        DataSource.CLASSyear.Value = CLASSyear.Value
        DataSource.Insert(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInsert", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Insert Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        InsertRow = (Errors.Count = 0)
    End Function
'End Class1 InsertRow Method

'Class1 UpdateRow Method {1-074[illegible]
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.CLASSname.Value = CLASSname.Value
        DataSource.CLASSteacher.Value = CLASSteacher.Value
        DataSource.CLASSyear.Value = CLASSyear.Value
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End Class1 UpdateRow Method

'Class1 DeleteRow Method {1-16C[illegible]
    Function DeleteRow()
```

```
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeDelete", Me)
        If NOT DeleteAllowed Then DeleteRow = False : Exit Function
        DataSource.Delete(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterDelete", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Delete Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        DeleteRow = (Errors.Count = 0)
    End Function
'End Class1 DeleteRow Method

'Class1 Show Method 02-8A78E1EE
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Class1" & IIf(EditMode, "↙
    :Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(CLASSname, CLASSteacher, CLASSyear, DatePicker_CLASSyear↙
    , CLASSid, Button_Insert, Button_Update, Button_Delete, Button1,         ↙
    Button2))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record Class1", DataSource.Connection.     ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        CLASSname.Value = Recordset.Fields("CLASSname")
                        CLASSteacher.Value = Recordset.Fields(               ↙
    "CLASSteacher")
                        CLASSyear.Value = Recordset.Fields("CLASSyear")
                    End If

                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
```

```
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Insert.Visible = NOT EditMode AND InsertAllowed
        Button_Update.Visible = EditMode AND UpdateAllowed
        Button_Delete.Visible = EditMode AND DeleteAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Class1 Show Method

End Class 'End Class1 Class 61-A61BA692

Class clsClass1DataSource 'Class1DataSource Class 62-24994BAA

'DataSource Variables 61-58851937
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource Fields
    Public CLASSname
    Public CLASSteacher
    Public CLASSyear
'End DataSource Variables

'DataSource Class_Initialize Event 61-97E32152
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set CLASSname = CCCreateField("CLASSname", "CLASSname", ccsText,  ↙
    Empty, Recordset)
        Set CLASSteacher = CCCreateField("CLASSteacher", "CLASSteacher",  ↙
    ccsText, Empty, Recordset)
        Set CLASSyear = CCCreateField("CLASSyear", "CLASSyear", ccsDate,  ↙
    Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"), ↙
    Recordset)
        Fields.AddFields Array(CLASSname, CLASSteacher, CLASSyear)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM Class"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Class"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method 62-60785216
    Public Sub BuildTableWhere()
        Dim WhereParams
```

```
        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlCLASSid", ccsFloat, Empty, Empty, Empty, ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "CLASSid", .        ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 62-DF951BED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event 62-41B4B08E
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Delete Method 62-45370177
    Sub Delete(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildDelete", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
```

```
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Delete operation. There↙
 are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "DELETE FROM Class" & IIf(Len(Where) > 0, " WHERE " &    ↙
Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteDelete",    ↙
Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteDelete", Me)
    End Sub
'End Delete Method

'Update Method 62-95A94090
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.       ↙
There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE Class SET " & _
            "CLASSname=" & Connection.ToSQL(CLASSname.SQLText, CLASSname. ↙
DataType) & ", " & _
            "CLASSteacher=" & Connection.ToSQL(CLASSteacher.SQLText,       ↙
CLASSteacher.DataType) & ", " & _
            "CLASSyear=" & Connection.ToSQL(CLASSyear.SQLText, CLASSyear. ↙
DataType) & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",    ↙
Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

'Insert Method 62-9B613CB4
    Sub Insert(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildInsert", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        Cmd.SQL = "INSERT INTO Class(" & _
            "CLASSname, " & _
            "CLASSteacher, " & _
            "CLASSyear" & _
        ") VALUES (" & _
            Connection.ToSQL(CLASSname.SQLText, CLASSname.DataType) & ", "↙
 & _
            Connection.ToSQL(CLASSteacher.SQLText, CLASSteacher.DataType) ↙
& ", " & _
            Connection.ToSQL(CLASSyear.SQLText, CLASSyear.DataType) & _
        ")"
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteInsert",    ↙
Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteInsert", Me)
    End Sub
```

```
'End Insert Method

End Class 'End ClassDataSource Class 41-A61BA893

Class clsRecordCLASSSTD 'CLASSSTD Class 041-C955B9A6

'CLASSSTD Variables 041-4F24F114

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public InsertAllowed
    Public UpdateAllowed
    Public DeleteAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim STDid
    Dim CStimestamp
    Dim Button_Insert
    Dim Button_Update
    Dim Button_Delete
'End CLASSSTD Variables

'CLASSSTD Class Initialize Event 041-7941E7E2
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsCLASSSTDDataSource
        Set Command = New clsCommand
        InsertAllowed = True
        UpdateAllowed = True
        DeleteAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "CLASSSTD")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "CLASSSTD"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set STDid = CCCreateControl(ccsTextBox, "STDid", "STDid",
    ccsText, Empty, CCGetRequestParam("STDid", Method))
            STDid.Required = True
            Set CStimestamp = CCCreateControl(ccsLabel, "CStimestamp",
    "CStimestamp", ccsDate, Array("dd", "/", "mm", "/", "yyyy"),
    CCGetRequestParam("CStimestamp", Method))
```

```
            Set Button_Insert = CCCreateButton("Button_Insert")
            Set Button_Update = CCCreateButton("Button_Update")
            Set Button_Delete = CCCreateButton("Button_Delete")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STDid)
        End If
    End Sub
'End CLASSSTD Class_Initialize Event

'CLASSSTD Initialize Method 642-2179AA22
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlSTDid") = CCGetRequestParam("STDid", ccsGET)
        End With
    End Sub
'End CLASSSTD Initialize Method

'CLASSSTD Class_Terminate Event 642-32B647C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End CLASSSTD Class_Terminate Event

'CLASSSTD Validate Method 642-7AE52B4D
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End CLASSSTD Validate Method

'CLASSSTD Operation Method 642-56355E1D
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "Button_Insert"↙
    )
            If Not IsEmpty(CCGetParam("Button_Insert", Empty)) Then
                PressedButton = "Button_Insert"
            ElseIf Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            ElseIf Not IsEmpty(CCGetParam("Button_Delete", Empty)) Then
                PressedButton = "Button_Delete"
            End If
        End If
        Redirect = "CLASS_DataEntry.asp?" & CCGetQueryString("QueryString"↙
    , Array("ccsForm", "Button_Insert", "Button_Update", "Button_Delete"))
        If PressedButton = "Button_Delete" Then
            If NOT Button_Delete.OnClick OR NOT DeleteRow() Then
                Redirect = ""
            Else
                Redirect = "CLASS_DataEntry.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm", "STDid"))
            End If
        ElseIf Validate() Then
            If PressedButton = "Button_Insert" Then
                If NOT Button_Insert.OnClick() OR NOT InsertRow() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_DataEntry.asp?" & CCGetQueryString( ↙
    "QueryString", Array("ccsForm", "STDid"))
```

```
                End If
            ElseIf PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_DataEntry.asp?" & CCGetQueryString( ↙
    "QueryString", Array("ccsForm", "STDid"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End CLASS_DataEntry Operation Method

'CLASS_DataEntry InsertRow Method 642-84057BCE
    Function InsertRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeInsert", Me)
        If NOT InsertAllowed Then InsertRow = False : Exit Function
        DataSource.STDid.Value = STDid.Value
        DataSource.Insert(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInsert", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Insert Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        InsertRow = (Errors.Count = 0)
    End Function
'End CLASS_DataEntry InsertRow Method

'CLASS_DataEntry UpdateRow Method 642-04ED4431
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.STDid.Value = STDid.Value
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End CLASS_DataEntry UpdateRow Method

'CLASS_DataEntry DeleteRow Method 642-1BC28773
    Function DeleteRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeDelete", Me)
        If NOT DeleteAllowed Then DeleteRow = False : Exit Function
        DataSource.Delete(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterDelete", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Delete Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
```

```
        End If
        DeleteRow = (Errors.Count = 0)
    End Function
'End CLASSSTD DeleteRow Method

'CLASSSTD Show Method 642-BBFA30CE
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "CLASSSTD" & IIf(EditMode,↙
     ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(STDid, CStimestamp, Button_Insert, Button_Update,          ↙
    Button_Delete))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record CLASSSTD", DataSource.Connection.↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        STDid.Value = Recordset.Fields("STDid")
                    End If
                    CStimestamp.Value = Recordset.Fields("CStimestamp")
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Insert.Visible = NOT EditMode AND InsertAllowed
        Button_Update.Visible = EditMode AND UpdateAllowed
        Button_Delete.Visible = EditMode AND DeleteAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End CLASSSTD Show Method

End Class 'End CLASSSTD Class 642-A61BA899

Class clsCLASSSTDDataSource 'CLASSSTDDataSource Class 642-80976397

'DataSource Variables 642-B0158377
```

```
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' DataSource fields
    Public STDid
    Public CStimestamp
'End DataSource Variables

'DataSource Class_Initialize Event @42-1F391638
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,        ↙
    Recordset)
        Set CStimestamp = CCCreateField("CStimestamp", "CStimestamp",      ↙
    ccsDate, Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":"↙
    , "ss"), Recordset)
        Fields.AddFields Array(STDid, CStimestamp)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *   " & _
        "FROM CLASSSTD"
        CountSQL = "SELECT COUNT(*)   " & _
        "FROM CLASSSTD"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method @42-7E709E1C
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlSTDid", ccsText, Empty, Empty, Empty,     ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STDid", .          ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
```

```
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 642-3F951BED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event 642-41B4B0CD
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Delete Method 642-79B12504
    Sub Delete(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildDelete", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Delete operation. There↙
     are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "DELETE FROM CLASSSTD WHERE CLASSid = " & CLASS1.CLASSid↙
    .text & " AND " & Where
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteDelete",    ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteDelete", Me)
    End Sub
'End Delete Method

'Update Method 642-1F4BABAA
```

```
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE CLASSSTD SET " & _
            "STDid=" & Connection.ToSQL(STDid.SQLText, STDid.DataType) & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

'Insert Method 642-6D3833A0
    Sub Insert(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildInsert", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        Cmd.SQL = "INSERT INTO CLASSSTD(" & _
            "STDid, " & _
            "CLASSid" & _
        ") VALUES (" & _
            Connection.ToSQL(STDid.SQLText, STDid.DataType) & ", " & _
            class1.classid.text & _
        ")"
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteInsert",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteInsert", Me)
    End Sub
'End Insert Method

End Class 'End CLASSSTDDataSource Class 642-A61BA892


Class clsGridStudent1 'Student1 Class 629-4BAA6B1D

'Student1 Variables 629-9E6D020F

    ' Public variables
    Public ComponentName, CCSEvents
    Public Visible, Errors
    Public DataSource, PageSize
    Public PageNumber
    Public Command
    Public TemplateBlock
    Public Recordset

    Private CCSEventResult

    ' Grid Controls
    Public StaticControls, RowControls
    Dim Student1_TotalRecords
    Dim STDadmission
    Dim STDyear
    Dim STDid
```

```
    Dim STDname
    Dim Navigator
'End Student1 Variables

'Student1 Class_Initialize Event {E29-41EE6103
    Private Sub Class_Initialize()
        ComponentName = "Student1"
        Visible = True
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Errors = New clsErrors
        Set DataSource = New clsStudent1DataSource
        Set Command = New clsCommand
        PageSize = CCGetParam(ComponentName & "PageSize", Empty)
        If IsNumeric(PageSize) And Len(PageSize) > 0 Then
            If PageSize=0 Then Errors.AddError("(CCS06) Invalid page size.↙
    ")
            If PageSize > 50 Then PageSize = 50
        End If
        If NOT IsNumeric(PageSize) OR IsEmpty(PageSize) Then _
            PageSize = 50 _
        Else _
            PageSize = CInt(PageSize)
        PageNumber = CInt(CCGetParam(ComponentName & "Page", 1))

        Set STDadmission = CCCreateControl(ccsLabel, "STDadmission",        ↙
    "STDadmission", ccsInteger, Array(True, False, "", False, "", Array("0↙
    ", "0", "0", "0", "0", "0"), Empty, 1, True, "", False),                ↙
    CCGetRequestParam("STDadmission", ccsGet))
        Set STDyear = CCCreateControl(ccsLabel, "STDyear", "STDyear",       ↙
    ccsDate, Array("yy"), CCGetRequestParam("STDyear", ccsGet))
        Set STDid = CCCreateControl(ccsLink, "STDid", "STDid", ccsText,     ↙
    Empty, CCGetRequestParam("STDid", ccsGet))
        Set STDname = CCCreateControl(ccsLabel, "STDname", "STDname",       ↙
    ccsText, Empty, CCGetRequestParam("STDname", ccsGet))
        Set Student1_TotalRecords = CCCreateControl(ccsLabel,               ↙
    "Student1_TotalRecords", "Student1_TotalRecords", ccsText, Empty,       ↙
    CCGetRequestParam("Student1_TotalRecords", ccsGet))
        Set Navigator = CCCreateNavigator(ComponentName, "Navigator",       ↙
    FileName, 10, tpSimple)
    End Sub
'End Student1 Class_Initialize Event

'Student1 Initialize Method {19-593E5952
    Sub Initialize(objConnection)
        If NOT Visible Then Exit Sub

        Set DataSource.Connection = objConnection
        DataSource.PageSize = PageSize
        DataSource.AbsolutePage = PageNumber
    End Sub
'End Student1 Initialize Method

'Student1 Class_Terminate Event {19-D1518398
    Private Sub Class_Terminate()
        Set DataSource = Nothing
        Set Command = Nothing
        Set Errors = Nothing
    End Sub
'End Student1 Class_Terminate Event

'Student1 Show Method {19-53B15207
    Sub Show(Tpl)
        If NOT Visible Then Exit Sub

        Dim RecordCounter, ShownRecords
        Dim RowBlock

        With DataSource
```

```
            .Parameters("urlCLASSid") = CCGetRequestParam("CLASSid",
    ccsGET)
        End With

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)

        Set TemplateBlock = Tpl.Block("Grid " & ComponentName)
        Set RowBlock = TemplateBlock.Block("Row")
        Set StaticControls = CCCreateCollection(TemplateBlock, Null,
    ccsParseOverwrite, _
            Array(Student1_TotalRecords, Navigator))
        Set RowControls = CCCreateCollection(RowBlock, Null,
    ccsParseAccumulate, _
            Array(STDadmission, STDyear, STDid, STDname))

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If NOT Visible Then Exit Sub

        Errors.AddErrors DataSource.Errors
        If Errors.Count > 0 Then
            TemplateBlock.HTML = CCFormatError("Grid Student1", Errors)
        Else

            ' Show NoRecords block if no records are found
            If Recordset.EOF Then
                TemplateBlock.Block("NoRecords").Parse ccsParseOverwrite
            End If
            While NOT Recordset.EOF AND ShownRecords < PageSize
                STDadmission.Value = Recordset.Fields("STDadmission")
                STDyear.Value = Recordset.Fields("STDyear")
                STDid.Value = Recordset.Fields("STDid")
                STDid.Parameters = CCGetQueryString("QueryString", Array(
    "ccsForm"))
                STDid.Parameters = CCAddParam(STDid.Parameters, "STDid",
    Recordset.Fields("STDid_param1"))
                STDid.Page = "CLASS_DataEntry.asp"
                STDname.Value = Recordset.Fields("STDname")
                CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShowRow",
    Me)
                RowControls.Show
                Recordset.MoveNext
                ShownRecords = ShownRecords + 1
            Wend

            Navigator.SetDataSource Recordset
            StaticControls.Show
        End If

    End Sub
'End Student1 Show Method

End Class 'End Student1 Class [illegible]

Class clsStudent1DataSource 'Student1DataSource Class [illegible]

'DataSource Variables [illegible]
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet
```

```
    Private CurrentOperation
    Private CCSEventResult

    ' DataSource Fields
    Public STDadmission
    Public STDyear
    Public STDid
    Public STDid_param1
    Public STDname
'End DataSource Variables

'DataSource Class_Initialize Event @29-4AD377F9
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDadmission = CCCreateField("STDadmission", "STDadmissionNum"↙
    , ccsInteger, Empty, Recordset)
        Set STDyear = CCCreateField("STDyear", "STDstampdate", ccsDate,   ↙
    Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"), ↙
    Recordset)
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,       ↙
    Recordset)
        Set STDid_param1 = CCCreateField("STDid_param1", "STDid", ccsText,↙
     Empty, Recordset)
        Set STDname = CCCreateField("STDname", "STDname", ccsText, Empty, ↙
    Recordset)
        Fields.AddFields Array(STDadmission, STDyear, STDid, STDid_param1,↙
     STDname)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM lstStudenList"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM lstStudenList"
        Where = ""
        Order = "STDname"
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method @29-45[illegible]7E4
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlCLASSid", ccsFloat, Empty, Empty, Empty, ↙
    False
            .Criterion(1) = .Operation(opEqual, False, "CLASSid", .       ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
```

```
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method {19-F17016E0}
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.CountSQL = CountSQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {19-4184B0E0}
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End StudentDataSource Class {19-A61BAE9E}


%>
```

```
<%
'Include Common Files @1-C6EDE132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-13653708
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim ClassSearch
Dim Class1

Redirect = ""
TemplateFileName = "CLASS_List.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "CLASS_List.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-82717544
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set ClassSearch = new clsRecordClassSearch
Set Class1 = New clsGridClass1
Class1.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-5803EB3E
ClassSearch.Operation
'End Execute Components

'Go to destination page @1-63352433
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0159E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
```

```
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page 61-7C7FA1010
ClassSearch.Show Tpl
Class1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page 61-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub 61-22B36ACA
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set ClassSearch = Nothing
    Set Class1 = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordClassSearch 'ClassSearch Class 61-A6B451D0

'ClassSearch Variables 61-35572E53

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim Button_DoSearch
    Dim Button2
    Dim Button1
'End ClassSearch Variables

'ClassSearch Class_Initialize Event 61-F9B96940
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode
```

```
        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "ClassSearch")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "ClassSearch"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set Button_DoSearch = CCCreateButton("Button_DoSearch")
            Set Button2 = CCCreateButton("Button2")
            Set Button1 = CCCreateButton("Button1")
            Set ValidatingControls = new clsControls
        End If
    End Sub
'End ClassSearch Class_Initialize Event

'ClassSearch Class_Terminate Event 613-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End ClassSearch Class_Terminate Event

'ClassSearch Validate Method 613-7AE52642
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End ClassSearch Validate Method

'ClassSearch Operation Method 613-9E9E8433
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoSearch"
            If Not IsEmpty(CCGetParam("Button_DoSearch", Empty)) Then
                PressedButton = "Button_DoSearch"
            ElseIf Not IsEmpty(CCGetParam("Button2", Empty)) Then
                PressedButton = "Button2"
            ElseIf Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            End If
        End If
        Redirect = "CLASS_List.asp"
        If Validate() Then
            If PressedButton = "Button_DoSearch" Then
                If NOT Button_DoSearch.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_DataEntry.asp?" & CCGetQueryString( ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button2" Then
                If NOT Button2.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STAFF_StdList.asp?" & CCGetQueryString(   ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
```

```
                    Redirect = ""
                Else
                    Redirect = "STAFF_options.asp?" & CCGetQueryString(   ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End ClassSearch Operation Method

'ClassSearch Show Method 013-24625125
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.            ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "ClassSearch" & IIf      ↙
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,           ↙
    ccsParseOverwrite, _
            Array(Button_DoSearch, Button2, Button1))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End ClassSearch Show Method

End Class 'End ClassSearch Class 013-A61BAE9E

Class clsGridClass1 'Class1 Class 013-79CB6EA4

'Class1 Variables 013-2C1253B9

    ' Public variables
    Public ComponentName, CCSEvents
    Public Visible, Errors
    Public DataSource, PageSize
    Public PageNumber
    Public Command
    Public TemplateBlock
    Public ActiveSorter, SortingDirection
    Public Recordset

    Private CCSEventResult

    ' Grid Controls
    Public StaticControls, RowControls
    Dim Sorter_CLASSid
    Dim Sorter_CLASSname
    Dim Sorter_CLASSteacher
```

```
    Dim Sorter_CLASSyear
    Dim CLASSid
    Dim CLASSname
    Dim CLASSteacher
    Dim CLASSyear
    Dim Navigator
'End Class1 Variables

'Class1 Class_Initialize Event @12-B2E967E6
    Private Sub Class_Initialize()
        ComponentName = "Class1"
        Visible = True
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Errors = New clsErrors
        Set DataSource = New clsClass1DataSource
        Set Command = New clsCommand
        PageSize = CCGetParam(ComponentName & "PageSize", Empty)
        If IsNumeric(PageSize) And Len(PageSize) > 0 Then
            If PageSize=0 Then Errors.AddError("(CCS06) Invalid page size.↙
    ")
            If PageSize > 20 Then PageSize = 20
        End If
        If NOT IsNumeric(PageSize) OR IsEmpty(PageSize) Then _
            PageSize = 20 _
        Else _
            PageSize = CInt(PageSize)
        PageNumber = CInt(CCGetParam(ComponentName & "Page", 1))
        ActiveSorter = CCGetParam("Class1Order", Empty)
        SortingDirection = CCGetParam("Class1Dir", Empty)
        If NOT(SortingDirection = "ASC" OR SortingDirection = "DESC") Then↙

     _
            SortingDirection = Empty

        Set CLASSid = CCCreateControl(ccsLink, "CLASSid", "CLASSid",      ↙
    ccsFloat, Array(True, False, "", False, "", Array("0", "0", "0", "0"),↙
     Empty, 1, True, "", False), CCGetRequestParam("CLASSid", ccsGet))
        Set CLASSname = CCCreateControl(ccsLabel, "CLASSname", "CLASSname"↙
    , ccsText, Empty, CCGetRequestParam("CLASSname", ccsGet))
        Set CLASSteacher = CCCreateControl(ccsLabel, "CLASSteacher",      ↙
    "CLASSteacher", ccsText, Empty, CCGetRequestParam("CLASSteacher",     ↙
    ccsGet))
        Set CLASSyear = CCCreateControl(ccsLabel, "CLASSyear", "CLASSyear"↙
    , ccsDate, Array("dd", "/", "mm", "/", "yyyy"), CCGetRequestParam(    ↙
    "CLASSyear", ccsGet))
        Set Sorter_CLASSid = CCCreateSorter("Sorter_CLASSid", Me,         ↙
    FileName)
        Set Sorter_CLASSname = CCCreateSorter("Sorter_CLASSname", Me,     ↙
    FileName)
        Set Sorter_CLASSteacher = CCCreateSorter("Sorter_CLASSteacher", Me↙
    , FileName)
        Set Sorter_CLASSyear = CCCreateSorter("Sorter_CLASSyear", Me,     ↙
    FileName)
        Set Navigator = CCCreateNavigator(ComponentName, "Navigator",     ↙
    FileName, 10, tpCentered)
    End Sub
'End Class1 Class_Initialize Event

'Class1 Initialize Method @16-1ABA3995
    Sub Initialize(objConnection)
        If NOT Visible Then Exit Sub

        Set DataSource.Connection = objConnection
        DataSource.PageSize = PageSize
        DataSource.SetOrder ActiveSorter, SortingDirection
        DataSource.AbsolutePage = PageNumber
    End Sub
'End Class1 Initialize Method
```

```
'Class1 Class_Terminate Event {12-E1518296
    Private Sub Class_Terminate()
        Set DataSource = Nothing
        Set Command = Nothing
        Set Errors = Nothing
    End Sub
'End Class1 Class_Terminate Event

'Class1 Show Method {12-52263913
    Sub Show(Tpl)
        If NOT Visible Then Exit Sub

        Dim RecordCounter, ShownRecords
        Dim RowBlock

        With DataSource
            .Parameters("urls_CLASSid") = CCGetRequestParam("s_CLASSid",  ↙
    ccsGET)
            .Parameters("urls_CLASSname") = CCGetRequestParam("s_CLASSname↙
    ", ccsGET)
            .Parameters("urls_CLASSteacher") = CCGetRequestParam(         ↙
    "s_CLASSteacher", ccsGET)
            .Parameters("urls_CLASSyear") = CCGetRequestParam("s_CLASSyear↙
    ", ccsGET)
        End With

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)

        Set TemplateBlock = Tpl.Block("Grid " & ComponentName)
        Set RowBlock = TemplateBlock.Block("Row")
        Set StaticControls = CCCreateCollection(TemplateBlock, Null,      ↙
    ccsParseOverwrite, _
            Array(Sorter_CLASSid, Sorter_CLASSname, Sorter_CLASSteacher,  ↙
    Sorter_CLASSyear, Navigator))
        Set RowControls = CCCreateCollection(RowBlock, Null,              ↙
    ccsParseAccumulate, _
            Array(CLASSid, CLASSname, CLASSteacher, CLASSyear))

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If NOT Visible Then Exit Sub

        Errors.AddErrors DataSource.Errors
        If Errors.Count > 0 Then
            TemplateBlock.HTML = CCFormatError("Grid Class1", Errors)
        Else

            ' Show NoRecords block if no records are found
            If Recordset.EOF Then
                TemplateBlock.Block("NoRecords").Parse ccsParseOverwrite
            End If
            While NOT Recordset.EOF AND ShownRecords < PageSize
                CLASSid.Value = Recordset.Fields("CLASSid")
                CLASSid.Parameters = CCGetQueryString("QueryString", Array↙
    ("ccsForm"))
                CLASSid.Parameters = CCAddParam(CLASSid.Parameters,       ↙
    "CLASSid", Recordset.Fields("CLASSid_param1"))
                CLASSid.Page = "CLASS_DataEntry.asp"
                CLASSname.Value = Recordset.Fields("CLASSname")
                CLASSteacher.Value = Recordset.Fields("CLASSteacher")
                CLASSyear.Value = Recordset.Fields("CLASSyear")
                CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShowRow", ↙
    Me)
                RowControls.Show
                Recordset.MoveNext
                ShownRecords = ShownRecords + 1
            Wend
            Navigator.SetDataSource Recordset
```

```
            StaticControls.Show
        End If

    End Sub
'End Class1 Show Method

End Class 'End Class1 Class 612-A61BA896

Class clsClass1DataSource 'Class1DataSource Class 612-24994BA8

'DataSource Variables 612-F9793291
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public CLASSid
    Public CLASSid_param1
    Public CLASSname
    Public CLASSteacher
    Public CLASSyear
'End DataSource Variables

'DataSource Class_Initialize Event 611-3B71B395
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set CLASSid = CCCreateField("CLASSid", "CLASSid", ccsFloat, Empty,↙
     Recordset)
        Set CLASSid_param1 = CCCreateField("CLASSid_param1", "CLASSid",   ↙
    ccsText, Empty, Recordset)
        Set CLASSname = CCCreateField("CLASSname", "CLASSname", ccsText,  ↙
    Empty, Recordset)
        Set CLASSteacher = CCCreateField("CLASSteacher", "CLASSteacher",  ↙
    ccsText, Empty, Recordset)
        Set CLASSyear = CCCreateField("CLASSyear", "CLASSyear", ccsDate,  ↙
    Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"), ↙
    Recordset)
        Fields.AddFields Array(CLASSid, CLASSid_param1, CLASSname,        ↙
    CLASSteacher, CLASSyear)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing
        Orders = Array( _
            Array("Sorter_CLASSid", "CLASSid", ""), _
            Array("Sorter_CLASSname", "CLASSname", ""), _
            Array("Sorter_CLASSteacher", "CLASSteacher", ""), _
            Array("Sorter_CLASSyear", "CLASSyear", ""))

        SQL = "SELECT *  " & _
        "FROM Class"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Class"
        Where = ""
```

```
        Order = "CLASSid"
    End Sub
'End DataSource Class_Initialize Event

'SetOrder Method @12-68FC9574
    Sub SetOrder(Column, Direction)
        Order = Recordset.GetOrder(Order, Column, Direction, Orders)
    End Sub
'End SetOrder Method

'BuildTableWhere Method @12-00B4537F
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urls_CLASSid", ccsFloat, Empty, Empty, Empty↙
    , False
            .AddParameter 2, "urls_CLASSname", ccsText, Empty, Empty,     ↙
    Empty, False
            .AddParameter 3, "urls_CLASSteacher", ccsText, Empty, Empty,  ↙
    Empty, False
            .AddParameter 4, "urls_CLASSyear", ccsDate, Array("dd", "/",  ↙
    "mm", "/", "yyyy"), Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":"↙
    , "nn", ":", "ss"), Empty, False
            .Criterion(1) = .Operation(opEqual, False, "CLASSid", .       ↙
    getParamByID(1))
            .Criterion(2) = .Operation(opContains, False, "CLASSname", .  ↙
    getParamByID(2))
            .Criterion(3) = .Operation(opContains, False, "CLASSteacher", ↙
    .getParamByID(3))
            .Criterion(4) = .Operation(opEqual, False, "CLASSyear", .     ↙
    getParamByID(4))
            .AssembledWhere = .opAND(False, .opAND(False, .opAND(False, . ↙
    Criterion(1), .Criterion(2)), .Criterion(3)), .Criterion(4))
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method @11-F1A16EC
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.CountSQL = CountSQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",   ↙
```

```
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event @12-41B4B08D
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End Class1DataSource Class @12-A61BA892


%>
```

```
<%
'Include Common Files {1-C6CCE13}
%>
<!-- #INCLUDE FILE="./Common.asp"-->
<!-- #INCLUDE FILE="./Cache.asp" -->
<!-- #INCLUDE FILE="./Templates.asp" -->
<!-- #INCLUDE FILE="./Grammar.asp" -->
<!-- #INCLUDE FILE="./Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page {1-EB2560C2}
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Login

Redirect = ""
TemplateFileName = "LoginPage.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "LoginPage.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects {1-C2195180}
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Login = new clsRecordLogin
Login.Initialize DBConnection1

' Events
%>
<!-- #INCLUDE FILE="LoginPage_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components {1-6AA0CE03}
Login.Operation
'End Execute Components

'Go to destination page {1-6D35A42D}
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page
```

```
'Initialize HTML Template @1-0155E10E
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-C559E43E
Login.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C68
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-89FD8590
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Login = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordLogin 'Login Class @2-C3C9A674

'Login Variables @2-25EAC7EE

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim password
    Dim Button_DoLogin
'End Login Variables

'Login Class Initialize Event @2-8041A846
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsLoginDataSource
```

```
        Set Command = New clsCommand
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Login")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Login"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set password = CCCreateControl(ccsTextBox, "password",
    "password", ccsText, Empty, CCGetRequestParam("password", Method))
            password.Required = True
            Set Button_DoLogin = CCCreateButton("Button_DoLogin")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(password)
        End If
    End Sub
'End Login_Class_Initialize Event

'Login_Initialize Method @2-36217959
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
        End With
    End Sub
'End Login_Initialize Method

'Login_Class_Terminate Event @2-33B54729
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Login_Class_Terminate Event

'Login_Validate Method @2-7AE51645
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Login_Validate Method

'Login_Operation Method @2-7A439837
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoLogin"
            If Not IsEmpty(CCGetParam("Button_DoLogin", Empty)) Then
                PressedButton = "Button_DoLogin"
            End If
        End If
        Redirect = FileName & ""
        If Validate() Then
            If PressedButton = "Button_DoLogin" Then
                If NOT Button_DoLogin.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "StudentList.asp?" & CCGetQueryString(
```

```
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Login Operation Method

'Login Show Method 03-41280C1E
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Login" & IIf(EditMode, ":↙
    Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(password, Button_DoLogin))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record Login", DataSource.Connection.      ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then

                    End If
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Login Show Method

End Class 'End Login Class 01-A610AF9E

Class clsLoginDataSource 'LoginDataSource Class 02-5D4A1E16

'DataSource Variables 01-09B05153
```

```
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
'End DataSource Variables

'DataSource Class_Initialize Event @2-52A1C29E
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Fields.AddFields Array()

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method @2-96E5A98F
    Public Sub BuildTableWhere()
    End Sub
'End BuildTableWhere Method

'Open Method @1-B139F2E3
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method
```

```
'DataSource Class_Terminate Event 02-41B4D082
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End LoginDataSource Class 02-A61BA892


%>
```

```
<%
'BindEvents Method 11-3261B678
Sub BindEvents()
    Set Login.Button_DoLogin.CCSEvents("OnClick") = GetRef(
    "Login_Button_DoLogin_OnClick")
End Sub
'End BindEvents Method

Function Login_Button_DoLogin_OnClick() 'Login_Button_DoLogin_OnClick 03-
    ACB4691A

'Login 04-AA430BB1
    With Login
        If NOT CCLoginUser(.password.Value) Then
            .Errors.addError("Login or Password is incorrect.")
            Login_Button_DoLogin_OnClick = False
            .password.Value = ""
        Else
            If Not IsEmpty(CCGetParam("ret_link", Empty)) Then _
                Redirect = CCGetParam("ret_link", Empty)
            Login_Button_DoLogin_OnClick = True
        End If
    End With
'End Login

End Function 'Close Login_Button_DoLogin_OnClick 03-54034B28


%>
```

```
<%
'Include Common Files @1-2BEDE132
%>
<!-- #INCLUDE FILE="./Common.asp"-->
<!-- #INCLUDE FILE="./Cache.asp" -->
<!-- #INCLUDE FILE="./Template.asp" -->
<!-- #INCLUDE FILE="./Crypter.asp" -->
<!-- #INCLUDE FILE="./Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-6EDD18FB
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Page controls
Dim Login

Redirect = ""
TemplateFileName = "STD_login.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STD_login.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-B13A506F

' Controls
Set Login = new clsRecordLogin

' Events
%>
<!-- #INCLUDE FILE="STD_login_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-6AA0CE03
Login.Operation
'End Execute Components

'Go to destination page @1-6D95F4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-2159E4B0
Login.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-DE8F79E6
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    Set Login = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordLogin 'Login Class @2-23799874

'Login Variables @2-552903FE

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim login
    Dim password
    Dim Button_DoLogin
'End Login Variables

'Login Class Initialize Event @2-AB01F770
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Login")
            End If
            If UBound(OperationMode) > 0 Then
```

```
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Login"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set login = CCCreateControl(ccsTextBox, "login", "login",      ↙
    ccsText, Empty, CCGetRequestParam("login", Method))
            login.Required = True
            Set password = CCCreateControl(ccsTextBox, "password",         ↙
    "password", ccsText, Empty, CCGetRequestParam("password", Method))
            password.Required = True
            Set Button_DoLogin = CCCreateButton("Button_DoLogin")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(login, password)
        End If
    End Sub
'End Login Class_Initialize Event

'Login Class_Terminate Event 62-32B84709
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Login Class_Terminate Event

'Login Validate Method 62-7AE5264D
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Login Validate Method

'Login Operation Method 62-9046F5B3
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoLogin"
            If Not IsEmpty(CCGetParam("Button_DoLogin", Empty)) Then
                PressedButton = "Button_DoLogin"
            End If
        End If
        Redirect = FileName & ""
        If Validate() Then
            If PressedButton = "Button_DoLogin" Then
                If NOT Button_DoLogin.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STD_DataEntry.asp?" & CCGetQueryString(    ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Login Operation Method

'Login Show Method 62-C4058633
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.              ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Login" & IIf(EditMode, ":↙
    Edit", ""))
```

```
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,            ↙
    ccsParseOverwrite, _
            Array(login, password, Button_DoLogin))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Login Show Method

End Class 'End Login Class C2-A61BA892


%>
```

```
<%
'BindEvents Method @1-3261B67B
Sub BindEvents()
    Set Login.Button_DoLogin.CCSEvents("OnClick") = GetRef(
    "Login_Button_DoLogin_OnClick")
End Sub
'End BindEvents Method

Function Login_Button_DoLogin_OnClick() 'Login_Button_DoLogin_OnClick @3-
    ADB4691A

'Login @4-AA430BB1
    With Login
        If NOT CCLoginUser(.login.Value, .password.Value) Then
            .Errors.addError("Login or Password is incorrect.")
            Login_Button_DoLogin_OnClick = False
            .password.Value = ""
        Else
            If Not IsEmpty(CCGetParam("ret_link", Empty)) Then _
                Redirect = CCGetParam("ret_link", Empty)
            Login_Button_DoLogin_OnClick = True
        End If
    End With
'End Login

End Function 'Close Login_Button_DoLogin_OnClick @3-54C34B2F


%>
```

```
<%
'Include Common Files @1-C6EDE132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Connection.asp" -->
<!-- #INCLUDE FILE=".\Navigation.asp" -->
<%
'End Include Common Files

'Initialize Page @1-BC70ECAF
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Page controls
Dim Login

Redirect = ""
TemplateFileName = "STDlogin.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STDlogin.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-1BA59A41

' Controls
Set Login = new clsRecordLogin

' Events
%>
<!-- #INCLUDE FILE="STDlogin_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-6AA0CE03
Login.Operation
'End Execute Components

'Go to destination page @1-6D35F4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-C9590F4A
Login.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-3E8F79E6
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    Set Login = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordLogin 'Login Class @1-23C90874

'Login Variables @1-55290FFE

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim login
    Dim password
    Dim Button_DoLogin
'End Login Variables

'Login Class_Initialize Event @1-AB01F776
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Login")
            End If
            If UBound(OperationMode) > 0 Then
```

```
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Login"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set login = CCCreateControl(ccsTextBox, "login", "login",      ↙
    ccsText, Empty, CCGetRequestParam("login", Method))
            login.Required = True
            Set password = CCCreateControl(ccsTextBox, "password",         ↙
    "password", ccsText, Empty, CCGetRequestParam("password", Method))
            password.Required = True
            Set Button_DoLogin = CCCreateButton("Button_DoLogin")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(login, password)
        End If
    End Sub
'End Login Class_Initialize Event

'Login Class_Terminate Event 62-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Login Class_Terminate Event

'Login Validate Method 62-7AE5264D
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Login Validate Method

'Login Operation Method 62-4006E3F6
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoLogin"
            If Not IsEmpty(CCGetParam("Button_DoLogin", Empty)) Then
                PressedButton = "Button_DoLogin"
            End If
        End If
        Redirect = FileName & ""
        If Validate() Then
            If PressedButton = "Button_DoLogin" Then
                If NOT Button_DoLogin.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = ServerURL & PathToCurrentPage &             ↙
    "STD_DataEntry.asp?" & CCGetQueryString("QueryString", Array("ccsForm"↙
    ))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Login Operation Method

'Login Show Method 62-C4058633
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.              ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Login" & IIf(EditMode, ":↙
```

```
    Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,            ↙
    ccsParseOverwrite, _
            Array(login, password, Button_DoLogin))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Login Show Method

End Class 'End Login Class 62-A61BA692


%>
```

```
<%
'BindEvents Method 01-3261B678
Sub BindEvents()
    Set Login.Button_DoLogin.CCSEvents("OnClick") = GetRef(
    "Login_Button_DoLogin_OnClick")
End Sub
'End BindEvents Method

Function Login_Button_DoLogin_OnClick() 'Login_Button_DoLogin_OnClick 03-
    ADB4691A

'Login 04-AA430BB1
    With Login
        If NOT CCLoginUser(.login.Value, .password.Value) Then
            .Errors.addError("Login or Password is incorrect.")
            Login_Button_DoLogin_OnClick = False
            .password.Value = ""
        Else
            If Not IsEmpty(CCGetParam("ret_link", Empty)) Then _
                Redirect = CCGetParam("ret_link", Empty)
            Login_Button_DoLogin_OnClick = True
        End If
    End With
'End Login

End Function 'Close Login_Button_DoLogin_OnClick 03-54034B28


%>
```

```
<%
'Include Common Files @1-D6EDD132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-6E756D18
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Student1

Redirect = ""
TemplateFileName = "StudentList.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "StudentList.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-50DF5918
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Student1 = New clsGridStudent1
Student1.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Go to destination page @1-6D35F4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template
```

```
'Show Page @1-D37E5002
Student1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-FB1B42B7
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Student1 = Nothing
End Sub
'End UnloadPage Sub


Class clsGridStudent1 'Student1 Class @16-4BAA6B1D

'Student1 Variables @16-F3643518

    ' Public variables
    Public ComponentName, CCSEvents
    Public Visible, Errors
    Public DataSource, PageSize
    Public PageNumber
    Public Command
    Public TemplateBlock
    Public ActiveSorter, SortingDirection
    Public Recordset

    Private CCSEventResult

    ' Grid Controls
    Public StaticControls, RowControls
    Dim Sorter_STDid
    Dim Sorter_STDname
    Dim Sorter_STDpwd
    Dim STDid
    Dim STDname
    Dim STDpwd
    Dim Navigator
'End Student1 Variables

'Student1 Class_Initialize Event @16-55373647
    Private Sub Class_Initialize()
        ComponentName = "Student1"
        Visible = True
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Errors = New clsErrors
        Set DataSource = New clsStudent1DataSource
        Set Command = New clsCommand
        PageSize = CCGetParam(ComponentName & "PageSize", Empty)
        If IsNumeric(PageSize) And Len(PageSize) > 0 Then
            If PageSize=0 Then Errors.AddError("(CCS06) Invalid page size.
    ")
            If PageSize > 20 Then PageSize = 20
        End If
        If NOT IsNumeric(PageSize) OR IsEmpty(PageSize) Then _
            PageSize = 20 _
```

```
        Else _
            PageSize = CInt(PageSize)
        PageNumber = CInt(CCGetParam(ComponentName & "Page", 1))
        ActiveSorter = CCGetParam("Student1Order", Empty)
        SortingDirection = CCGetParam("Student1Dir", Empty)
        If NOT(SortingDirection = "ASC" OR SortingDirection = "DESC") Then↙
 _
            SortingDirection = Empty

        Set STDid = CCCreateControl(ccsLink, "STDid", "STDid", ccsText,    ↙
    Empty, CCGetRequestParam("STDid", ccsGet))
        Set STDname = CCCreateControl(ccsLabel, "STDname", "STDname",      ↙
    ccsText, Empty, CCGetRequestParam("STDname", ccsGet))
        Set STDpwd = CCCreateControl(ccsLabel, "STDpwd", "STDpwd", ccsDate↙
    , Array("mm", "/", "yyyy"), CCGetRequestParam("STDpwd", ccsGet))
        Set Sorter_STDid = CCCreateSorter("Sorter_STDid", Me, FileName)
        Set Sorter_STDname = CCCreateSorter("Sorter_STDname", Me,          ↙
    FileName)
        Set Sorter_STDpwd = CCCreateSorter("Sorter_STDpwd", Me, FileName)
        Set Navigator = CCCreateNavigator(ComponentName, "Navigator",      ↙
    FileName, 10, tpCentered)
    End Sub
'End Student1 Class_Initialize Event

'Student1 Initialize Method {16-9AEA397
    Sub Initialize(objConnection)
        If NOT Visible Then Exit Sub

        Set DataSource.Connection = objConnection
        DataSource.PageSize = PageSize
        DataSource.SetOrder ActiveSorter, SortingDirection
        DataSource.AbsolutePage = PageNumber
    End Sub
'End Student1 Initialize Method

'Student1 Class_Terminate Event {16-E1928F06
    Private Sub Class_Terminate()
        Set DataSource = Nothing
        Set Command = Nothing
        Set Errors = Nothing
    End Sub
'End Student1 Class_Terminate Event

'Student1 Show Method {16-A390BA76
    Sub Show(Tpl)
        If NOT Visible Then Exit Sub

        Dim RecordCounter, ShownRecords
        Dim RowBlock


        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)

        Set TemplateBlock = Tpl.Block("Grid " & ComponentName)
        Set RowBlock = TemplateBlock.Block("Row")
        Set StaticControls = CCCreateCollection(TemplateBlock, Null,       ↙
    ccsParseOverwrite, _
            Array(Sorter_STDid, Sorter_STDname, Sorter_STDpwd, Navigator))
        Set RowControls = CCCreateCollection(RowBlock, Null,               ↙
    ccsParseAccumulate, _
            Array(STDid, STDname, STDpwd))

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If NOT Visible Then Exit Sub

        Errors.AddErrors DataSource.Errors
        If Errors.Count > 0 Then
```

```
            TemplateBlock.HTML = CCFormatError("Grid Student1", Errors)
        Else

            ' Show NoRecords block if no records are found
            If Recordset.EOF Then
                TemplateBlock.Block("NoRecords").Parse ccsParseOverwrite
            End If
            While NOT Recordset.EOF AND ShownRecords < PageSize
                STDid.Value = Recordset.Fields("STDid")
                STDid.Parameters = CCGetQueryString("QueryString", Array( ↙
    "ccsForm"))
                STDid.Parameters = CCAddParam(STDid.Parameters, "STDid",  ↙
    Recordset.Fields("STDid_param1"))
                STDid.Page = "STD_DataEntry.asp"
                STDname.Value = Recordset.Fields("STDname")
                STDpwd.Value = Recordset.Fields("STDpwd")
                CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShowRow", ↙
    Me)
                RowControls.Show
                Recordset.MoveNext
                ShownRecords = ShownRecords + 1
            Wend
            Navigator.SetDataSource Recordset
            StaticControls.Show
        End If

    End Sub
'End Student1 Show Method

End Class 'End Student1 Class @16-A61BA89C

Class clsStudent1DataSource 'Student1DataSource Class @16-E75C3B23

'DataSource Variables @16-B40621B7
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STDid
    Public STDid_param1
    Public STDname
    Public STDpwd
'End DataSource Variables

'DataSource Class_Initialize Event @16-0CAB4633
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,        ↙
    Recordset)
        Set STDid_param1 = CCCreateField("STDid_param1", "STDid", ccsText,↙
     Empty, Recordset)
```

```
        Set STDname = CCCreateField("STDname", "STDname", ccsText, Empty, ↙
    Recordset)
        Set STDpwd = CCCreateField("STDpwd", "STDstampdate", ccsDate,      ↙
    Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"), ↙
    Recordset)
        Fields.AddFields Array(STDid, STDid_param1, STDname, STDpwd)
        Orders = Array( _
            Array("Sorter_STDid", "STDid", ""), _
            Array("Sorter_STDname", "STDname", ""), _
            Array("Sorter_STDpwd", "STDpwd", ""))

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'SetOrder Method @16-68FC9576
    Sub SetOrder(Column, Direction)
        Order = Recordset.GetOrder(Order, Column, Direction, Orders)
    End Sub
'End SetOrder Method

'BuildTableWhere Method @16-98E5A92F
    Public Sub BuildTableWhere()
    End Sub
'End BuildTableWhere Method

'Open Method @16-D0383B6E
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.CountSQL = CountSQL
        Cmd.Where = Where
        Cmd.OrderBy = Order
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event @16-41B4B0E0
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End StudentDataSource Class @16-A61BA892
```

```
%>
```

```
<%
'BindEvents Method C1-E2E72FEA
Sub BindEvents()
    Set Student.Student_TotalRecords.CCSEvents("BeforeShow") = GetRef(
    "Student_Student_TotalRecords_BeforeShow")
End Sub
'End BindEvents Method

Function Student_Student_TotalRecords_BeforeShow()
    'Student_Student_TotalRecords_BeforeShow E7-0D1D9CAE

'Retrieve number of records 68-0D95DE3D
    Student.Student_TotalRecords.Value = Student.DataSource.Recordset.
    RecordCount
'End Retrieve number of records

End Function 'Close Student_Student_TotalRecords_BeforeShow 67-54M34B28


%>
```

```
<%
'Include Common Files @1-C6E3D13C
%>
<!-- #INCLUDE FILE="./Common.asp"-->
<!-- #INCLUDE FILE="./Cache.asp" -->
<!-- #INCLUDE FILE="./Template.asp" -->
<!-- #INCLUDE FILE="./Server.asp" -->
<!-- #INCLUDE FILE="./Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-7A490FBB
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Student

Redirect = ""
TemplateFileName = "STD_Password.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STD_Password.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-A0FAB129
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Student = new clsRecordStudent
Student.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-CB360911
Student.Operation
'End Execute Components

'Go to destination page @1-6035F4F0
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-4BA0BEF5
Student.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C6E
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-A4B96131
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Student = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordStudent 'Student Class @1-44D07E1F

'Student Variables @1-CCB4D94E

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public UpdateAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim STDid
    Dim STDpwd
    Dim Button_Update
'End Student Variables

'Student Class_Initialize Event @1-0F5E7F9A
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsStudentDataSource
        Set Command = New clsCommand
        UpdateAllowed = True
        Dim Method
        Dim OperationMode
```

```
        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Student")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Student"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set STDid = CCCreateControl(ccsLabel, "STDid", "STDid",
    ccsText, Empty, CCGetRequestParam("STDid", Method))
            Set STDpwd = CCCreateControl(ccsTextBox, "STDpwd", "STDpwd",
    ccsText, Empty, CCGetRequestParam("STDpwd", Method))
            STDpwd.Required = True
            Set Button_Update = CCCreateButton("Button_Update")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STDpwd)
        End If
    End Sub
'End Student Class_Initialize Event

'Student Initialize Method 62-0EE95DAE
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("sesUserID") = Session("UserID")
        End With
    End Sub
'End Student Initialize Method

'Student Class_Terminate Event 62-32B84709
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Student Class_Terminate Event

'Student Validate Method 62-7AE02843
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Student Validate Method

'Student Operation Method 62-10348EA9
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "")
            If Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            End If
        End If
        Redirect = "STD_Password.asp?" & CCGetQueryString("QueryString",
    Array("ccsForm", "Button_Update"))
        If Validate() Then
            If PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
```

```
                    Redirect = "STD_DataEntry.asp?" & CCGetQueryString(    ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Student Operations Method

'Student UpdateRow Method {E2-82A03A83
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.STDpwd.Value = STDpwd.Value
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End Student UpdateRow Method

'Student Show Method {D-9ED4D633
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.              ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Student" & IIf(EditMode, ↙
    ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,             ↙
    ccsParseOverwrite, _
            Array(STDid, STDpwd, Button_Update))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record Student", DataSource.Connection. ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        STDpwd.Value = Recordset.Fields("STDpwd")
                    End If
                    STDid.Value = Recordset.Fields("STDid")
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
```

```
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Update.Visible = EditMode AND UpdateAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Student Show Method

End Class 'End Student Class 62-A61BA892

Class clsStudentDataSource 'StudentDataSource Class 62-2F9B506E

'DataSource Variables 62-36056840
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STDid
    Public STDpwd
'End DataSource Variables

'DataSource Class_Initialize Event 62-4CB68B2E
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,        ↙
    Recordset)
        Set STDpwd = CCCreateField("STDpwd", "STDpwd", ccsText, Empty,     ↙
    Recordset)
        Fields.AddFields Array(STDid, STDpwd)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildSQLWhere Method 62-1D679488
```

```
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "sesUserID", ccsText, Empty, Empty, Empty,    ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STDid", .          ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method {8-DF9518ED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {8-41B4B0ED
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Update Method {8-A5329BF7
    Sub Update(Cmd)
```

```
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.      ↙
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE Student SET " & _
            "STDpwd=" & Connection.ToSQL(STDpwd.SQLText, STDpwd.DataType) ↙
    & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",   ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

End Class 'End StudentDataSource Class 02-A61BA092


%>
```

```
<%
'Include Common Files @1-26E2E132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-5E3E295A
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Student
Dim Logout

Redirect = ""
TemplateFileName = "STD_DataEntry.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STD_DataEntry.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-37253B1F
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Student = new clsRecordStudent
Set Logout = CCCreateControl(ccsLink, "Logout", "Logout", ccsText, Empty, ↙
    CCGetRequestParam("Logout", ccsGet))

Logout.Parameters = CCGetQueryString("QueryString", Array("STDid",        ↙
    "ccsForm"))
Logout.Parameters = CCAddParam(Logout.Parameters, "Logout", "True")
Logout.Page = "STD_End.asp"
Student.Initialize DBConnection1

' Events
%>
<!-- #INCLUDE FILE="STD_DataEntry_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-CB363918
Student.Operation
```

```
'End Events' Connections

'Go to destination page @1-6235E4F2
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-5EF044B4
Student.Show Tpl
Logout.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C61
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-A4B96131
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Student = Nothing
End Sub
'End UnloadPage Sub

'DEL        Sub Insert_Add()
'DEL            CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildInsert"
    , Nothing)
'DEL            Set Add.Connection = Connection1
'DEL            Cmd.CommandGenerator = "odbcDev"
'DEL            Add.CommandType = adTable
'DEL            Cmd.CommandParameters = Empty
'DEL            Add.SQL = "INSERT INTO Students" & _
'DEL              "(SSID, " & _
'DEL                "SSFname, " & _
'DEL                "SSSex, " & _
'DEL                "SSReligion, " & _
'DEL                "SSRace, " & _
'DEL                "SSBirthdate, " & _
'DEL                "SSAge, " & _
'DEL                "SSUser, " & _
'DEL                "SSMiddlelast, " & _
'DEL                "SSFatherName, " & _
'DEL                "SSFatherAdd, " & _
'DEL                "SSGuardianName, " & _
'DEL                "SSGuardianAdd, " & _
'DEL                "SSGuardianType" & _
'DEL            ") VALUES (" & _
'DEL              Session("UserID") & ", " & _
'DEL                Connection1.ToSQL(SSFname.SQLValue, SSFname.DataType) & ",
    " &
```

```
'DEL                Connection.ToSQL(STDsex.SQLText, STDsex.DataType) & ", "↙
    &
'DEL                Connection.ToSQL(STDreligion.SQLText, STDreligion.      ↙
    DataType) & ", " & _
'DEL                Connection.ToSQL(STDrace.SQLText, STDrace.DataType) & ",↙
    " & _
'DEL                Connection.ToSQL(STDbirthcert.SQLText, STDbirthcert.    ↙
    DataType) & ", " & _
'DEL                Connection.ToSQL(STDdob.SQLText, STDdob.DataType) & ", "↙
    & _
'DEL                Connection.ToSQL(STDpob.SQLText, STDpob.DataType) & ", "↙
    & _
'DEL                Connection.ToSQL(STDiniclass.SQLText, STDiniclass.      ↙
    DataType) & ", " & _
'DEL                Connection.ToSQL(STDfatlarrange.SQLText, STDfatlarrange.↙
    DataType) & ", " & _
'DEL                Connection.ToSQL(STDfatlaradd.SQLText, STDfatlaradd.    ↙
    DataType) & ", " & _
'DEL                Connection.ToSQL(STDguardianname.SQLText,               ↙
    STDguardianname.DataType) & ", " & _
'DEL                Connection.ToSQL(STDguardianadd.SQLText, STDguardianadd.↙
    DataType) & ", " & _
'DEL                Connection.ToSQL(STDguardiantype.SQLText,               ↙
    STDguardiantype.DataType) & _
'DEL            ")"
'DEL            CCSEventResult = CCRaiseEvent(CCSEvents,                    ↙
    "BeforeExecuteInsert", Me)
'DEL            Cmd.Execute(Errors)
'DEL            CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteInsert↙
    ", Me)
'DEL        End Sub

Class clsRecordStudent 'Student Class {E0-44D0E61F

'Student Variables {8E-049DAA16

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public UpdateAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim STDid
    Dim STDname
    Dim STDsex
    Dim STDsexDataSource
    Dim STDreligion
    Dim STDrace
    Dim STDbirthcert
    Dim STDdob
    Dim DatePicker_STDdob
    Dim STDpob
    Dim STDiniclass
```

```
    Dim STDfathername
    Dim STDfatheradd
    Dim STDguardianname
    Dim STDguardianadd
    Dim STDguardiantype
    Dim STDstampdate
    Dim Button_Update
    Dim Button1
'End Student Variables

'Student Class_Initialize Event 622-1A3DF456
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsStudentDataSource
        Set Command = New clsCommand
        UpdateAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Student")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Student"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set STDid = CCCreateControl(ccsLabel, "STDid", "STDid",          ↙
    ccsText, Empty, CCGetRequestParam("STDid", Method))
            Set STDname = CCCreateControl(ccsTextBox, "STDname", "STDname"↙
    , ccsText, Empty, CCGetRequestParam("STDname", Method))
            Set STDsexDataSource = CCCreateDataSource(dsTable,               ↙
    DBConnection1, Array("SELECT *  " & _
"FROM Type", "", ""))
            Set STDsex = CCCreateList(ccsListBox, "STDsex", "STDsex",        ↙
    ccsText, CCGetRequestParam("STDsex", Method), STDsexDataSource)
            Set STDreligion = CCCreateControl(ccsTextBox, "STDreligion",     ↙
    "STDreligion", ccsText, Empty, CCGetRequestParam("STDreligion",          ↙
    Method))
            Set STDrace = CCCreateControl(ccsTextBox, "STDrace", "STDrace"↙
    , ccsText, Empty, CCGetRequestParam("STDrace", Method))
            Set STDbirthcert = CCCreateControl(ccsTextBox, "STDbirthcert",↙
     "STDbirthcert", ccsText, Empty, CCGetRequestParam("STDbirthcert",       ↙
    Method))
            Set STDdob = CCCreateControl(ccsTextBox, "STDdob", "STDdob",      ↙
    ccsDate, Array("dd", "/", "mm", "/", "yyyy"), CCGetRequestParam(          ↙
    "STDdob", Method))
            Set DatePicker_STDdob = CCCreateDatePicker("DatePicker_STDdob"↙
    , "Student", "STDdob")
            Set STDpob = CCCreateControl(ccsTextBox, "STDpob", "STDpob",      ↙
    ccsText, Empty, CCGetRequestParam("STDpob", Method))
            Set STDiniclass = CCCreateControl(ccsTextBox, "STDiniclass",      ↙
    "STDiniclass", ccsText, Empty, CCGetRequestParam("STDiniclass",           ↙
    Method))
            Set STDfathername = CCCreateControl(ccsTextBox, "STDfathername↙
    ", "STDfathername", ccsText, Empty, CCGetRequestParam("STDfathername",↙
     Method))
            Set STDfatheradd = CCCreateControl(ccsTextArea, "STDfatheradd"↙
    , "STDfatheradd", ccsText, Empty, CCGetRequestParam("STDfatheradd",       ↙
    Method))
            Set STDguardianname = CCCreateControl(ccsTextBox,                 ↙
    "STDguardianname", "STDguardianname", ccsText, Empty,                     ↙
    CCGetRequestParam("STDguardianname", Method))
```

```
            Set STDguardianadd = CCCreateControl(ccsTextArea,                ↙
    "STDguardianadd", "STDguardianadd", ccsText, Empty, CCGetRequestParam ↙
    ("STDguardianadd", Method))
            Set STDguardiantype = CCCreateControl(ccsTextBox,                ↙
    "STDguardiantype", "STDguardiantype", ccsText, Empty,                    ↙
    CCGetRequestParam("STDguardiantype", Method))
            Set STDstampdate = CCCreateControl(ccsLabel, "STDstampdate",     ↙
    "STDstampdate", ccsDate, Array("dd", "/", "mm", "/", "yyyy"),            ↙
    CCGetRequestParam("STDstampdate", Method))
            Set Button_Update = CCCreateButton("Button_Update")
            Set Button1 = CCCreateButton("Button1")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STDname, STDsex,            ↙
    STDreligion, STDrace, STDbirthcert, STDdob, STDpob, STDiniclass,         ↙
    STDfathername, STDfatheradd, STDguardianname, STDguardianadd,            ↙
    STDguardiantype)
        End If
    End Sub
'End Student Class_Initialize Event

'Student Initialize Method {22-0EE95DAE
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("sesUserID") = Session("UserID")
        End With
    End Sub
'End Student Initialize Method

'Student Class_Terminate Event {21-3EB64720
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Student Class_Terminate Event

'Student Validate Method {22-7AC02640
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Student Validate Method

'Student Operation Method {22-47563220
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "Button1")
            If Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            ElseIf Not IsEmpty(CCGetParam("Button1", Empty)) Then
                PressedButton = "Button1"
            End If
        End If
        Redirect = "STD_DataEntry.asp?" & CCGetQueryString("QueryString", ↙
    Array("ccsForm", "Button_Update", "Button1"))
        If Validate() Then
            If PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
                    Redirect = "STD_Password.asp?" & CCGetQueryString(    ↙
```

```
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button1" Then
                If NOT Button1.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "STD_Password.asp?" & CCGetQueryString(     ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Student Operation Method

'Student UpdateRow Method 622-B0467FD8
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.STDname.Value = STDname.Value
        DataSource.STDsex.Value = STDsex.Value
        DataSource.STDreligion.Value = STDreligion.Value
        DataSource.STDrace.Value = STDrace.Value
        DataSource.STDbirthcert.Value = STDbirthcert.Value
        DataSource.STDdob.Value = STDdob.Value
        DataSource.STDpob.Value = STDpob.Value
        DataSource.STDiniclass.Value = STDiniclass.Value
        DataSource.STDfathername.Value = STDfathername.Value
        DataSource.STDfatheradd.Value = STDfatheradd.Value
        DataSource.STDguardianname.Value = STDguardianname.Value
        DataSource.STDguardianadd.Value = STDguardianadd.Value
        DataSource.STDguardiantype.Value = STDguardiantype.Value
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End Student UpdateRow Method

'Student Show Method 622-E8E0F335
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.              ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Student" & IIf(EditMode, ↙
    ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
```

```
        Set Controls = CCCreateCollection(TemplateBlock, Null,                ↙
    ccsParseOverwrite, _
            Array(STDid, STDname, STDsex, STDreligion, STDrace,               ↙
    STDbirthcert, STDdob, DatePicker_STDdob, STDpob, STDiniclass,             ↙
    STDfathername, STDfatheradd, STDguardianname, STDguardianadd,             ↙
    STDguardiantype, STDstampdate, Button_Update, Button1))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record Student", DataSource.Connection. ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        STDname.Value = Recordset.Fields("STDname")
                        STDsex.Value = Recordset.Fields("STDsex")
                        STDreligion.Value = Recordset.Fields("STDreligion"↙
    )
                        STDrace.Value = Recordset.Fields("STDrace")
                        STDbirthcert.Value = Recordset.Fields(            ↙
    "STDbirthcert")
                        STDdob.Value = Recordset.Fields("STDdob")
                        STDpob.Value = Recordset.Fields("STDpob")
                        STDiniclass.Value = Recordset.Fields("STDiniclass"↙
    )
                        STDfathername.Value = Recordset.Fields(           ↙
    "STDfathername")
                        STDfatheradd.Value = Recordset.Fields(            ↙
    "STDfatheradd")
                        STDguardianname.Value = Recordset.Fields(         ↙
    "STDguardianname")
                        STDguardianadd.Value = Recordset.Fields(          ↙
    "STDguardianadd")
                        STDguardiantype.Value = Recordset.Fields(         ↙
    "STDguardiantype")
                    End If
                    STDid.Value = Recordset.Fields("STDid")
                    STDstampdate.Value = Recordset.Fields("STDstampdate")

                    if STDstampdate.Value <> "" then EditMode = False
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Update.Visible = EditMode AND UpdateAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Student_Show Method

End Class 'End Student Class 621-A61BA892

Class clsStudentDataSource 'StudentDataSource Class 621-EF9B50BE

'DataSource Variables 621-7B5C8347
    Public Errors, Connection, Parameters, CCSEvents
```

```
    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STDid
    Public STDname
    Public STDsex
    Public STDreligion
    Public STDrace
    Public STDbirthcert
    Public STDdob
    Public STDpob
    Public STDiniclass
    Public STDfathername
    Public STDfatheradd
    Public STDguardianname
    Public STDguardianadd
    Public STDguardiantype
    Public STDstampdate
'End DataSource Variables

'Constructor Class_Initialize Event [C1-46D5A7]
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,        ↙
    Recordset)
        Set STDname = CCCreateField("STDname", "STDname", ccsText, Empty, ↙
    Recordset)
        Set STDsex = CCCreateField("STDsex", "STDsex", ccsText, Empty,     ↙
    Recordset)
        Set STDreligion = CCCreateField("STDreligion", "STDreligion",      ↙
    ccsText, Empty, Recordset)
        Set STDrace = CCCreateField("STDrace", "STDrace", ccsText, Empty, ↙
    Recordset)
        Set STDbirthcert = CCCreateField("STDbirthcert", "STDbirthcert",   ↙
    ccsText, Empty, Recordset)
        Set STDdob = CCCreateField("STDdob", "STDdob", ccsDate, Array(     ↙
    "yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"),        ↙
    Recordset)
        Set STDpob = CCCreateField("STDpob", "STDpob", ccsText, Empty,     ↙
    Recordset)
        Set STDiniclass = CCCreateField("STDiniclass", "STDiniclass",      ↙
    ccsText, Empty, Recordset)
        Set STDfathername = CCCreateField("STDfathername", "STDfathername"↙
    , ccsText, Empty, Recordset)
        Set STDfatheradd = CCCreateField("STDfatheradd", "STDfatheradd",   ↙
    ccsText, Empty, Recordset)
        Set STDguardianname = CCCreateField("STDguardianname",             ↙
    "STDguardianname", ccsText, Empty, Recordset)
        Set STDguardianadd = CCCreateField("STDguardianadd",               ↙
    "STDguardianadd", ccsText, Empty, Recordset)
        Set STDguardiantype = CCCreateField("STDguardiantype",             ↙
```

```
    "STDguardiantype", ccsText, Empty, Recordset)
        Set STDstampdate = CCCreateField("STDstampdate", "STDstampdate",  ↙
    ccsDate, Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":"↙
    , "ss"), Recordset)
        Fields.AddFields Array(STDid, STDname, STDsex, STDreligion,       ↙
    STDrace, STDbirthcert, STDdob, STDpob, STDiniclass, STDfathername,    ↙
    STDfatheradd, STDguardianname, STDguardianadd, STDguardiantype,       ↙
    STDstampdate)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method {22-1D97F488
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "sesUserID", ccsText, Empty, Empty, Empty,   ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STDid", .         ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method {22-2F951BE3
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",   ↙
```

```
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {22F-41B4B0E3
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Update Method {22F-99CEFF49
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.      ↙
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE Student SET " & _
            "STDname=" & Connection.ToSQL(STDname.SQLText, STDname.       ↙
    DataType) & ", " & _
            "STDsex=" & Connection.ToSQL(STDsex.SQLText, STDsex.DataType) ↙
    & ", " & _
            "STDreligion=" & Connection.ToSQL(STDreligion.SQLText,        ↙
    STDreligion.DataType) & ", " & _
            "STDrace=" & Connection.ToSQL(STDrace.SQLText, STDrace.       ↙
    DataType) & ", " & _
            "STDbirthcert=" & Connection.ToSQL(STDbirthcert.SQLText,      ↙
    STDbirthcert.DataType) & ", " & _
            "STDdob=" & Connection.ToSQL(STDdob.SQLText, STDdob.DataType) ↙
    & ", " & _
            "STDpob=" & Connection.ToSQL(STDpob.SQLText, STDpob.DataType) ↙
    & ", " & _
            "STDiniclass=" & Connection.ToSQL(STDiniclass.SQLText,        ↙
    STDiniclass.DataType) & ", " & _
            "STDfathername=" & Connection.ToSQL(STDfathername.SQLText,    ↙
    STDfathername.DataType) & ", " & _
            "STDfatheradd=" & Connection.ToSQL(STDfatheradd.SQLText,      ↙
    STDfatheradd.DataType) & ", " & _
            "STDguardianname=" & Connection.ToSQL(STDguardianname.SQLText,↙
     STDguardianname.DataType) & ", " & _
            "STDguardianadd=" & Connection.ToSQL(STDguardianadd.SQLText,  ↙
    STDguardianadd.DataType) & ", " & _
            "STDguardiantype=" & Connection.ToSQL(STDguardiantype.SQLText,↙
     STDguardiantype.DataType) & ", " & _
            "STDstampdate= CURRENT_TIMESTAMP" & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",   ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method
```

```
End Class 'End StudentDataSource Class {EE-A61BAB0C

%>
```

```
<%
'BindEvents Method @1-[illegible]
Sub BindEvents()
    Set CCSEvents("AfterInitialize") = GetRef("Page_AfterInitialize")
End Sub
'End BindEvents Method

Function Page_AfterInitialize() 'Page_AfterInitialize @1-[illegible]

'Logout @43-[illegible]
    If NOT IsEmpty(CCGetParam("Logout", Empty)) Then
    CCLogoutUser
    Redirect = "STD_DataEntry.asp"
    End If
'End Logout

End Function 'Close Page_AfterInitialize @1-54C34B28


%>
```

```
<%
'Include Common Files @1-D6E3D1A1
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Server.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-96CC2CAB
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

Redirect = ""
TemplateFileName = "STD_End.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STD_End.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-74971E36

' Events
%>
<!-- #INCLUDE FILE="STD_End_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Go to destination page @1-63152A3E
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155E1DE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-F7A3B75E
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>    ↙
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page
```

```
'Unload Page @1-CB210092
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-29B0B101
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
End Sub
'End UnloadPage Sub


%>
```

```
    If NOT IsEmpty(CCGetParam("Logout", Empty)) Then
```

```
<%
'Include Common Files @1-36E3E132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Counter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-4B79C78D
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Student

Redirect = ""
TemplateFileName = "DataEntry.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "DataEntry.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-A0FAB1D0
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Student = new clsRecordStudent
Student.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-7B956912
Student.Operation
'End Execute Components

'Go to destination page @1-6A35B4F2
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-015DC1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page 61-45AC0EF5
Student.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page 61-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub 61-A4B96131
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Student = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordStudent 'Student Class 62-44D07613

'Student Variables 62-97545942

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public InsertAllowed
    Public UpdateAllowed
    Public DeleteAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim STDid
    Dim STDname
    Dim Button_Insert
    Dim Button_Update
    Dim Button_Delete
'End Student Variables

'Student Class_Initialize Event 62-E1217C47
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsStudentDataSource
```

```
        Set Command = New clsCommand
        InsertAllowed = True
        UpdateAllowed = True
        DeleteAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Student")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Student"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set STDid = CCCreateControl(ccsTextBox, "STDid", "STDid",      ↙
    ccsText, Empty, CCGetRequestParam("STDid", Method))
            STDid.Required = True
            Set STDname = CCCreateControl(ccsTextBox, "STDname", "STDname"↙
    , ccsText, Empty, CCGetRequestParam("STDname", Method))
            Set Button_Insert = CCCreateButton("Button_Insert")
            Set Button_Update = CCCreateButton("Button_Update")
            Set Button_Delete = CCCreateButton("Button_Delete")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STDid, STDname)
        End If
    End Sub
'End Student Class_Initialize Event

'Student Initialize Method 02-2179AA22
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlSTDid") = CCGetRequestParam("STDid", ccsGET)
        End With
    End Sub
'End Student Initialize Method

'Student Class_Terminate Event 01-31B64F29
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Student Class_Terminate Event

'Student Validate Method 02-78D52643
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Student Validate Method

'Student Operation Method 02-A46STD36
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "Button_Insert"↙
    )
            If Not IsEmpty(CCGetParam("Button_Insert", Empty)) Then
                PressedButton = "Button_Insert"
```

```
            ElseIf Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            ElseIf Not IsEmpty(CCGetParam("Button_Delete", Empty)) Then
                PressedButton = "Button_Delete"
            End If
        End If
        Redirect = "StudentList.asp?" & CCGetQueryString("QueryString",    ↙
    Array("ccsForm", "Button_Insert", "Button_Update", "Button_Delete"))
        If PressedButton = "Button_Delete" Then
            If NOT Button_Delete.OnClick OR NOT DeleteRow() Then
                Redirect = ""
            Else
                Redirect = "StudentList.asp?" & CCGetQueryString(          ↙
    "QueryString", Array("ccsForm"))
            End If
        ElseIf Validate() Then
            If PressedButton = "Button_Insert" Then
                If NOT Button_Insert.OnClick() OR NOT InsertRow() Then
                    Redirect = ""
                Else
                    Redirect = "StudentList.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
                    Redirect = "StudentList.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Student Operation Method

'Student InsertRow Method 02-79E07720
    Function InsertRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeInsert", Me)
        If NOT InsertAllowed Then InsertRow = False : Exit Function
        DataSource.STDid.Value = STDid.Value
        DataSource.STDname.Value = STDname.Value
        'DataSource.STDpwd.Value = STDid.Value ' AE
        DataSource.Insert(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInsert", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Insert Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        InsertRow = (Errors.Count = 0)
    End Function
'End Student InsertRow Method

'Student UpdateRow Method 02-12D01325
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.STDid.Value = STDid.Value
        DataSource.STDname.Value = STDname.Value
        'DataSource.STDpwd.Value = STDid.Value '<AE
        DataSource.Update(Command)
```

```
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End Student UpdateRow Method

'Student DeleteRow Method 82-1BC29772
    Function DeleteRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeDelete", Me)
        If NOT DeleteAllowed Then DeleteRow = False : Exit Function
        DataSource.Delete(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterDelete", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Delete Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        DeleteRow = (Errors.Count = 0)
    End Function
'End Student DeleteRow Method

'Student Show Method 61-8B98BA34
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Student" & IIf(EditMode, ↙
    ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(STDid, STDname, Button_Insert, Button_Update,              ↙
    Button_Delete))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record Student", DataSource.Connection. ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        STDid.Value = Recordset.Fields("STDid")
                        STDname.Value = Recordset.Fields("STDname")
                    End If
                Else
                    EditMode = False
                End If
            End If
```

```
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Insert.Visible = NOT EditMode AND InsertAllowed
        Button_Update.Visible = EditMode AND UpdateAllowed
        Button_Delete.Visible = EditMode AND DeleteAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Student Show Method

End Class 'End Student Class 02-A61BA892

Class clsStudentDataSource 'StudentDataSource Class 02-239B50BE

'DataSource Variables 02-E9290AAE
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Parameters fields
    Public STDid
    Public STDname
    Public STDpwd ' AE
'End DataSource Variables

'DataSource Class Initialize Event 02-18A23941
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,        ↙
    Recordset)
        Set STDname = CCCreateField("STDname", "STDname", ccsText, Empty, ↙
    Recordset)
        Fields.AddFields Array(STDid, STDname)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
```

```
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method C2-7E709E1C
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlSTDid", ccsText, Empty, Empty, Empty,     ↙
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "STDid", .          ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method C2-2F951BED
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event C2-41B4B0C2
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
```

```
    End Sub
'End DataSource Class_Terminate Event

'Delete Method {2-41250483
    Sub Delete(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildDelete", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Delete operation. There↙
     are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "DELETE FROM Student" & IIf(Len(Where) > 0, " WHERE " & ↙
    Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteDelete",   ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteDelete", Me)
    End Sub
'End Delete Method

'Update Method {2-5BA478A7
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.      ↙
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE Student SET " & _
            "STDid=" & Connection.ToSQL(STDid.SQLText, STDid.DataType) & "↙
    , " & _
            "STDname=" & Connection.ToSQL(STDname.SQLText, STDname.       ↙
    DataType) & ", " & _
            "STDpwd=" & Connection.ToSQL(STDid.SQLtext, STDid.DataType) & ↙
    _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",   ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

'Insert Method {2-2B8E0507
    Sub Insert(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildInsert", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        Cmd.SQL = "INSERT INTO Student(" & _
            "STDid, " & _
            "STDname, " & _
            "STDpwd" & _
        ") VALUES (" & _
            Connection.ToSQL(STDid.SQLText, STDid.DataType) & ", " & _
            Connection.ToSQL(STDname.SQLText, STDname.DataType) & ", " & _
            Connection.ToSQL(STDid.SQLText, STDid.DataType) & _
```

```
        ")"
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteInsert",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteInsert", Me)
    End Sub
'End Insert Method

End Class 'End StudentDataSource Class @1-A01BA1A1


%>
```

```
<%
'Include Common Files @1-36E3E132
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-FCCB1952
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim StudentSearch
Dim Student

Redirect = ""
TemplateFileName = "SearchList.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "SearchList.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-4543C67D
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set StudentSearch = new clsRecordStudentSearch
Set Student = new clsEditableGridStudent
Student.Initialize DBConnection1

' Events
%>
<!-- #INCLUDE FILE="SearchList_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-B195329E
StudentSearch.Operation
Student.ProcessOperations
'End Execute Components

'Go to destination page @1-6DA594F3
If NOT ( Redirect = "" ) Then
    UnloadPage
```

```
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template {1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page {1-1.TTDB26
StudentSearch.Show Tpl
Student.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page {1-7B110C6E
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub {1-BA533009
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set StudentSearch = Nothing
    Set Student = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordStudentSearch 'StudentSearch Class {3-D9B9D9D4

'StudentSearch Variables {3-7AEDE716

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim s_STDid
    Dim Button_DoSearch
'End StudentSearch Variables

'StudentSearch Class_Initialize Event {3-DB013C3F
    Private Sub Class_Initialize()
```

```
        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "StudentSearch")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "StudentSearch"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set s_STDid = CCCreateControl(ccsTextBox, "s_STDid", "s_STDid"↙
    , ccsText, Empty, CCGetRequestParam("s_STDid", Method))
            Set Button_DoSearch = CCCreateButton("Button_DoSearch")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(s_STDid)
        End If
    End Sub
'End StudentSearch_Class_Initialize Event

'StudentSearch_Class_Terminate Event @3-32B647C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End StudentSearch_Class_Terminate Event

'StudentSearch_Validate Method @3-7AC5A643
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End StudentSearch_Validate Method

'StudentSearch_Operation Method @3-56197E77
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoSearch"
            If Not IsEmpty(CCGetParam("Button_DoSearch", Empty)) Then
                PressedButton = "Button_DoSearch"
            End If
        End If
        Redirect = "SearchList.asp"
        If Validate() Then
            If PressedButton = "Button_DoSearch" Then
                If NOT Button_DoSearch.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "SearchList.asp?" & CCGetQueryString("Form"↙
    , Array("Button_DoSearch"))
                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End StudentSearch_Operation Method

'StudentSearch_Show Method @3-9FB1A8EF
```

```
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "StudentSearch" & IIf        ↙
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite, _
            Array(s_STDid, Button_DoSearch))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End StudentSearch Show Method

End Class 'End StudentSearch Class 03-A81BAF92

Class clsEditableGridStudent 'Student Class 08-0907ACF2

'Student Variables 02-0790CDFF

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public HTMLFormMethod
    Public PressedButton
    Public Errors
    Public IsFormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock
    Public PageSize
    Public PageNumber
    Public CachedColumns
    Public CachedColumnsNames
    Public CachedColumnsNumber
    Public SubmittedRows
    Public EmptyRows
    Public ErrorMessages

    Public CCSEvents
    Private CCSEventResult

    Public ActiveSorter, SortingDirection
    Public InsertAllowed
    Public UpdateAllowed
    Public DeleteAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls
```

```
    Private MaxCachedValues
    Private CachedValuesNumber
    Private NewEmptyRows

    ' Class variables
    Dim Student_TotalRecords
    Dim Sorter_STDid
    Dim Sorter_STDname
    Dim STDid
    Dim STDname
    Dim CheckBox_Delete
    Dim Navigator
    Dim Button_Submit
    Public Row
'End Student Variables

'Student Class_Initialize Event [C-27442846
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsStudentDataSource
        Set Command = New clsCommand
        InsertAllowed = True
        UpdateAllowed = True
        DeleteAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            ComponentName = "Student"

            ActiveSorter = CCGetParam("StudentOrder", Empty)
            SortingDirection = CCGetParam("StudentDir", Empty)
            If NOT(SortingDirection = "ASC" OR SortingDirection = "DESC") ↙
    Then _
                SortingDirection = Empty

            PageSize = CCGetParam(ComponentName & "PageSize", Empty)
            If NOT IsNumeric(PageSize) OR IsEmpty(PageSize) Then _
                PageSize = 5 _
            Else _
                PageSize = CInt(PageSize)
            PageNumber = CInt(CCGetParam(ComponentName & "Page", 1))

            If Request.QueryString("ccsForm") = ComponentName Then
                IsFormSubmitted = True
                EditMode = True
            Else
                IsFormSubmitted = False
                EditMode = False
            End If
            Method = IIf(IsFormSubmitted, ccsPost, ccsGet)
            Set Student_TotalRecords = CCCreateControl(ccsLabel,          ↙
    "Student_TotalRecords", "Student_TotalRecords", ccsText, Empty,       ↙
    CCGetRequestParam("Student_TotalRecords", Method))
            Set Sorter_STDid = CCCreateSorter("Sorter_STDid", Me,         ↙
    FileName)
            Set Sorter_STDname = CCCreateSorter("Sorter_STDname", Me,     ↙
    FileName)
            Set STDid = CCCreateControl(ccsTextBox, "STDid", "STDid",     ↙
    ccsText, Empty, CCGetRequestParam("STDid", Method))
            STDid.Required = True
            Set STDname = CCCreateControl(ccsTextBox, "STDname", "STDname"↙
    , ccsText, Empty, CCGetRequestParam("STDname", Method))
            Set CheckBox_Delete = CCCreateControl(ccsCheckBox,            ↙
    "CheckBox_Delete", "CheckBox_Delete", ccsBoolean, Array("True", "False↙
```

```
    ", Empty), CCGetRequestParam("CheckBox_Delete", Method))
            CheckBox_Delete.CheckedValue = true
            CheckBox_Delete.UncheckedValue = false
            Set Navigator = CCCreateNavigator(ComponentName, "Navigator", ↙
    FileName, 10, tpCentered)
            Set Button_Submit = CCCreateButton("Button_Submit")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(STDid, STDname)
        End If

        SubmittedRows = 0
        NewEmptyRows = 0
        EmptyRows = 3

        InitCachedColumns()

    End Sub
'End Student Class_Initialize Event

'Student InitCachedColumns Method 62-41961501
    Sub InitCachedColumns()
        Dim RetrievedNumber, i
        CachedColumnsNumber = 1
        ReDim CachedColumnsNames(CachedColumnsNumber)
        CachedColumnsNames(1) = "STDid"

        RetrievedNumber = 0
        CachedColumns = GetCachedColumns()

        If IsArray(CachedColumns) Then
            RetrievedNumber = UBound(CachedColumns)
            If RetrievedNumber > 0 Then
                MaxCachedValues = CInt(RetrievedNumber /                    ↙
    CachedColumnsNumber)
                If (RetrievedNumber Mod CachedColumnsNumber) > 0 Then
                    MaxCachedValues = MaxCachedValues + 1
                End If
                CachedValuesNumber = MaxCachedValues
            End If
        End If

        If RetrievedNumber = 0 Then
            MaxCachedValues = 50
            ReDim CachedColumns(MaxCachedValues * CachedColumnsNumber)
            CachedValuesNumber = 0
        End If

        If SubmittedRows > 0 Or NewEmptyRows > 0 Then
            EmptyRows = NewEmptyRows
        End If

        DataSource.CachedColumns = CachedColumns
        DataSource.CachedColumnsNumber = CachedColumnsNumber
        ReDim ErrorMessages(SubmittedRows + EmptyRows)
    End Sub
'End Student InitCachedColumns Method

'Student Initialize Method 62-ED1931EA
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub

        Set DataSource.Connection = objConnection
        With DataSource

            Set .Connection = objConnection
            .PageSize = PageSize
            .SetOrder ActiveSorter, SortingDirection
```

```
            .AbsolutePage = PageNumber
            .Parameters("urls_STDid") = CCGetRequestParam("s_STDid",        ↙
    ccsGET)
        End With
    End Sub
'End Student Initialize Method

'Student Class_Terminate Event B2-93B84739
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Student Class_Terminate Event

'Student Validate Method B2-41E1955B
    Function Validate()
        Dim Validation
        Dim i, InsertedRows, Method, IsDeleted, IsEmptyRow, IsNewRow
        Method = IIf(IsFormSubmitted, ccsPost, ccsGet)
        Validation = True

        If SubmittedRows > 0 Then
            For i = 1 To SubmittedRows
                IsDeleted = (Len(CCGetRequestParam("CheckBox_Delete_" &    ↙
    CStr(i), Method)) > 0)
                IsEmptyRow = (Len(CCGetRequestParam("STDid_" & CStr(i),    ↙
    Method)) = 0)
                IsEmptyRow = IsEmptyRow And (Len(CCGetRequestParam(        ↙
    "STDname_" & CStr(i), Method)) = 0)

                If (Not IsDeleted) And (Not IsEmptyRow Or (i <             ↙
    SubmittedRows - EmptyRows + 1)) Then
                    STDid.Errors.Clear
                    STDid.Text = CCGetRequestParam("STDid_" & CStr(i),     ↙
    Method)
                    STDname.Errors.Clear
                    STDname.Text = CCGetRequestParam("STDname_" & CStr(i),↙
     Method)
                    If Not ValidatingControls.Validate Then
                        ErrorMessages(i) = ValidatingControls.GetErrors
                        Validation = False
                    End If
                End If
            Next
        End If

        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Student Validate Method

'Student ProcessOperations Method B1-22412B04
    Sub ProcessOperations()
        Dim TmpWhere: TmpWhere = Datasource.Where

        If Not ( Visible And IsFormSubmitted ) Then Exit Sub

        If IsFormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Submit", "Button_Submit"↙
    )
            If Not IsEmpty(CCGetParam("Button_Submit", Empty)) Then
                PressedButton = "Button_Submit"
            End If
        End If
        Redirect = FileName & "?" & CCGetQueryString("QueryString",Array( ↙
    "ccsForm", "Button_Submit"))

        If Validate() Then
            If PressedButton = "Button_Submit" Then
```

```
                CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSubmit",  ↙
    Me)
                If NOT Button_Submit.OnClick() OR NOT InsertRows() OR NOT ↙
    UpdateRows() OR NOT DeleteRows() Then
                    Redirect = ""
                End If
                CCSEventResult = CCRaiseEvent(CCSEvents, "AfterSubmit",   ↙
    Me)
            End If
        Else
            Redirect = ""
        End If

        Datasource.Where = TmpWhere
    End Sub
'End Student ProcessOperations Method

'Student InsertRows Method 02-63872DDB
    Function InsertRows()
        If NOT InsertAllowed Then InsertRows = False : Exit Function

        Dim i, InsertedRows, Method, IsDeleted, IsEmptyRow

        Method = IIf(IsFormSubmitted, ccsPost, ccsGet)

        If SubmittedRows > 0 Then
            i = SubmittedRows - EmptyRows
            For i = (SubmittedRows - EmptyRows + 1) To SubmittedRows
                IsDeleted = (Len(CCGetRequestParam("CheckBox_Delete_" &   ↙
    CStr(i), Method)) > 0)
                IsEmptyRow = True
                IsEmptyRow = IsEmptyRow And (Len(CCGetRequestParam("STDid_↙
    " & CStr(i), Method)) = 0)
                IsEmptyRow = IsEmptyRow And (Len(CCGetRequestParam(       ↙
    "STDname_" & CStr(i), Method)) = 0)

                If (Not IsDeleted) And (Not IsEmptyRow) Then
                    STDid.Text = CCGetRequestParam("STDid_" & CStr(i),    ↙
    Method)
                    DataSource.STDid.Value = STDid.Value
                    STDname.Text = CCGetRequestParam("STDname_" & CStr(i),↙
     Method)
                    DataSource.STDname.Value = STDname.Value
                    DataSource.Insert(Command)


                    If DataSource.Errors.Count > 0 Then
                        PrintDBError "EditableGrid " & ComponentName & " /↙
     Insert Operation", DataSource.Connection.LastSQL, DataSource.Errors. ↙
    ToString()
                        DataSource.Errors.Clear
                        Errors.AddError "Database command error."
                    End If
                End If
            Next
        End If

        InsertRows = (Errors.Count = 0)
    End Function
'End Student InsertRows Method

'Student UpdateRows Method 02-397A71A7
    Function UpdateRows()
        If NOT UpdateAllowed Then UpdateRows = False : Exit Function

        Dim i, InsertedRows, Method, IsDeleted
        Method = IIf(IsFormSubmitted, ccsPost, ccsGet)
        If SubmittedRows > 0 Then
```

```
            For i = 1 To SubmittedRows - EmptyRows
                IsDeleted = (Len(CCGetRequestParam("CheckBox_Delete_" &    ↙
    CStr(i), Method)) > 0)

                If Not IsDeleted Then
                    STDid.Text = CCGetRequestParam("STDid_" & CStr(i),     ↙
    Method)
                    DataSource.STDid.Value = STDid.Value
                    STDname.Text = CCGetRequestParam("STDname_" & CStr(i),↙
     Method)
                    DataSource.STDname.Value = STDname.Value
                    DataSource.CurrentRow = i
                    DataSource.Update(Command)


                    If DataSource.Errors.Count > 0 Then
                        PrintDBError "EditableGrid " & ComponentName & " /↙
     Update Operation", DataSource.Connection.LastSQL, DataSource.Errors. ↙
    ToString()
                        DataSource.Errors.Clear
                        Errors.AddError "Database command error."
                    End If
                End If
            Next
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterSubmit", Me)
        UpdateRows = (Errors.Count = 0)
    End Function
'End Student UpdateRows Method

'Student DeleteRows Method 62-49AD7BA3
    Function DeleteRows()
        Dim i, Method

        Method = IIf(IsFormSubmitted, ccsPost, ccsGet)
        If NOT DeleteAllowed Then DeleteRows = False : Exit Function


        If SubmittedRows > 0 Then
            For i = 1 To SubmittedRows - EmptyRows
                If Len(CCGetRequestParam("CheckBox_Delete_" & CStr(i),     ↙
    Method)) > 0 Then
                    DataSource.CurrentRow = i
                    DataSource.Delete(Command)


                    If DataSource.Errors.Count > 0 Then
                        PrintDBError "EditableGrid " & ComponentName & " /↙
     Delete Operation", DataSource.Connection.LastSQL, DataSource.Errors. ↙
    ToString()
                        DataSource.Errors.Clear
                        Errors.AddError "Database command error."
                    End If
                End If
            Next
        End If

        DeleteRows = (Errors.Count = 0)
    End Function
'End Student DeleteRows Method

'GetFormScript Method 62-E2C89B42
    Function GetFormScript(TotalRows)
        Dim script,i: script = ""
        GetFormScript = script
    End Function
'End GetFormScript Method
```

```
'GetFormState Method 62-5833F920
    Function GetFormState
        Dim FormState, i, LastValueIndex, NewRows

        FormState = ""
        LastValueIndex = CachedValuesNumber * CachedColumnsNumber - 1

        If EditMode And LastValueIndex >= 0 Then

            For i = 1 To CachedColumnsNumber
                FormState = FormState & CachedColumnsNames(i)
                If i < CachedColumnsNumber Or LastValueIndex >= 0 Then
    FormState = FormState & ";"
            Next

            For i = 0 To LastValueIndex
                FormState = FormState & CachedColumns(i)
                If i < LastValueIndex Then FormState = FormState & ";"
            Next
        End If

        NewRows = IIf(InsertAllowed, EmptyRows, 0)
        GetFormState = CStr(SubmittedRows - NewRows) & ";" & CStr(NewRows)
        If Len(FormState) > 0 Then GetFormState = GetFormState & ";" &
    FormState

    End Function
'End GetFormState Method

'GetCachedColumns Method 62-A8DC0C1F
    Function GetCachedColumns
        Dim FormState, i, TotalValuesNumber, TempColumns, NewCachedColumns
    , TempValuesNumber
        Dim NewSubmittedRows : NewSubmittedRows = 0

        NewCachedColumns = Empty
        FormState = CCGetRequestParam("FormState", ccsPost)

        If Request.QueryString("ccsForm") = ComponentName Then
            If Not IsNull(FormState) Then
                If InStr(FormState,";") > 0 Then
                    TempColumns = Split(FormState,";")
                    If IsArray(TempColumns) Then
                        TempValuesNumber = UBound(TempColumns) - 1
                        If TempValuesNumber >= 0 Then
                            NewSubmittedRows = TempColumns(0)
                            NewEmptyRows     = TempColumns(1)
                        End If
                        SubmittedRows = CLng(NewSubmittedRows) + CLng
    (NewEmptyRows)

                        If TempValuesNumber > 1 And TempValuesNumber >=
    CachedColumnsNumber Then
                            ReDim NewCachedColumns(TempValuesNumber -
    CachedColumnsNumber + 1)
                            For i = 0 To TempValuesNumber -
    CachedColumnsNumber - 1
                                NewCachedColumns(i) = TempColumns(i +
    CachedColumnsNumber + 2)
                            Next
                        End If
                    End If
                Else
                    SubmittedRows = FormState
                End If
            End If
        End If
```

```
        GetCachedColumns = NewCachedColumns
    End Function
'End GetCachedColumns Method

'Student Show Method 62-ECC260F5
    Sub Show(Tpl)
        Dim StaticControls,RowControls

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If

        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Student")
        Set TemplateBlock = Tpl.Block("EditableGrid " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("Action") = HTMLFormAction
        TemplateBlock.Variable("HTMLFormProperties") = "action=""" &         ↙
    HTMLFormAction & """ method=""post"" " & "name=""" & ComponentName &    ↙
    """"
        TemplateBlock.Variable("HTMLFormEnctype") = "application/x-www-     ↙
    form-urlencoded"

        Set StaticControls = CCCreateCollection(TemplateBlock, Null,         ↙
    ccsParseOverwrite, _
            Array(Student_TotalRecords, Sorter_STDid, Sorter_STDname,        ↙
    Navigator, Button_Submit))
        Set RowControls = CCCreateCollection(TemplateBlock.Block("Row"),     ↙
    Null, ccsParseAccumulate, _
            Array(STDid, STDname, CheckBox_Delete))

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If NOT Visible Then Exit Sub

        If Not DeleteAllowed Then CheckBox_Delete.Visible = False

        Dim i, j
        i = 1 : j = 0
        If EditMode Then
            If Recordset.Errors.Count > 0 Then
                PrintDBError "EditableGrid Student", DataSource.Connection↙
    .LastSQL, Recordset.Errors.ToString
            ElseIf Not Recordset.EOF Then
                While Not Recordset.EoF AND (i-1) < PageSize


                    If Not IsFormSubmitted Then

                    End If

                    If Not IsFormSubmitted Then
                        STDid.Value = Recordset.Fields("STDid")
                    Else
                        STDid.Text = CCGetRequestParam("STDid_" & CStr(i),↙
     ccsPost)
                    End If
                    If Not IsFormSubmitted Then
                        STDname.Value = Recordset.Fields("STDname")
                    Else
                        STDname.Text = CCGetRequestParam("STDname_" & CStr↙
     (i), ccsPost)
```

```
                End If
                STDid.ExternalName = "STDid_" & CStr(i)
                STDname.ExternalName = "STDname_" & CStr(i)
                CheckBox_Delete.ExternalName = "CheckBox_Delete_" &    ↙
CStr(i)

                If j >= MaxCachedValues Then
                        MaxCachedValues = MaxCachedValues + 50
                        ReDim Preserve CachedColumns(MaxCachedValues* ↙
CachedColumnsNumber)
                End If
                CachedColumns(j * CachedColumnsNumber) = Recordset.    ↙
Fields("STDid")
                CachedValuesNumber = i

                If IsFormSubmitted Then
                    If Len(ErrorMessages(i)) > 0 Then
                        With TemplateBlock.Block("Row").Block(         ↙
"RowError")
                            .Variable("Error") = ErrorMessages(i)
                            .Parse False
                        End With
                    Else
                        TemplateBlock.Block("Row").Block("RowError"). ↙
Visible = False
                    End If
                End If

                CCSEventResult = CCRaiseEvent(CCSEvents,               ↙
"BeforeShowRow", Me)
                RowControls.Show
                i = i + 1: j = j + 1

                Recordset.MoveNext
            Wend
        End If

    ElseIf Not EditMode And Not InsertAllowed Then
        TemplateBlock.Block("NoRecords").Parse ccsParseOverwrite
    End If

    If Not InsertAllowed And Not UpdateAllowed And Not DeleteAllowed  ↙
Then
        Button_Submit.Visible = False
    End If

    CheckBox_Delete.Visible = False
    STDid.Value = ""
    STDname.Value = ""

    Dim NewRows
    NewRows = IIf(InsertAllowed, EmptyRows, 0)
    For i = i To i + NewRows - 1
        STDid.ExternalName = "STDid_" & CStr(i)
        STDname.ExternalName = "STDname_" & CStr(i)

        If IsFormSubmitted Then
            STDid.Text = CCGetRequestParam("STDid_" & CStr(i),         ↙
ccsPost)
            STDname.Text = CCGetRequestParam("STDname_" & CStr(i),     ↙
ccsPost)

            If Len(ErrorMessages(i)) > 0 Then
                With TemplateBlock.Block("Row").Block("RowError")
                    .Variable("Error") = ErrorMessages(i)
                    .Parse False
                End With
            Else
```

```
                    TemplateBlock.Block("Row").Block("RowError").Visible =↙
    False
                End If
            End If

            CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShowRow", Me)
            RowControls.Show
        Next

        SubmittedRows = i - 1
        TemplateBlock.Variable("FormScript") = GetFormScript(i - 1)
        TemplateBlock.Variable("FormState")  = GetFormState()

        If IsFormSubmitted Then
            If Errors.Count > 0 Or DataSource.Errors.Count > 0 Then
                With TemplateBlock.Block("Error")
                    .Variable("Error") = Errors.ToString _
                        & DataSource.Errors.ToString
                    .Parse False
                End With
            End If
        End If

        Navigator.SetDataSource Recordset
        StaticControls.Show

    End Sub
'End Student Show Method

End Class 'End Student Class 62-A61BA89E

Class clsStudentDataSource 'StudentDataSource Class 62-1D9B5DDE

'DataSource Variables 62-AC130CB3
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet
    Public CachedColumns
    Public CachedColumnsNumber
    Public CurrentRow

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STDid
    Public STDname
'End DataSource Variables

'DataSource Class_Initialize Event 62-0A3CE940
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,        ↙
    Recordset)
        Set STDname = CCCreateField("STDname", "STDname", ccsText, Empty, ↙
```

```
    Recordset)
        Fields.AddFields Array(STDid, STDname)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing
        Orders = Array( _
            Array("Sorter_STDid", "STDid", ""), _
            Array("Sorter_STDname", "STDname", ""))

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
        Order = "STDid"
    End Sub
'End DataSource Class_Initialize Event

'SetOrder Method 62-68FC9576
    Sub SetOrder(Column, Direction)
        Order = Recordset.GetOrder(Order, Column, Direction, Orders)
    End Sub
'End SetOrder Method

'BuildTableWhere Method 62-A6F3D015
    Public Sub BuildTableWhere()
        If CurrentRow > 0 Then
            Where = "STDid=" & Connection.ToSQL(CachedColumns((CurrentRow ↙
    - 1) * CachedColumnsNumber), ccsText)
        End If
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urls_STDid", ccsText, Empty, Empty, Empty,  ↙
    False
            .Criterion(1) = .Operation(opContains, False, "STDid", .      ↙
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method 62-51D015EC
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.CountSQL = CountSQL
        BuildTableWhere
```

```
        Cmd.Where = Where
        Cmd.OrderBy = Order
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {2-41B4B0C2
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Delete Method {2-41C50483
    Sub Delete(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildDelete", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Delete operation. There
     are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "DELETE FROM Student" & IIf(Len(Where) > 0, " WHERE " &
    Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteDelete",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteDelete", Me)
    End Sub
'End Delete Method

'Update Method {2-5BA478A3
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE Student SET " & _
            "STDid=" & Connection.ToSQL(STDid.SQLText, STDid.DataType) & "
    , " & _
            "STDname=" & Connection.ToSQL(STDname.SQLText, STDname.
    DataType) & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
```

```
'End Update Method

'Insert Method 62-CB8E0507
    Sub Insert(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildInsert", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        Cmd.SQL = "INSERT INTO Student(" & _
            "STDid, " & _
            "STDname" & _
        ") VALUES (" & _
            Connection.ToSQL(STDid.SQLText, STDid.DataType) & ", " & _
            Connection.ToSQL(STDname.SQLText, STDname.DataType) & _
        ")"
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteInsert",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteInsert", Me)
    End Sub
'End Insert Method

End Class 'End StudentDataSource Class 62-A61BA892

%>
```

```
<%
'LandMarks Marked (1-EEE2FTA
Sub BindEvents()
    Set Student.Student_TotalRecords.CCSEvents("BeforeShow") = GetRef(
    "Student_Student_TotalRecords_BeforeShow")
End Sub
'End BindEvents Model

Function Student_Student_TotalRecords_BeforeShow()
    'Student_Student_TotalRecords_BeforeShow 67-0C1DC6AE

'Retrieve number of records 68-0C95CE3D
    Student.Student_TotalRecords.Value = Student.DataSource.Recordset.
    RecordCount
'End Retrieve number of records

End Function 'Close Student_Student_TotalRecords_BeforeShow 67-54C34B2F


%>
```

```
<%
'Include Common Files @1-36E3E1A2
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sender.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-6047E553
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Student1

Redirect = ""
TemplateFileName = "SearchStudent.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "SearchStudent.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-50CF5018
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Student1 = New clsGridStudent1
Student1.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Go to destination page @1-6235A4FD
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155B1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template
```

```
'Show Page @1-C97E5011
Student1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C6C
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-FB1B42B7
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Student1 = Nothing
End Sub
'End UnloadPage Sub



Class clsGridStudent1 'Student1 Class @16-4BAA6B1D

'Student1 Variables @16-F3643518

    ' Public variables
    Public ComponentName, CCSEvents
    Public Visible, Errors
    Public DataSource, PageSize
    Public PageNumber
    Public Command
    Public TemplateBlock
    Public ActiveSorter, SortingDirection
    Public Recordset

    Private CCSEventResult

    ' Grid Controls
    Public StaticControls, RowControls
    Dim Sorter_STDid
    Dim Sorter_STDname
    Dim Sorter_STDpwd
    Dim STDid
    Dim STDname
    Dim STDpwd
    Dim Navigator
'End Student1 Variables

'Student1 Class_Initialize Event @16-5219C7A0
    Private Sub Class_Initialize()
        ComponentName = "Student1"
        Visible = True
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Errors = New clsErrors
        Set DataSource = New clsStudent1DataSource
        Set Command = New clsCommand
        PageSize = CCGetParam(ComponentName & "PageSize", Empty)
        If IsNumeric(PageSize) And Len(PageSize) > 0 Then
            If PageSize=0 Then Errors.AddError("(CCS06) Invalid page size.
    ")
            If PageSize > 10 Then PageSize = 10
        End If
        If NOT IsNumeric(PageSize) OR IsEmpty(PageSize) Then _
            PageSize = 10 _
```

```
        Else _
            PageSize = CInt(PageSize)
        PageNumber = CInt(CCGetParam(ComponentName & "Page", 1))
        ActiveSorter = CCGetParam("Student1Order", Empty)
        SortingDirection = CCGetParam("Student1Dir", Empty)
        If NOT(SortingDirection = "ASC" OR SortingDirection = "DESC") Then↙
 _
            SortingDirection = Empty

        Set STDid = CCCreateControl(ccsLink, "STDid", "STDid", ccsText,    ↙
    Empty, CCGetRequestParam("STDid", ccsGet))
        Set STDname = CCCreateControl(ccsLabel, "STDname", "STDname",      ↙
    ccsText, Empty, CCGetRequestParam("STDname", ccsGet))
        Set STDpwd = CCCreateControl(ccsLabel, "STDpwd", "STDpwd", ccsText↙
    , Empty, CCGetRequestParam("STDpwd", ccsGet))
        Set Sorter_STDid = CCCreateSorter("Sorter_STDid", Me, FileName)
        Set Sorter_STDname = CCCreateSorter("Sorter_STDname", Me,          ↙
    FileName)
        Set Sorter_STDpwd = CCCreateSorter("Sorter_STDpwd", Me, FileName)
        Set Navigator = CCCreateNavigator(ComponentName, "Navigator",      ↙
    FileName, 10, tpCentered)
    End Sub
'End Student1 Class_Initialize Event

'Student1 Initialize Method {16-2ADA3976}
    Sub Initialize(objConnection)
        If NOT Visible Then Exit Sub

        Set DataSource.Connection = objConnection
        DataSource.PageSize = PageSize
        DataSource.SetOrder ActiveSorter, SortingDirection
        DataSource.AbsolutePage = PageNumber
    End Sub
'End Student1 Initialize Method

'Student1 Class_Terminate Event {16-[illegible]}
    Private Sub Class_Terminate()
        Set DataSource = Nothing
        Set Command = Nothing
        Set Errors = Nothing
    End Sub
'End Student1 Class_Terminate Event

'Student1 Show Method {16-[illegible]}
    Sub Show(Tpl)
        If NOT Visible Then Exit Sub

        Dim RecordCounter, ShownRecords
        Dim RowBlock


        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)

        Set TemplateBlock = Tpl.Block("Grid " & ComponentName)
        Set RowBlock = TemplateBlock.Block("Row")
        Set StaticControls = CCCreateCollection(TemplateBlock, Null,       ↙
    ccsParseOverwrite, _
            Array(Sorter_STDid, Sorter_STDname, Sorter_STDpwd, Navigator))
        Set RowControls = CCCreateCollection(RowBlock, Null,               ↙
    ccsParseAccumulate, _
            Array(STDid, STDname, STDpwd))

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If NOT Visible Then Exit Sub

        Errors.AddErrors DataSource.Errors
        If Errors.Count > 0 Then
```

```
            TemplateBlock.HTML = CCFormatError("Grid Student1", Errors)
        Else

            ' Show NoRecords block if no records are found
            If Recordset.EOF Then
                TemplateBlock.Block("NoRecords").Parse ccsParseOverwrite
            End If
            While NOT Recordset.EOF AND ShownRecords < PageSize
                STDid.Value = Recordset.Fields("STDid")
                STDid.Parameters = CCGetQueryString("QueryString", Array( ↙
    "ccsForm"))
                STDid.Parameters = CCAddParam(STDid.Parameters, "STDid",  ↙
    Recordset.Fields("STDid_param1"))
                STDid.Page = "DataEntry.asp"
                STDname.Value = Recordset.Fields("STDname")
                STDpwd.Value = Recordset.Fields("STDpwd")
                CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShowRow", ↙
    Me)
                RowControls.Show
                Recordset.MoveNext
                ShownRecords = ShownRecords + 1
            Wend
            Navigator.SetDataSource Recordset
            StaticControls.Show
        End If

    End Sub
'End Student1 Show Method

End Class 'End Student1 Class @16-A61BA8C0

Class clsStudent1DataSource 'Student1DataSource Class @16-C75CDB13

'DataSource Variables @16-B40521B7
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public STDid
    Public STDid_param1
    Public STDname
    Public STDpwd
'End DataSource Variables

'DataSource Class_Initialize Event @16-74B45874
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set STDid = CCCreateField("STDid", "STDid", ccsText, Empty,       ↙
    Recordset)
        Set STDid_param1 = CCCreateField("STDid_param1", "STDid", ccsText,↙
     Empty, Recordset)
```

```
        Set STDname = CCCreateField("STDname", "STDname", ccsText, Empty, ↙
    Recordset)
        Set STDpwd = CCCreateField("STDpwd", "STDpwd", ccsText, Empty,     ↙
    Recordset)
        Fields.AddFields Array(STDid, STDid_param1, STDname, STDpwd)
        Orders = Array( _
            Array("Sorter_STDid", "STDid", ""), _
            Array("Sorter_STDname", "STDname", ""), _
            Array("Sorter_STDpwd", "STDpwd", ""))

        SQL = "SELECT *  " & _
        "FROM Student"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Student"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'SetOrder Method {16-68F09576
    Sub SetOrder(Column, Direction)
        Order = Recordset.GetOrder(Order, Column, Direction, Orders)
    End Sub
'End SetOrder Method

'BuildTableWhere Method {16-98E5A92F
    Public Sub BuildTableWhere()
    End Sub
'End BuildTableWhere Method

'Open Method {16-3D3C306E
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.CountSQL = CountSQL
        Cmd.Where = Where
        Cmd.OrderBy = Order
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",    ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {16-41B4B0E3
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End StudentDataSource Class {16-A61BA892

%>
```

```
<%
'Include Common Files @1-D6EDE1A1
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Server.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-8B821597
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Login

Redirect = ""
TemplateFileName = "STAFF_Login.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "STAFF_Login.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-A7F49BFE
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Login = new clsRecordLogin
Login.Initialize DBConnection1

' Events
%>
<!-- #INCLUDE FILE="STAFF_Login_events.asp" -->
<%
BindEvents

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-6AA0CD94
Login.Operation
'End Execute Components

'Go to destination page @1-67355472
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page
```

```
'Initialize HTML Template @1-0155E1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-C559B43C
Login.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210262
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-89FDB590
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Login = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordLogin 'Login Class @2-C3C99E74

'Login Variables @2-95290F3E

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim login
    Dim password
    Dim Button_DoLogin
'End Login Variables

'Login Class_Initialize Event @2-6C797537
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
```

```
        Set DataSource = New clsLoginDataSource
        Set Command = New clsCommand
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Login")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Login"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set login = CCCreateControl(ccsTextBox, "login", "login",
    ccsText, Empty, CCGetRequestParam("login", Method))
            login.Required = True
            Set password = CCCreateControl(ccsTextBox, "password",
    "password", ccsText, Empty, CCGetRequestParam("password", Method))
            password.Required = True
            Set Button_DoLogin = CCCreateButton("Button_DoLogin")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(login, password)
        End If
    End Sub
'End Login_Class_Initialize Event

'Login_Initialize Method @1-[illegible]
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
        End With
    End Sub
'End Login_Initialize Method

'Login_Class_Terminate Event @1-[illegible]
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Login_Class_Terminate Event

'Login_Validate Method @1-[illegible]
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End Login_Validate Method

'Login_Operation Method @1-[illegible]
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoLogin"
            If Not IsEmpty(CCGetParam("Button_DoLogin", Empty)) Then
                PressedButton = "Button_DoLogin"
            End If
        End If
        Redirect = FileName & ""
        If Validate() Then
            If PressedButton = "Button_DoLogin" Then
```

```
                If NOT Button_DoLogin.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = FileName & "?" & CCGetQueryString("Form",  ↙
    Array("Button_DoLogin"))
                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Login Operation Method

'Login Show Method C2-870BD699
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.            ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Login" & IIf(EditMode, ":↙
    Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,           ↙
    ccsParseOverwrite, _
            Array(login, password, Button_DoLogin))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
                    PrintDBError "Record Login", DataSource.Connection.  ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        login.Value = Recordset.Fields("login")
                        password.Value = Recordset.Fields("password")
                    End If
                Else
                    EditMode = False
                End If
            End If
        End If
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Login Show Method

End Class 'End Login Class C1-A61BA80C
```

```
Class clsLoginDataSource 'DataSource Class @1-5248A1E6

'DataSource Variables @1-34[illegible]
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public login
    Public password
'End DataSource Variables

'DataSource Class_Initialize Event @2-[illegible]
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set login = CCCreateField("login", "STAFFlogin", ccsText, Empty, ↙
    Recordset)
        Set password = CCCreateField("password", "STAFFpwd", ccsText, ↙
    Empty, Recordset)
        Fields.AddFields Array(login, password)

        SQL = "SELECT *  " & _
        "FROM lstStaffLogin" _
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM lstStaffLogin"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method @2-[illegible]
    Public Sub BuildTableWhere()
    End Sub
'End BuildTableWhere Method

'Open Method @2-B135[illegible]
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
            Set Open = New clsEmptyDataSource
            Exit Function
```

```
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {2-41B4B06E}
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End LoginDataSource Class {2-A51BA892}


%>
```

```
<%
'BindEvents Method @1-32618678
Sub BindEvents()
    Set Login.Button_DoLogin.CCSEvents("OnClick") = GetRef(
    "Login_Button_DoLogin_OnClick")
End Sub
'End BindEvents Method

Function Login_Button_DoLogin_OnClick() 'Login_Button_DoLogin_OnClick @3-
    ACB4691A

'Login @4-AA430BB1
    With Login
        If NOT CCLoginUser(.login.Value, .password.Value) Then
            .Errors.addError("Login or Password is incorrect.")
            Login_Button_DoLogin_OnClick = False
            .password.Value = ""
        Else
            If Not IsEmpty(CCGetParam("ret_link", Empty)) Then _
                Redirect = CCGetParam("ret_link", Empty)
            Login_Button_DoLogin_OnClick = True
        End If
    End With
'End Login

End Function 'Close Login_Button_DoLogin_OnClick @3-54C34B28


%>
```

```
<%
'Include Common Files @1-A0CD7E31
%>
<!-- #INCLUDE FILE="./Common.asp"-->
<!-- #INCLUDE FILE="./Cache.asp" -->
<!-- #INCLUDE FILE="./Connections.asp" -->
<!-- #INCLUDE FILE="./Common.asp" -->
<!-- #INCLUDE FILE="./Navigation.asp" -->
<%
'End Include Common Files

'Initialize Page @1-25000935
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim Class1

Redirect = ""
TemplateFileName = "CLASS_DataEntry.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "CLASS_DataEntry.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-A7567CC2
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set Class1 = new clsRecordClass1
Class1.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-6C275F04
Class1.Operation
'End Execute Components

'Go to destination page @1-61354451
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-015DD1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
```

```
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page @1-3247988E7
Class1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page @1-CB210C62
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub @1-4142E5C3
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set Class1 = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordClass1 'Class1 Class @2-A0274E60

'Class1 Variables @2-01622F12

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public InsertAllowed
    Public UpdateAllowed
    Public DeleteAllowed
    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim CLASSname
    Dim CLASSteacher
    Dim CLASSyear
    Dim DatePicker_CLASSyear
    Dim Link1
    Dim Button_Insert
    Dim Button_Update
    Dim Button_Delete
'End Class1 Variables

'Class1 Class_Initialize Event @2-1510C4C7
    Private Sub Class_Initialize()

        Visible = True
```

```
        Set Errors = New clsErrors
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set DataSource = New clsClass1DataSource
        Set Command = New clsCommand
        InsertAllowed = True
        UpdateAllowed = True
        DeleteAllowed = True
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "Class1")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "Class1"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set CLASSname = CCCreateControl(ccsTextBox, "CLASSname", "Name↙
    ", ccsText, Empty, CCGetRequestParam("CLASSname", Method))
            Set CLASSteacher = CCCreateControl(ccsTextBox, "CLASSteacher",↙
     "Teacher", ccsText, Empty, CCGetRequestParam("CLASSteacher", Method))
            Set CLASSyear = CCCreateControl(ccsTextBox, "CLASSyear", "Year↙
    ", ccsDate, Array("mm", "/", "yyyy"), CCGetRequestParam("CLASSyear",  ↙
    Method))
            Set DatePicker_CLASSyear = CCCreateDatePicker(                ↙
    "DatePicker_CLASSyear", "Class1", "CLASSyear")
            Set Link1 = CCCreateControl(ccsLink, "Link1", "Link1", ccsText↙
    , Empty, CCGetRequestParam("Link1", Method))
            Set Button_Insert = CCCreateButton("Button_Insert")
            Set Button_Update = CCCreateButton("Button_Update")
            Set Button_Delete = CCCreateButton("Button_Delete")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(CLASSname, CLASSteacher, ↙
    CLASSyear)
        End If
    End Sub
'End Class1 Class_Initialize Event

'Class1 Initialize Method @2-4F9B1051
    Sub Initialize(objConnection)

        If NOT Visible Then Exit Sub


        Set DataSource.Connection = objConnection
        With DataSource
            .Parameters("urlCLASSid") = CCGetRequestParam("CLASSid",      ↙
    ccsGET)
        End With
    End Sub
'End Class1 Initialize Method

'Class1 Class_Terminate Event @2-61B64129
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End Class1 Class_Terminate Event

'Class1 Validate Method @1-7BF51643
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
```

```
'End Class1 Validate Method

'Class1 Operation Method 02-333A193C
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = IIf(EditMode, "Button_Update", "Button_Insert"↙
    )
            If Not IsEmpty(CCGetParam("Button_Insert", Empty)) Then
                PressedButton = "Button_Insert"
            ElseIf Not IsEmpty(CCGetParam("Button_Update", Empty)) Then
                PressedButton = "Button_Update"
            ElseIf Not IsEmpty(CCGetParam("Button_Delete", Empty)) Then
                PressedButton = "Button_Delete"
            End If
        End If
        Redirect = "CLASS_DataEntry.asp?" & CCGetQueryString("QueryString"↙
    , Array("ccsForm", "Button_Insert", "Button_Update", "Button_Delete"))
        If PressedButton = "Button_Delete" Then
            If NOT Button_Delete.OnClick OR NOT DeleteRow() Then
                Redirect = ""
            Else
                Redirect = "CLASS_List.asp?" & CCGetQueryString(          ↙
    "QueryString", Array("ccsForm"))
            End If
        ElseIf Validate() Then
            If PressedButton = "Button_Insert" Then
                If NOT Button_Insert.OnClick() OR NOT InsertRow() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_List.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm"))

                End If
            ElseIf PressedButton = "Button_Update" Then
                If NOT Button_Update.OnClick() OR NOT UpdateRow() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_List.asp?" & CCGetQueryString(      ↙
    "QueryString", Array("ccsForm"))

                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End Class1 Operation Method

'Class1 InsertRow Method 02-B4E33A4E
    Function InsertRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeInsert", Me)
        If NOT InsertAllowed Then InsertRow = False : Exit Function
        DataSource.CLASSname.Value = CLASSname.Value
        DataSource.CLASSteacher.Value = CLASSteacher.Value
        DataSource.CLASSyear.Value = CLASSyear.Value
        DataSource.Insert(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInsert", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Insert Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        InsertRow = (Errors.Count = 0)
```

```
    End Function
'End Class1 InsertRow Method

'Class1 UpdateRow Method 01-07407479
    Function UpdateRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUpdate", Me)
        If NOT UpdateAllowed Then UpdateRow = False : Exit Function
        DataSource.CLASSname.Value = CLASSname.Value
        DataSource.CLASSteacher.Value = CLASSteacher.Value
        DataSource.CLASSyear.Value = CLASSyear.Value
        DataSource.Update(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterUpdate", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Update Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        UpdateRow = (Errors.Count = 0)
    End Function
'End Class1 UpdateRow Method

'Class1 DeleteRow Method 01-1BC26772
    Function DeleteRow()
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeDelete", Me)
        If NOT DeleteAllowed Then DeleteRow = False : Exit Function
        DataSource.Delete(Command)


        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterDelete", Me)
        If DataSource.Errors.Count > 0 Then
            PrintDBError "Record " & ComponentName & " / Delete Operation"↙
    , DataSource.Connection.LastSQL, DataSource.Errors.ToString()
            DataSource.Errors.Clear
            Errors.AddError "Database command error."
        End If
        DeleteRow = (Errors.Count = 0)
    End Function
'End Class1 DeleteRow Method

'Class1 Show Method 01-5BA7BC71
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)
        If Recordset.State = adStateOpen Then
            EditMode = NOT Recordset.EOF
        Else
            EditMode = False
        End If
        HTMLFormAction = FileName & "?" & CCAddParam(Request.          ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "Class1" & IIf(EditMode, "↙
    :Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,         ↙
    ccsParseOverwrite, _
            Array(CLASSname, CLASSteacher, CLASSyear, DatePicker_CLASSyear↙
    , Link1, Button_Insert, Button_Update, Button_Delete))
        If EditMode Then
            If Errors.Count = 0 Then
                If Recordset.Errors.Count > 0 Then
```

```
                    PrintDBError "Record Class1", DataSource.Connection. ↙
    LastSQL, Recordset.Errors.ToString
                ElseIf Not Recordset.EOF Then
                    If Not FormSubmitted Then
                        CLASSname.Value = Recordset.Fields("CLASSname")
                        CLASSteacher.Value = Recordset.Fields(          ↙
    "CLASSteacher")
                        CLASSyear.Value = Recordset.Fields("CLASSyear")
                    End If

                Else
                    EditMode = False
                End If
            End If
        End If
        Link1.Parameters = CCGetQueryString("QueryString", Array("ccsForm"↙
    ))
        Link1.Page = ""
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString _
                    & DataSource.Errors.ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction
        Button_Insert.Visible = NOT EditMode AND InsertAllowed
        Button_Update.Visible = EditMode AND UpdateAllowed
        Button_Delete.Visible = EditMode AND DeleteAllowed

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End Class1 Show Method

End Class 'End Class1 Class 02-A61BA8D2

Class clsClass1DataSource 'Class1DataSource Class 02-A4704BA6

'DataSource Variables 02-5B651A93
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource Fields
    Public CLASSname
    Public CLASSteacher
    Public CLASSyear
'End DataSource Variables

'DataSource Class_Initialize Event 01-6A73A5A3
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
```

```
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set CLASSname = CCCreateField("CLASSname", "CLASSname", ccsText,
    Empty, Recordset)
        Set CLASSteacher = CCCreateField("CLASSteacher", "CLASSteacher",
    ccsText, Empty, Recordset)
        Set CLASSyear = CCCreateField("CLASSyear", "CLASSyear", ccsDate,
    Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"),
    Recordset)
        Fields.AddFields Array(CLASSname, CLASSteacher, CLASSyear)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing

        SQL = "SELECT *  " & _
        "FROM Class"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Class"
        Where = ""
        Order = ""
    End Sub
'End DataSource Class_Initialize Event

'BuildTableWhere Method C1-60CB531B
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urlCLASSid", ccsFloat, Empty, Empty, Empty,
    False
            AllParamsSet = .AllParamsSet
            .Criterion(1) = .Operation(opEqual, False, "CLASSid", .
    getParamByID(1))
            .AssembledWhere = .Criterion(1)
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method C1-3F9518EC
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        If Not AllParamsSet Then
```

```
            Set Open = New clsEmptyDataSource
            Exit Function
        End If
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event 61-41048083
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

'Delete Method 62-45370177
    Sub Delete(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildDelete", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Delete operation. There
     are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "DELETE FROM Class" & IIf(Len(Where) > 0, " WHERE " &
    Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteDelete",
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteDelete", Me)
    End Sub
'End Delete Method

'DoMate Method 63-93120448C
    Sub Update(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildUpdate", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        BuildTableWhere
        If NOT AllParamsSet Then
            Errors.AddError "Unable to perform the Update operation.
    There are one or more unspecified parameters."
            Exit Sub
        End If
        Cmd.SQL = "UPDATE Class SET " & _
            "CLASSname=" & Connection.ToSQL(CLASSname.SQLText, CLASSname.
    DataType) & ", " & _
            "CLASSteacher=" & Connection.ToSQL(CLASSteacher.SQLText,
    CLASSteacher.DataType) & ", " & _
            "CLASSyear=" & Connection.ToSQL(CLASSyear.SQLText, CLASSyear.
    DataType) & _
            IIf(Len(Where) > 0, " WHERE " & Where, "")
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteUpdate",
    Me)
```

```
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteUpdate", Me)
    End Sub
'End Update Method

'Insert Method E2-E503CDB1
    Sub Insert(Cmd)
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildInsert", Me)
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdExec
        Cmd.CommandType = dsTable
        Cmd.CommandParameters = Empty
        Cmd.SQL = "INSERT INTO Class(" & _
            "CLASSname, " & _
            "CLASSteacher, " & _
            "CLASSyear" & _
        ") VALUES (" & _
            Connection.ToSQL(CLASSname.SQLText, CLASSname.DataType) & ", "↙
     & _
            Connection.ToSQL(CLASSteacher.SQLText, CLASSteacher.DataType) ↙
    & ", " & _
            Connection.ToSQL(CLASSyear.SQLText, CLASSyear.DataType) & _
        ")"
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteInsert",    ↙
    Me)
        Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteInsert", Me)
    End Sub
'End Insert Method

End Class 'End ClassClassSource Class E2-A01BA89C


%>
```

```
<%
'Include Common Files @1-D6EDD1A2
%>
<!-- #INCLUDE FILE=".\Common.asp"-->
<!-- #INCLUDE FILE=".\Cache.asp" -->
<!-- #INCLUDE FILE=".\Template.asp" -->
<!-- #INCLUDE FILE=".\Sorter.asp" -->
<!-- #INCLUDE FILE=".\Navigator.asp" -->
<%
'End Include Common Files

'Initialize Page @1-13653708
' Variables
Dim PathToRoot, ScriptPath
Dim FileName
Dim Redirect
Dim Tpl, HTMLTemplate
Dim TemplateFileName
Dim ComponentName
Dim PathToCurrentPage

' Events
Dim CCSEvents
Dim CCSEventResult

' Connections
Dim DBConnection1

' Page controls
Dim ClassSearch
Dim Class1

Redirect = ""
TemplateFileName = "CLASS_List.html"
Set CCSEvents = CreateObject("Scripting.Dictionary")
PathToCurrentPage = "./"
FileName = "CLASS_List.asp"
PathToRoot = "./"
ScriptPath = Left(Request.ServerVariables("PATH_TRANSLATED"), Len(Request.↙
    ServerVariables("PATH_TRANSLATED")) - Len(FileName))
'End Initialize Page

'Initialize Objects @1-B2F13544
Set DBConnection1 = New clsDBConnection1
DBConnection1.Open

' Controls
Set ClassSearch = new clsRecordClassSearch
Set Class1 = New clsGridClass1
Class1.Initialize DBConnection1

CCSEventResult = CCRaiseEvent(CCSEvents, "AfterInitialize", Nothing)
'End Initialize Objects

'Execute Components @1-5803EB3E
ClassSearch.Operation
'End Execute Components

'Go to destination page @1-6235A4A2
If NOT ( Redirect = "" ) Then
    UnloadPage
    Response.Redirect Redirect
End If
'End Go to destination page

'Initialize HTML Template @1-0155D1BE
CCSEventResult = CCRaiseEvent(CCSEvents, "OnInitializeView", Nothing)
Set HTMLTemplate = new clsTemplate
```

```
Set HTMLTemplate.Cache = TemplatesRepository
HTMLTemplate.LoadTemplate ScriptPath & TemplateFileName
Set Tpl = HTMLTemplate.Block("main")
CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Nothing)
'End Initialize HTML Template

'Show Page {1-7CFA1010
ClassSearch.Show Tpl
Class1.Show Tpl
HTMLTemplate.Parse "main", False
response.write Replace(HTMLTemplate.GetHTML("main"),"</body>","<center>
    <font face=""Arial""><small></small></font></center></body>")
'End Show Page

'Unload Page {1-CB210C52
UnloadPage
Set Tpl = Nothing
Set HTMLTemplate = Nothing
'End Unload Page

'UnloadPage Sub {1-22B38ACA
Sub UnloadPage()
    CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeUnload", Nothing)
    If DBConnection1.State = adStateOpen Then _
        DBConnection1.Close
    Set DBConnection1 = Nothing
    Set ClassSearch = Nothing
    Set Class1 = Nothing
End Sub
'End UnloadPage Sub

Class clsRecordClassSearch 'ClassSearch Class {13-A9B45120

'ClassSearch Variables {13-777C2324

    ' Public variables
    Public ComponentName
    Public HTMLFormAction
    Public PressedButton
    Public Errors
    Public FormSubmitted
    Public EditMode
    Public Visible
    Public Recordset
    Public TemplateBlock

    Public CCSEvents
    Private CCSEventResult

    Public DataSource
    Public Command
    Public ValidatingControls
    Public Controls

    ' Class variables
    Dim s_CLASSid
    Dim s_CLASSname
    Dim s_CLASSteacher
    Dim s_CLASSyear
    Dim DatePicker_s_CLASSyear
    Dim Button_DoSearch
'End ClassSearch Variables

'ClassSearch Class_Initialize Event {13-9AE5D402
    Private Sub Class_Initialize()

        Visible = True
        Set Errors = New clsErrors
```

```
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Dim Method
        Dim OperationMode

        If Visible Then
            OperationMode = Split(Request.QueryString("ccsForm"), ":")
            If UBound(OperationMode) > -1 Then
                FormSubmitted = (OperationMode(0) = "ClassSearch")
            End If
            If UBound(OperationMode) > 0 Then
                EditMode = (OperationMode(1) = "Edit")
            End If
            ComponentName = "ClassSearch"
            Method = IIf(FormSubmitted, ccsPost, ccsGet)
            Set s_CLASSid = CCCreateControl(ccsTextBox, "s_CLASSid",
    "s_CLASSid", ccsFloat, Empty, CCGetRequestParam("s_CLASSid", Method))
            Set s_CLASSname = CCCreateControl(ccsTextBox, "s_CLASSname",
    "s_CLASSname", ccsText, Empty, CCGetRequestParam("s_CLASSname",
    Method))
            Set s_CLASSteacher = CCCreateControl(ccsTextBox,
    "s_CLASSteacher", "s_CLASSteacher", ccsText, Empty, CCGetRequestParam
    ("s_CLASSteacher", Method))
            Set s_CLASSyear = CCCreateControl(ccsTextBox, "s_CLASSyear",
    "s_CLASSyear", ccsDate, Array("dd", "/", "mm", "/", "yyyy"),
    CCGetRequestParam("s_CLASSyear", Method))
            Set DatePicker_s_CLASSyear = CCCreateDatePicker(
    "DatePicker_s_CLASSyear", "ClassSearch", "s_CLASSyear")
            Set Button_DoSearch = CCCreateButton("Button_DoSearch")
            Set ValidatingControls = new clsControls
            ValidatingControls.addControls Array(s_CLASSid, s_CLASSname,
    s_CLASSteacher, s_CLASSyear)
        End If
    End Sub
'End ClassSearch Class_Initialize Event

'ClassSearch Class_Terminate Event {13-32B847C9
    Private Sub Class_Terminate()
        Set Errors = Nothing
    End Sub
'End ClassSearch Class_Terminate Event

'ClassSearch Validate Method {13-7AE52643
    Function Validate()
        Dim Validation
        Validation = ValidatingControls.Validate
        CCSEventResult = CCRaiseEvent(CCSEvents, "OnValidate", Me)
        Validate = Validation And (Errors.Count = 0)
    End Function
'End ClassSearch Validate Method

'ClassSearch Operation Method {13-CEDEC7E1
    Sub Operation()
        If NOT ( Visible AND FormSubmitted ) Then Exit Sub

        If FormSubmitted Then
            PressedButton = "Button_DoSearch"
            If Not IsEmpty(CCGetParam("Button_DoSearch", Empty)) Then
                PressedButton = "Button_DoSearch"
            End If
        End If
        Redirect = "CLASS_List.asp"
        If Validate() Then
            If PressedButton = "Button_DoSearch" Then
                If NOT Button_DoSearch.OnClick() Then
                    Redirect = ""
                Else
                    Redirect = "CLASS_List.asp?" & CCGetQueryString("Form"
    , Array("Button_DoSearch"))
```

```
                End If
            End If
        Else
            Redirect = ""
        End If
    End Sub
'End ClassSearch Operation Method

'ClassSearch Show Method [13-C0B0C4BB
    Sub Show(Tpl)

        If NOT Visible Then Exit Sub

        EditMode = False
        HTMLFormAction = FileName & "?" & CCAddParam(Request.                ↙
    ServerVariables("QUERY_STRING"), "ccsForm", "ClassSearch" & IIf          ↙
    (EditMode, ":Edit", ""))
        Set TemplateBlock = Tpl.Block("Record " & ComponentName)
        TemplateBlock.Variable("HTMLFormName") = ComponentName
        TemplateBlock.Variable("HTMLFormEnctype") ="application/x-www-form↙
    -urlencoded"
        Set Controls = CCCreateCollection(TemplateBlock, Null,               ↙
    ccsParseOverwrite,
            Array(s_CLASSid, s_CLASSname, s_CLASSteacher, s_CLASSyear,       ↙
    DatePicker_s_CLASSyear, Button_DoSearch))
        If Not FormSubmitted Then
        End If
        If FormSubmitted Then
            With TemplateBlock.Block("Error")
                .Variable("Error") = ValidatingControls.GetErrors & Errors↙
    .ToString
                .Parse False
            End With
        End If
        TemplateBlock.Variable("Action") = HTMLFormAction

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If Visible Then Controls.Show
    End Sub
'End ClassSearch Show Method

End Class 'End ClassSearch Class [13-BA1BA8]

Class clsGridClass1 'Class1 Class [1A-7F7B0BA4

'Class1 Variables [A1-C0115BB9

    ' Public Variables
    Public ComponentName, CCSEvents
    Public Visible, Errors
    Public DataSource, PageSize
    Public PageNumber
    Public Command
    Public TemplateBlock
    Public ActiveSorter, SortingDirection
    Public Recordset

    Private CCSEventResult

    ' Grid Controls
    Public StaticControls, RowControls
    Dim Sorter_CLASSid
    Dim Sorter_CLASSname
    Dim Sorter_CLASSteacher
    Dim Sorter_CLASSyear
    Dim CLASSid
    Dim CLASSname
    Dim CLASSteacher
```

```
    Dim CLASSyear
    Dim Navigator
'End Class1 Variables

'Class1 Class_Initialize Event @12-4E785FAB
    Private Sub Class_Initialize()
        ComponentName = "Class1"
        Visible = True
        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Errors = New clsErrors
        Set DataSource = New clsClass1DataSource
        Set Command = New clsCommand
        PageSize = CCGetParam(ComponentName & "PageSize", Empty)
        If IsNumeric(PageSize) And Len(PageSize) > 0 Then
            If PageSize=0 Then Errors.AddError("(CCS06) Invalid page size.↙
    ")
            If PageSize > 20 Then PageSize = 20
        End If
        If NOT IsNumeric(PageSize) OR IsEmpty(PageSize) Then _
            PageSize = 20 _
        Else _
            PageSize = CInt(PageSize)
        PageNumber = CInt(CCGetParam(ComponentName & "Page", 1))
        ActiveSorter = CCGetParam("Class1Order", Empty)
        SortingDirection = CCGetParam("Class1Dir", Empty)
        If NOT(SortingDirection = "ASC" OR SortingDirection = "DESC") Then↙
     _
            SortingDirection = Empty

        Set CLASSid = CCCreateControl(ccsLink, "CLASSid", "CLASSid",      ↙
    ccsFloat, Empty, CCGetRequestParam("CLASSid", ccsGet))
        Set CLASSname = CCCreateControl(ccsLabel, "CLASSname", "CLASSname"↙
    , ccsText, Empty, CCGetRequestParam("CLASSname", ccsGet))
        Set CLASSteacher = CCCreateControl(ccsLabel, "CLASSteacher",      ↙
    "CLASSteacher", ccsText, Empty, CCGetRequestParam("CLASSteacher",     ↙
    ccsGet))
        Set CLASSyear = CCCreateControl(ccsLabel, "CLASSyear", "CLASSyear"↙
    , ccsDate, Array("dd", "/", "mm", "/", "yyyy"), CCGetRequestParam(    ↙
    "CLASSyear", ccsGet))
        Set Sorter_CLASSid = CCCreateSorter("Sorter_CLASSid", Me,         ↙
    FileName)
        Set Sorter_CLASSname = CCCreateSorter("Sorter_CLASSname", Me,     ↙
    FileName)
        Set Sorter_CLASSteacher = CCCreateSorter("Sorter_CLASSteacher", Me↙
    , FileName)
        Set Sorter_CLASSyear = CCCreateSorter("Sorter_CLASSyear", Me,     ↙
    FileName)
        Set Navigator = CCCreateNavigator(ComponentName, "Navigator",     ↙
    FileName, 10, tpCentered)
    End Sub
'End Class1 Class_Initialize Event

'Class1 Initialize Method @12-CAEA3075
    Sub Initialize(objConnection)
        If NOT Visible Then Exit Sub

        Set DataSource.Connection = objConnection
        DataSource.PageSize = PageSize
        DataSource.SetOrder ActiveSorter, SortingDirection
        DataSource.AbsolutePage = PageNumber
    End Sub
'End Class1 Initialize Method

'Class1 Class_Terminate Event @12-E1928F9B
    Private Sub Class_Terminate()
        Set DataSource = Nothing
        Set Command = Nothing
        Set Errors = Nothing
```

```
    End Sub
'End Class1 Class_Terminate Event

'Class1 Show Method @12-2218B9B3
    Sub Show(Tpl)
        If NOT Visible Then Exit Sub

        Dim RecordCounter, ShownRecords
        Dim RowBlock

        With DataSource
            .Parameters("urls_CLASSid") = CCGetRequestParam("s_CLASSid",  ↙
    ccsGET)
            .Parameters("urls_CLASSname") = CCGetRequestParam("s_CLASSname↙
    ", ccsGET)
            .Parameters("urls_CLASSteacher") = CCGetRequestParam(         ↙
    "s_CLASSteacher", ccsGET)
            .Parameters("urls_CLASSyear") = CCGetRequestParam("s_CLASSyear↙
    ", ccsGET)
        End With

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeSelect", Me)
        Set Recordset = DataSource.Open(Command)

        Set TemplateBlock = Tpl.Block("Grid " & ComponentName)
        Set RowBlock = TemplateBlock.Block("Row")
        Set StaticControls = CCCreateCollection(TemplateBlock, Null,      ↙
    ccsParseOverwrite, _
            Array(Sorter_CLASSid, Sorter_CLASSname, Sorter_CLASSteacher,  ↙
    Sorter_CLASSyear, Navigator))
        Set RowControls = CCCreateCollection(RowBlock, Null,              ↙
    ccsParseAccumulate, _
            Array(CLASSid, CLASSname, CLASSteacher, CLASSyear))

        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShow", Me)
        If NOT Visible Then Exit Sub

        Errors.AddErrors DataSource.Errors
        If Errors.Count > 0 Then
            TemplateBlock.HTML = CCFormatError("Grid Class1", Errors)
        Else

            ' Show NoRecords block if no records are found
            If Recordset.EOF Then
                TemplateBlock.Block("NoRecords").Parse ccsParseOverwrite
            End If
            While NOT Recordset.EOF AND ShownRecords < PageSize
                CLASSid.Value = Recordset.Fields("CLASSid")
                CLASSid.Parameters = CCGetQueryString("QueryString", Array↙
    ("ccsForm"))
                CLASSid.Parameters = CCAddParam(CLASSid.Parameters,       ↙
    "CLASSid", Recordset.Fields("CLASSid_param1"))
                CLASSid.Page = "CLASS_DataEntry.asp"
                CLASSname.Value = Recordset.Fields("CLASSname")
                CLASSteacher.Value = Recordset.Fields("CLASSteacher")
                CLASSyear.Value = Recordset.Fields("CLASSyear")
                CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeShowRow", ↙
    Me)
                RowControls.Show
                Recordset.MoveNext
                ShownRecords = ShownRecords + 1
            Wend
            Navigator.SetDataSource Recordset
            StaticControls.Show
        End If

    End Sub
'End Class1 Show Method
```

```
End Class 'End Class1 Class 611-A610A800

Class clsClass1DataSource 'Class1DataSource Class 611-E4094DAA

'DataSource Variables 612-2A7A0201
    Public Errors, Connection, Parameters, CCSEvents

    Public Recordset
    Public SQL, CountSQL, Order, Where, Orders
    Public RecordsCount
    Public PageSize
    Public PageCount
    Public AbsolutePage
    Public Fields
    Dim WhereParameters
    Public AllParamsSet

    Private CurrentOperation
    Private CCSEventResult

    ' Datasource fields
    Public CLASSid
    Public CLASSid_param1
    Public CLASSname
    Public CLASSteacher
    Public CLASSyear
'End DataSource Variables

'DataSource Class_Initialize Event 612-8B710C66
    Private Sub Class_Initialize()

        Set CCSEvents = CreateObject("Scripting.Dictionary")
        Set Fields = New clsFields
        Set Recordset = New clsDataSource
        Set Errors = New clsErrors
        AllParamsSet = True
        Set CLASSid = CCCreateField("CLASSid", "CLASSid", ccsFloat, Empty,↙
     Recordset)
        Set CLASSid_param1 = CCCreateField("CLASSid_param1", "CLASSid",   ↙
    ccsText, Empty, Recordset)
        Set CLASSname = CCCreateField("CLASSname", "CLASSname", ccsText,  ↙
    Empty, Recordset)
        Set CLASSteacher = CCCreateField("CLASSteacher", "CLASSteacher",  ↙
    ccsText, Empty, Recordset)
        Set CLASSyear = CCCreateField("CLASSyear", "CLASSyear", ccsDate,  ↙
    Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":", "nn", ":", "ss"), ↙
    Recordset)
        Fields.AddFields Array(CLASSid, CLASSid_param1, CLASSname,        ↙
    CLASSteacher, CLASSyear)
        Set Parameters = Server.CreateObject("Scripting.Dictionary")
        Set WhereParameters = Nothing
        Orders = Array( _
            Array("Sorter_CLASSid", "CLASSid", ""), _
            Array("Sorter_CLASSname", "CLASSname", ""), _
            Array("Sorter_CLASSteacher", "CLASSteacher", ""), _
            Array("Sorter_CLASSyear", "CLASSyear", ""))

        SQL = "SELECT *  " & _
        "FROM Class"
        CountSQL = "SELECT COUNT(*)  " & _
        "FROM Class"
        Where = ""
        Order = "CLASSid"
    End Sub
'End DataSource Class_Initialize Event

'GetOrder Method 612-6E9F9E36
```

```
    Sub SetOrder(Column, Direction)
        Order = Recordset.GetOrder(Order, Column, Direction, Orders)
    End Sub
'End SetOrder Method

'BuildTableWhere Method @12-00B45377
    Public Sub BuildTableWhere()
        Dim WhereParams

        If Not WhereParameters Is Nothing Then _
            Exit Sub
        Set WhereParameters = new clsSQLParameters
        With WhereParameters
            Set .Connection = Connection
            Set .ParameterSources = Parameters
            Set .DataSource = Me
            .AddParameter 1, "urls_CLASSid", ccsFloat, Empty, Empty, Empty↙
    , False
            .AddParameter 2, "urls_CLASSname", ccsText, Empty, Empty,     ↙
    Empty, False
            .AddParameter 3, "urls_CLASSteacher", ccsText, Empty, Empty,  ↙
    Empty, False
            .AddParameter 4, "urls_CLASSyear", ccsDate, Array("dd", "/",  ↙
    "mm", "/", "yyyy"), Array("yyyy", "-", "mm", "-", "dd", " ", "HH", ":"↙
    , "nn", ":", "ss"), Empty, False
            .Criterion(1) = .Operation(opEqual, False, "CLASSid", .       ↙
    getParamByID(1))
            .Criterion(2) = .Operation(opContains, False, "CLASSname", .  ↙
    getParamByID(2))
            .Criterion(3) = .Operation(opContains, False, "CLASSteacher", ↙
    .getParamByID(3))
            .Criterion(4) = .Operation(opEqual, False, "CLASSyear", .     ↙
    getParamByID(4))
            .AssembledWhere = .opAND(False, .opAND(False, .opAND(False, . ↙
    Criterion(1), .Criterion(2)), .Criterion(3)), .Criterion(4))
            WhereParams = .AssembledWhere
            If Len(Where) > 0 Then
                If Len(WhereParams) > 0 Then _
                    Where = Where & " AND " & WhereParams
            Else
                If Len(WhereParams) > 0 Then _
                    Where = WhereParams
            End If
        End With
    End Sub
'End BuildTableWhere Method

'Open Method @12-F13016B9
    Function Open(Cmd)
        Errors.Clear
        Set Recordset.DataSource = Me
        Set Cmd.Connection = Connection
        Cmd.CommandOperation = cmdOpen
        Cmd.PageSize = PageSize
        Cmd.ActivePage = AbsolutePage
        Cmd.CommandType = dsTable
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeBuildSelect", Me)
        Cmd.SQL = SQL
        Cmd.CountSQL = CountSQL
        BuildTableWhere
        Cmd.Where = Where
        Cmd.OrderBy = Order
        CCSEventResult = CCRaiseEvent(CCSEvents, "BeforeExecuteSelect",   ↙
    Me)
        If Errors.Count = 0 And CCSEventResult Then _
            Set Recordset = Cmd.Exec(Errors)
        CCSEventResult = CCRaiseEvent(CCSEvents, "AfterExecuteSelect", Me)
        Set Recordset.FieldsCollection = Fields
```

```
        Set Open = Recordset
    End Function
'End Open Method

'DataSource Class_Terminate Event {1E-41B4BC8D
    Private Sub Class_Terminate()
        If Recordset.State = adStateOpen Then _
            Recordset.Close
        Set Recordset = Nothing
        Set Parameters = Nothing
        Set Errors = Nothing
    End Sub
'End DataSource Class_Terminate Event

End Class 'End Class:DataSource Class {1E-A61BA692


%>
```

Printing Module

frmLogin.frm

```
Private Sub Form_Load()
lblVersion.Caption = "Versi " & App.Major & "." & App.Minor & "." & App.Revision
End Sub

Private Sub cmdBatal_Click()
  Unload Me
End Sub

Private Sub cmdOK_Click()
Dim rs As ADODB.Recordset

If txtLogin = "" Then
  MsgBox "Please Enter your username.", , "System User Verification"
  txtLogin.SetFocus
  Exit Sub
End If

Set rs = myDB("SELECT * FROM STAFF where STAFFlogin = '" & txtLogin.Text & "' AND STAFFprint = 1", 1)

If Not rs.EOF Then
  If rs!STAFFpwd = txtPassword.Text Then
    UserName = rs!STAFFlogin
    UserID = rs!STAFFid
    frmMain.Show
    Unload Me
  Else
    MsgBox "The password that you entered is not correct!" & vbLf & _
        "Please try again.", , "System User Verification"
    txtLogin.SetFocus
  End If

Else

  MsgBox "Your User login was not recognized!" & vbLf & _
      "Please try again.", , "System User Verification"
  txtPassword.Text = ""
  txtLogin.SetFocus
  txtLogin.SelStart = 0
  txtLogin.SelLength = Len(txtSISUserName

End If

End Sub


Private Sub txtPassword_KeyPress(KeyAscii As Integer)
If KeyAscii = vbEnter Then
  Call cmdOK_Click
ElseIf KeyAscii = 27 Then
  End
End If
End Sub

Private Sub txtLogin_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
  Call cmdOK_Click
ElseIf KeyAscii = 27 Then
  End
End If
End Sub
```

frmMain.frm

```
Private Sub chkStdInfo_Click()

If chkStdInfo.Value Then
   DTs_year.Enabled = True
Else
   DTs_year.Enabled = False
End If

End Sub

Private Sub chkYear_Click()

If chkYear.Value Then
   txtClassYear.Enabled = True
Else
   txtClassYear.Enabled = False
End If

End Sub

Private Sub cmdClear_Click()

FramePrint.Visible = False

txtClassID.Text = ""
txtClassName.Text = ""
txtClassTeacher.Text = ""
txtClassYear.Value = Date
txtClassID.SetFocus
chkYear.Value = 0

lstClass.ListItems.Clear

End Sub

Private Sub cmdPCancel_Click()
FramePrint.Visible = False
End Sub

Private Sub cmdPOk_Click()
Dim rs As ADODB.Recordset
Dim strSql As String

strSql = "SELECT * FROM lstSTDinfo WHERE YEAR(STDstampdate) = " & Format(DTPicker1.Value, "yyyy")

Set rs = myDB(strSql, 1)

lstAttendance.DataControl1.Recordset = rs
lstAttendance.lblReportTitle.Caption = "Student Information for " & Format(DTPicker1.Value, "yyyy") & " Intake"

lstAttendance.PageSettings.BottomMargin = 1440

lstAttendance.Show

FramePrint.Visible = False

End Sub

Private Sub cmdPrint_Click()
Dim rs As ADODB.Recordset
Dim strSql As String

If lblClassID.Caption = "" Then Exit Sub

strSql = "SELECT * FROM lstSTDinfo WHERE CLASSid = " & lblClassID.Caption

Set rs = myDB(strSql, 1)
```

```
lstAttendance.DataControl1.Recordset = rs
lstAttendance.lblReportTitle.Caption = "Student Information for Class " & lblClassName.Caption & " (" & lblClassID.Caption & ")"

lstAttendance.PageSettings.BottomMargin = 1440

lstAttendance.Show

End Sub

Private Sub cmdPrintYr_Click()

If FramePrint.Visible Then
   FramePrint.Visible = False
Else
   FramePrint.ZOrder
   FramePrint.Visible = True
End If

End Sub

Private Sub cmdSearch_Click()
Dim myLst As ListItem
Dim rs As ADODB.Recordset
Dim sqlStr As String

On Error GoTo errHandler

FramePrint.Visible = False

sqlStr = "SELECT * FROM lstClass WHERE CLASSname LIKE '%" & txtClassName.Text & "%' "

If chkYear.Value Then sqlStr = sqlStr & "AND YEAR(CLASSyear) = " & CInt(Format(txtClassYear.Value, "yyyy")) & " "
If txtClassID.Text <> "" Then sqlStr = sqlStr & "AND CLASSid = " & CInt(txtClassID.Text) & " "
'If txtClassName.Text <> "" Then sqlStr = sqlStr & "AND CLASSname = '" & txtClassName.Text & "' "
If txtClassTeacher.Text <> "" Then sqlStr = sqlStr & "AND CLASSteacher LIKE '%" & txtClassTeacher.Text & "%' "

Set rs = myDB(sqlStr, 1)

lstClass.ListItems.Clear

If rs.EOF Then
   Set myLst = lstClass.ListItems.Add(, , "-")
      myLst.SubItems(1) = "No Record Found"
Else

   While Not rs.EOF
      Set myLst = lstClass.ListItems.Add(, , rs!CLASSid)
         myLst.SubItems(1) = rs!ClassName
         myLst.SubItems(2) = rs!CLASSteacher
         myLst.SubItems(3) = Format(rs!CLASSyear, "yyyy")
         myLst.SubItems(4) = rs!StdCount
      rs.MoveNext
   Wend

End If

Exit Sub

errHandler:
MsgBox "An Error has occured while executing the search module! " & vbLf & _
     "Error Number: " & Err.Number & vbLf & _
     "Error Desc: " & Err.Description & " ", vbOKOnly + vbExclamation, "Error Message"

End Sub

Private Sub cmdStdCancel_Click()

picStdSearch.Visible = False

End Sub
```

```
Private Sub cmdStdClear_Click()

lstStdInfo.ListItems.Clear
txtS_StdID.Text = ""
txtS_StdName.Text = ""
txtS_Admission.Text = ""
DTs_year.Value = Date
chkStdInfo.Value = 0

txtS_StdID.SetFocus

End Sub

Private Sub cmdStdPrint_Click()
Dim rs As ADODB.Recordset
Dim strSql As String

If lblSTDid.Caption = "" Then Exit Sub

strSql = "SELECT * FROM lstSTDinfo WHERE STDid = '" & lblSTDid.Caption & "'"

Set rs = myDB(strSql, 1)

lstAttendance.DataControl1.Recordset = rs
lstAttendance.lblReportTitle.Caption = "Report for Student Number " & lblSTDadmissionNum.Caption

lstAttendance.Show

End Sub

Private Sub cmdStdSearch_Click()
Dim rs As ADODB.Recordset
Dim strSql As String
Dim myLst As ListItem
Dim tmpSTDnum As String

On Error GoTo errHandler

strSql = "SELECT * FROM Student WHERE STDid LIKE '%" & txtS_StdID.Text & "%' "

If txtS_StdName.Text <> "" Then strSql = strSql & "AND STDname Like '%" & txtS_StdName.Text & "%' "
If txtS_Admission.Text <> "" Then strSql = strSql & "AND STDadmissionNum = " & Val(txtS_Admission.Text) & " "
If chkStdInfo.Value Then strSql = strSql & "AND Year(STDstampdate) = " & Format(DTs_year.Value, "yyyy")

Set rs = myDB(strSql, 1)

lstStdInfo.ListItems.Clear

If rs.EOF Then
   lblResult.Caption = "No Record Found"
Else
   lblResult.Caption = rs.RecordCount & " Record(s) Found"
End If

While Not rs.EOF
   tmpSTDnum = Format(rs!STDadmissionnum, "000000") & "/" & Format(rs!STDstampdate, "yy")
   Set myLst = lstStdInfo.ListItems.Add(, , tmpSTDnum)
      myLst.SubItems(1) = rs!STDid
      myLst.SubItems(2) = rs!STDname
      myLst.SubItems(3) = Format(rs!STDstampdate, "yyyy")
   rs.MoveNext
Wend

Exit Sub
errHandler:

MsgBox "An Error has occured while executing the search module! " & vbLf & _
     "Error Number: " & Err.Number & vbLf & _
     "Error Desc: " & Err.Description & " ", vbOKOnly + vbExclamation, "Error Message"
```

```
End Sub

Private Sub cmdStdSrcPic_Click()

lstStdInfo.ListItems.Clear
txtS_StdID.Text = ""
txtS_StdName.Text = ""
txtS_Admission.Text = ""
DTs_year.Value = Date
chkStdInfo.Value = 0

picStdSearch.ZOrder
picStdSearch.Visible = True

txtS_StdID.SetFocus

End Sub

Private Sub Form_Load()

txtClassYear.Value = Date
DTPicker1.Value = Date
SSTab1.Tab = 0

End Sub

Private Sub lstClass_DblClick()
Dim rs As ADODB.Recordset
Dim myLst As ListItem
Dim sqlStr As String
Dim tmpClassID As Integer
Dim tmpSTDnum As String

tmpClassID = lstClass.SelectedItem.Text
lstStudents.ListItems.Clear
lblClassID.Caption = ""
lblClassName.Caption = ""
lblClassTeacher.Caption = ""
lblClassYear.Caption = ""
lblStdCount.Caption = ""

sqlStr = "SELECT * FROM lstClass WHERE CLASSid = " & tmpClassID

Set rs = myDB(sqlStr, 1)

If Not rs.EOF Then
   lblClassID = rs!CLASSid
   lblClassName = rs!ClassName
   lblClassTeacher = rs!CLASSteacher
   lblClassYear = Format(rs!CLASSyear, "yyyy")
   lblStdCount = "Number of Students Enrolled: " & rs!StdCount

   sqlStr = "SELECT * FROM lstClassInfo WHERE CLASSid = " & tmpClassID

   Set rs = myDB(sqlStr, 1)

   While Not rs.EOF
      tmpSTDnum = Format(rs!STDadmissionnum, "000000") & "/" & Format(rs!STDstampdate, "yy")
      Set myLst = lstStudents.ListItems.Add(, , tmpSTDnum)
         myLst.SubItems(1) = rs!STDid
         myLst.SubItems(2) = rs!STDname
         myLst.SubItems(3) = rs!Type
         myLst.SubItems(4) = rs!STDreligion
         myLst.SubItems(5) = rs!STDrace
      rs.MoveNext
   Wend

   SSTab1.Tab = 1
End If
End Sub
```

```
Private Sub lstStdInfo_DblClick()
Dim rs As ADODB.Recordset
Dim myLst As ListItem
Dim strSql As String

On Error Resume Next

strSql = "SELECT * FROM lstStdInfo WHERE STDid ='" & lstStdInfo.SelectedItem.SubItems(1) & "'"

Set rs = myDB(strSql, 1)

If rs.EOF Then Exit Sub

Call cmdStdClear_Click
picStdSearch.Visible = False

lblSTDid = ""
lblSTDname = ""
lblSTDsex = ""
lblSTDrace = ""
lblSTDreligion = ""
lblSTDbirthcert = ""
lblSTDdob = ""
lblSTDpob = ""
lblSTDadmissionNum = ""
lblSTDfathername = ""
lblSTDfatheradd = ""
lblSTDguardianname = ""
lblSTDguardianadd = ""
lblSTDguardiantype = ""
lblSTDleaveDate = ""
lblSTDcertnum = ""
lblSTDleavereason = ""
optSTDsijilYes.Value = False
optSTDsijilNo.Value = False
lblSTDactive.Caption = ""

lblSTDid = rs!STDid
lblSTDname = rs!STDname
lblSTDsex = rs!Type
lblSTDrace = rs!STDrace
lblSTDreligion = rs!STDreligion
lblSTDbirthcert = rs!STDbirthcert
lblSTDdob = rs!STDdob
lblSTDpob = rs!STDpob
lblSTDadmissionNum = Format(rs!STDadmissionnum, "000000") & "/" & Format(rs!STDstampdate, "yy")
lblSTDfathername = rs!STDfathername
lblSTDfatheradd = rs!STDfatheradd
lblSTDguardianname = rs!STDguardianname
lblSTDguardianadd = rs!STDguardianadd
lblSTDguardiantype = rs!STDguardiantype
lblSTDleaveDate = Format(rs!STDleavedate, "dd/mmm/yyyy")
lblSTDcertnum = rs!STDcertnum
lblSTDleavereason = rs!STDleavereason

If rs!STDsijil Then
   optSTDsijilYes.Value = True
Else
   optSTDsijilNo.Value = True
End If

If rs!STDactive Then
   lblSTDactive.ForeColor = vbBlue
   lblSTDactive.Caption = "ACTIVE"
Else
   lblSTDactive.ForeColor = vbRed
   lblSTDactive.Caption = "NOT ACTIVE"
End If

End Sub
```

```
Private Sub lstStudents_DblClick()
Dim rs As ADODB.Recordset
Dim myLst As ListItem
Dim strSql As String

'On Error GoTo errHandler
On Error Resume Next

strSql = "SELECT * FROM lstStdInfo WHERE STDid ='" & lstStudents.SelectedItem.SubItems(1) & "'"

Set rs = myDB(strSql, 1)

If rs.EOF Then Exit Sub

lblSTDid = ""
lblSTDname = ""
lblSTDsex = ""
lblSTDrace = ""
lblSTDreligion = ""
lblSTDbirthcert = ""
lblSTDdob = ""
lblSTDpob = ""
lblSTDadmissionNum = ""
lblSTDfathername = ""
lblSTDfatheradd = ""
lblSTDguardianname = ""
lblSTDguardianadd = ""
lblSTDguardiantype = ""
lblSTDleaveDate = ""
lblSTDcertnum = ""
lblSTDleavereason = ""
optSTDsijilYes.Value = False
optSTDsijilNo.Value = False
lblSTDactive.Caption = ""

SSTab1.Tab = 2

lblSTDid = rs!STDid
lblSTDname = rs!STDname
lblSTDsex = rs!Type
lblSTDrace = rs!STDrace
lblSTDreligion = rs!STDreligion
lblSTDbirthcert = rs!STDbirthcert
lblSTDdob = rs!STDdob
lblSTDpob = rs!STDpob
lblSTDadmissionNum = Format(rs!STDadmissionnum, "000000") & "/" & Format(rs!STDstampdate, "yy")
lblSTDfathername = rs!STDfathername
lblSTDfatheradd = rs!STDfatheradd
lblSTDguardianname = rs!STDguardianname
lblSTDguardianadd = rs!STDguardianadd
lblSTDguardiantype = rs!STDguardiantype
lblSTDleaveDate = Format(rs!STDleavedate, "dd/mmm/yyyy")
lblSTDcertnum = rs!STDcertnum
lblSTDleavereason = rs!STDleavereason

If rs!STDsijil Then
   optSTDsijilYes.Value = True
Else
   optSTDsijilNo.Value = True
End If

If rs!STDactive Then
   lblSTDactive.ForeColor = vbBlue
   lblSTDactive.Caption = "ACTIVE"
Else
   lblSTDactive.ForeColor = vbRed
   lblSTDactive.Caption = "NOT ACTIVE"
End If

'Exit Sub
```

```
'errHandler:
'MsgBox "An Error has occured while executing the search module! " & vbLf & _
'     "Error Number: " & Err.Number & vbLf & _
'     "Error Desc: " & Err.Description & " ", vbOKOnly + vbExclamation, "Error Message"

End Sub


Module1.bas-----------------------------------------------------------

Public UserName , UserID As String ', myBLK As Integer
Public myLastFile As String, LastFile As String
Public myTarget As TextBox
Public iniDriver, iniServer, iniUID, iniPwd, iniDatabase As String
Public goDNS As Boolean

Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hwnd As Long, ByVal lpOperation As String,
ByVal lpFile As String, ByVal lpParameter As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

Public Function myDB(SQLtxt As String, DBmode As Integer) As ADODB.Recordset
Dim db As New ADODB.Connection
Dim rs As New ADODB.Recordset
Dim LockType As String

'On Error GoTo errHandler

If goDNS Then
  Set db = New ADODB.Connection
  db.ConnectionString = "DSN=REKOD;UID=sa;PWD="
  db.CursorLocation = adUseClient
  db.Open
Else
  Set db = New ADODB.Connection
  db.ConnectionString = "DRIVER=" & iniDriver & ";SERVER=" & iniServer & ";DSN=conDB;UID=" & iniUID & ";PWD=" &
iniPwd & ";DATABASE=" & iniDatabase
  db.CursorLocation = adUseClient
  db.Open

End If

  Set rs = New ADODB.Recordset
  rs.Open SQLtxt, db, adOpenDynamic, DBmode, adCmdText

  Set myDB = rs

Exit Function

errHandler:

   MsgBox "Database not connected." & vbCr & _
   "Please check the server and retry." & _
   vbCr & "Error Number " & Err.Number, vbInformation, "Sistem Fail MBI"

End Function

Public Function myForm(FormName As String) As Boolean
Dim i As Integer, fnamelc As String

fnamelc = LCase$(FormName)
myForm = False

For i = 1 To Forms.Count - 1
   If LCase$(Forms(i).Name) = fnamelc Then myForm = True
Next
End Function


Function noblank(iTems As String, iType As Integer) As String

   If iType = 0 Then
```

```
    If iTems <> "" Then
      noblank = iTems
    Else
      noblank = "-"
    End If
  ElseIf iType = 1 Then
    If iTems <> "" Then
      noblank = CSng(iTems)
    Else
      noblank = 0
    End If
  ElseIf iType = 2 Then
    If iTems <> "0" Then
      noblank = "TRUE"
    Else
      noblank = "FALSE"
    End If
  ElseIf iType = 3 Then
    If iTems <> "TRUE" Then
      noblank = "0"
    Else
      noblank = "1"
    End If
  End If

End Function

Sub Main()
On Error GoTo ErrorHandler
Dim myINI As SSINI.CIni
Dim myFile As FileSystemObject

  Set AppObject = App
  Set ErrObject = Err
  CallStack = True
  LogPath = App.Path

  goDNS = False

  Set myFile = New FileSystemObject

  If myFile.FileExists(App.Path & "\sistem.ini") = True Then
    Set myINI = New SSINI.CIni
    myINI.INIFile = App.Path & "\sistem.ini"
    myINI.INISection = "SYSTEM"
    iniDriver = myINI.GetINISetting("DBDRIVER")
    iniServer = myINI.GetINISetting("SERVER")
    iniUID = myINI.GetINISetting("UID")
    iniPwd = myINI.GetINISetting("PWD")
    iniDatabase = myINI.GetINISetting("DATABASE")
  Else
    goDNS = True
  End If

  'frmFileNum.Show
  frmLogin.Show
  'MDIForm1.Show

  Exit Sub

ErrorHandler:
    goDNS = True

End Sub

Function Crypt(Data As String)
Dim a As Long
Dim b As Long
Dim c As Long
```

```
Dim d As Long
Randomize 12345

For a = 1 To Len(Data)

   b = Asc(Mid$(Data, a, 1))

   If a / 2 = Int(a / 2) Then d = 5 Else d = -7

   c = b Xor (Int(Rnd(0) * 255) + a + d)
   Crypt = Crypt & Chr$(c)

Next
End Function

Function genFileID(mySelect As Long, myName As String) As String
Dim rs As ADODB.Recordset
Dim tmpFileID As String

If mySelect = 0 Then 'normal
   Set rs = myDB("SELECT * FROM IDlist WHERE IDuser = '-' ORDER BY FILEid", 3)

   If rs.EOF Then
      tmpFileID = getNewNumber
      rs.AddNew
      rs!FILEid = tmpFileID
      rs!IDuser = myName
      'rs!BLKid = mySelect
      rs.Update
   Else
      tmpFileID = rs!FILEid
      rs!IDuser = myName
      'rs!BLKid = mySelect
      rs.Update
   End If

Else 'block number
   Set rs = myDB("SELECT * FROM lstIDopen WHERE BLKid = '" & mySelect & "'", 1)

   If rs.EOF Then
      Set rs = myDB("SELECT * FROM IDlist WHERE IDuser = '-' ORDER BY FILEid", 3)

      If rs.EOF Then
         tmpFileID = getNewNumber
         rs.AddNew
         rs!FILEid = tmpFileID
         rs!IDuser = myName
         rs!BLKid = mySelect
         rs.Update
      Else
         tmpFileID = rs!FILEid
         rs!IDuser = myName
         rs!BLKid = mySelect
         rs.Update
      End If

   Else
      tmpFileID = rs!FILEid
   End If

End If

genFileID = tmpFileID

End Function

Function getNewNumber() As String
Dim rs As ADODB.Recordset
Dim tmpID As String, tmpChar As String
Dim tmpNum As Long
```

```
Set rs = myDB("SELECT * FROM IDlist ORDER BY FILEid DESC", 1)

If Not rs.EOF Then
   tmpID = rs!FILEid
   tmpChar = Left(tmpID, 1)
   tmpNum = Val(Right(tmpID, 5))
   tmpNum = tmpNum + 1

   If tmpNum > 99999 Then
      tmpChar = Chr(Asc(tmpChar) + 1)
      tmpNum = 1
   End If

   getNewNumber = tmpChar & "/" & Format(tmpNum, "00000")
Else
   getNewNumber = "A/00001"
End If

End Function
```