

Final Year Project Report
On
Student Information System (SIS)
(A desktop application)

Prepared By

Muhammad Shahid

BCS-15-03

Session: 2015-19

Morning

Supervised By

Dr.Shafiq Hussain

Department of Computer Science



Bahauddin Zakariya University Multan,
Campus Sahiwal

DEDICATION

Our Parents and Teachers all who've given us their support during the development of this project and for giving good ideas to prove ourselves as intellectuals in front of our Respected Teachers.

ACKNOWLEDGEMENT

Praise to Allah Almighty, Lord of the worlds, the Merciful and the Beneficent, who gave me strength, thoughts and co-operative people to enable me to accomplish this goal and fulfill the required functionalities.

This was all not possible without the guidance, continuous appreciation and moral support by “**Dr.Shafiq Hussain**”. He was always there whenever I need his help and ideas. I am really thankful to him who made my concepts clearer.

At last, I would like to acknowledge all of the assistance and contributions of “**Bahauddin Zakariya University**” for supporting us with all that is needed starting from the books, and ending with the full care that it is providing us with, to help us to be professionals in the field of Computer Science.

DECLARATION

I hereby declare that I have developed this application and accompanied report entirely on the basis of my personal efforts. Not any of the portions of the application work presented has been submitted of any application for any other qualification or degree of this or any other university or institute of learning.

Student Name & Signature

Muhammad Shahid

CERTIFICATE OF APPROVAL

It is to certify that the final year project of BS (CS) “**Student Information System** (A desktop application)” was developed by “**Muhammad Shahid**” under the supervision of “**Dr. Shafiq Hussain**” and that in his opinion, it is in scope, fully adequacy and quality of the degree of Bachelors of Science in Computer Sciences.

Supervisor

Dr. Shafiq Hussain

Assistant Professor

Department of Computer Science

BZU, Sahiwal

External Examiner

Head of Department

Dr. Shafiq Hussain

Department of Computer Science

BZU, Sahiwal

Abstract

Student information System is a model system to store information about students, teachers, students result, classes, subjects, fees, attendance. The system is designed to meet the purpose of dealing with student information system.

As project mainly concentrate on student information system so keeping the friendly user interface the system should provide all necessary student information facilities. It provide the Administrator the facility to save, update, delete and view records of students, teachers, attendance and result.

Table of Contents

Chapter 1	1
Introduction.....	1
1.1. Project Overview.....	1
1.2. Background study.....	1
1.2.1. Historical perspective.....	2
1.3. Scope	3
1.3.1. Modules.....	3
1.4. Features of Proposed System	5
1.5 Tools & Techniques.....	6
Chapter 2	7
Requirement Analysis.....	7
2.1. Software Requirement Specification.....	7
2.1.1. Introduction.....	7
2.1.2. Purpose.....	7
2.1.3. Proposed System.....	8
2.2. Specification Requirements.....	10
2.2.1. Functional Requirements	10
2.2.2. Non-Functional Requirements	11
2.2.3. Optional Requirements	11
2.3. Project Objective	11
Chapter 3	12
METHODOLOGY & WORKPLAN	12
3.1 What is Mythology and why we need it?.....	12
3.2 Adopted Methodology	12
3.3 Roles & Responsibilities.....	13
Chapter 4	14
Software Design.....	14
4.1 Activity Diagram.....	14
4.1.1 Login Activity Diagram for Admin.....	14
4.1.2 Activity Diagram for Student Information System	15

4.2	Sequence Diagram.....	16
4.2.1	SIS Sequence Diagram	16
4.2.2	User Login	17
4.2.3	Add Department.....	18
4.2.4	Add Class	19
4.2.5	Add Course	20
4.2.6	Add Student Information	21
4.2.7	Add Teacher Information.....	22
4.2.8	Add Student Result	23
4.2.9	Add Student Attendance	24
4.2.10	Add Student Fee Details	25
4.3	Use Case Diagram.....	26
4.3.1	Use Case Description.....	27
	<i>Use Case UC1: Login</i>	27
	Main Success scenario:	27
	Extension:	27
	<i>Use Case UC2: Manage Department</i>	28
	Main Success scenario:	28
	Extension:	28
	<i>Use Case UC3: Manage Class</i>	29
	Main Success scenario:	29
	Extension:	29
	<i>Use Case UC4: Manage Course</i>	30
	Main Success scenario:	30
	Extension:	30
	<i>Use Case UC5: Manage Student Information</i>	31
	Main Success scenario:	31
	Extension:	31
	<i>Use Case UC6: Manage Student Result</i>	32
	Main Success scenario:	32
	Extension:	32

<i>Use Case UC7: Manage Teacher Information</i>	33
Main Success scenario:	33
Extension:	33
<i>Use Case UC8: Manage Attendance</i>	34
Main Success scenario:	34
Extension:	34
Main Success scenario:	35
Extension:	35
<i>Use Case UC10: Logout</i>	36
Main Success scenario:	36
Extension:	36
4.4 Class Diagram	37
4.5 ERD Diagram	38
Chapter 5	39
Implementation	39
5.1 Introduction	39
5.2 Tools and Technology Used.....	39
5.2.1 Tools	39
5.2.2 Technology	40
5.2.3 Planning & Scheduling	40
Gantt chart.....	40
Chapter 6	63
Database Design	63
6.1 Database Design.....	63
6.2 Database	63
6.3 Database tables.....	67
Chapter 7	72
User Interface	72
7.1 Introduction	72
7.2 User Interface Front End.....	73
7.2.1 Login	73

7.2.2 Dashboard Page	74
7.2.3 Department Page	74
7.2.4 Course Page	75
7.2.5 Course Mapping Page	75
7.2.6 Student Register Page.....	76
7.2.7 Student View page.....	76
7.2.8 Teacher Register Page	77
7.2.9 Teacher View Page.....	77
7.2.10Add Result	78
7.2.11 View Result.....	78
7.2.12 Add Attendance.....	79
7.2.13 View Attendance	79
7.2.14 Fee Module	80
Chapter 8	81
SYSTEM TESTING	81
8.1 Introduction.....	81
8.2 Testing Plan	81
8.2.1 Unit Testing	81
8.2.2 System Testing.....	82
8.2.3 Integration Testing	82
8.2.4 User Acceptance Testing	82
8.3 Test Cases	83
8.4 Testing Results.....	84
Chapter 9	86
Conclusion & Future Work	86
9.1 Conclusion	86
9.2 Future Work	86
REFERENCES:	87

Table of Figures

Figure 3.1:Adopeted Methodology	13
Figure 4.1.1: Login Activity Diagram	14
Figure 4.1.2: Student Information System Activity Diagram.....	15
Figure 4.2.1: Student Information System Sequece Diagram.....	16
Figure 4.2.2: Login Sequece Diagram	17
Figure 4.2.3: Add department Sequece Diagram.....	18
Figure 4.2.4: Add class Sequece Diagram	19
Figure 4.2.5: Add Course Sequece Diagram	20
Figure 4.2.6: Add Student Sequece Diagram.....	21
Figure 4.2.7: Add Teacher Sequece Diagram.....	22
Figure 4.2.8: Add Student Result Sequece Diagram	23
Figure 4.2.9: Add Student Attedance Sequece Diagram	24
Figure 4.2.10: Add Student Fee Sequece Diagram.....	25
Figure 4.3: Student Information System Use Case Diagram.....	26
Figure 4.4: Student Information System Class Diagram	37
Figure 4.5: Student Information System ERD Diagram.....	38
Figure 6.3.1: Admin Database Table(screenshot).....	67
Figure 6.3.2: Department Database Table(screenshot).....	67
Figure 6.3.3: Class Database Table (screenshot)	68
Figure 6.3.4: Coursse Database Table(screenshot).....	68
Figure 6.3.5: CourseMapping Database Table(screenshot).....	69
Figure 6.3.6: Student Database Table(screenshot)	69
Figure 6.3.7: Teacher Database Table(screenshot).....	70
Figure 6.3.8: Student Attendance Database Table(screenshot)	70
Figure 6.3.9: Student Result Database Table(screenshot)	71
Figure 6.3.10: Student Fee Database Table(screenshot).....	71
Figure 7.2.1: Login Page(screenshot)	73
Figure 7.2.2: Dassh Board Page(screenshot)	74
Figure 7.2.3: Department Page(screenshot).....	74
Figure 7.2.4: Course Page(screenshot)	75
Figure 7.2.5: Course Mapping Page(screenshot).....	75
Figure 7.2.6: Student Registration Page(screenshot)	76
Figure 7.2.7: Student View Page(screenshot).....	76
Figure 7.2.8: Teacher Registration Page(screenshot)	77
Figure 7.2.9: Teacher View Page(screenshot)	77
Figure 7.2.10: Add Result Page(screenshot).....	78
Figure 7.2.11: View Result Page(screenshot).....	78
Figure 7.2.12: Student Attendance Page(screenshot)	79
Figure 7.2.13: Student Attendance View Page(screenshot).....	79
Figure 7.2.14: Student Fee Record Page(screenshot)	80

Chapter 1

Introduction

1.1. Project Overview

Student data are the main element for any educational institute. “Student Information Management System” provides us a simple interface for maintenance of student information. It can be used by educational institutes or colleges to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using Student Information Management System project. Throughout the project the focus has been on presenting information in an easy and intelligible manner. The project is very useful for school, colleges, academies and universities. The project provides facilities like department management, class management and course management, registration and profile creation of student as well as managing the student academic record and result thus reducing paperwork and automating the record generation process in an educational institution.

The objective of Student information System is to allow the administrator to edit and find out the academic record as well as students personal information. It’ll also facilitate keeping all the records of students, such as their id, name, mailing address, phone number, DOB etc. So, all the information about a student will be available in a few seconds. Overall, it’ll make Student Information Management an easier job for the administrator to manage the students in institute. The main purpose of this SRS document is to illustrate the requirements of the project Student information System and is intended to help any organization to maintain and manage its student’s academic record as well as personal data.

1.2. Background study

Technology evolution played an important role in the success of many businesses and helping business to become more competitive in the industry as well as the economy. Better quality of life is often achieved with the effect of the creation of IT. As technology continues to evolve, computer is largely dominating the work of a lot of people. We are now benefiting the fruit of its

wonders, advancement of connectivity, communication, accessible online devices were invented to enable man to add, subtract and to record simple transactions.

Student Information Management System is used in recording a student's information. A well-built one will reduce the load on the people that normally have to do all the work. Student Information Management System is a good step for a school. It is useful especially when the school retrieves the important information from the student. Through it the school can trace what are the standings of the students. Lack of enrollment system in a school can lead to chaos and troubles, the students will be confused on what they should do and how they will do to be able to enroll. It is extremely useful in the school in the way of making the processes of enrolling much easy.

1.2.1. Historical perspective

The Student Information Management System is the SIS that will hold the business in its goal to a paperless office environment. From the acronym itself, the SIS is that management process that will protect information and data to ensure that all the records and the files in the system database of the business organization is protected and easily accessed by users in the framework. In the shortest definition of the automated records management system, it is simply that process that is employed by the organization in the maintenance of records and files from the very moment that they arrive to the business or were created by the enterprise up to the period when they will be deemed ready for eradication and deletion from the archived or active databases. The records handled by the Student Information Management System may come in tangible forms that will be translated to digital formats with the use of the latest technologies in the market.

According to Muhenda & Lwanga (2000) noted that Student record used for storing large database or knowledge base. It can be used for knowing the current status of any aspect of the business due to its on-line real time processing capability. As governance issues take center stage in the management of Educational Institutions, the management of students' records becomes imperative in the improvement of services offered in Higher Educational Institutions (HEI's) in Uganda. Proper records management underpins policy formulation, decision making, protects interests of organization, and protects rights of employers and students in addition to helping Institutions conduct business and deliver services in a consistent and equitable manner. There are five Areas for Student Information Management System which includes:

- The Student Information Management System will affect in its inception to the business organization. Such as the system will work in the active records section. This mean that it can give you access and help you manage the most current documents and data that the operations are handling.
- The second sector would be the data protection and jumping, which is the area that will ensure all the files are properly filed and archived for record keeping purposes. In time, the next area and sector would be the determination of destruction of the documents in a secured manner that will ensure that all of the data from the files will still be kept secret and confidential.
- The other sections would be the records management area and the document management and imaging.

The student records management system will give the business organization streamlined operations that will reduce paper based transactions to the bare minimum. Simply, the automated records management system is a method that will do all the filing and management of documents for the organization.

1.3. Scope

Without Student information Management System, managing and maintaining the details of the student is a tedious job for any Institute. Student Information system will store all the details of the students including their academic result course offered background information, educational qualifications, personal. Student Information System also manage departments, classes and course offered in an institute.

1.3.1. Modules

1. Login module
2. Student registration Module
3. Student Information Module
4. Department Management Module
5. Class Management Module
6. Subject/Course Management Module
7. Student Result Management Module

8. Student Attendance Module
9. Student Fee Module
10. Teacher Registration Module
11. Teacher Profile View Module

1.3.1.1. Login Module

Login module will help in authentication of user account. User who has valid login id and password can only login into their respective accounts.

1.3.1.2. Student Registration Module

Student Registration module allow the administrator to add the student personal detail like name, father name, data of birth, CNIC, cell number, email and student address, department, class, session and registration date. The administrator can add, delete, and update the student information with ease.

1.3.1.3. Student View Module

Student view module allows the administrator to view the records of students registered in different departments. Administration can also view the single student details by using student id.

1.3.1.4. Department Management Module

Department management module will help the administrator to manage the departments in the institute or organization. The administrator can add new department, delete, and update and view the department detail.

1.3.1.5. Class Management Module

Class management module will help the administrator to manage the Class in the institute. The administrator can adds new Class, delete, and update and view the Class detail with ease.

1.3.1.6. Subject/Course Management Module

Course management module allows the user to add course detail like course name, course code, course credit hour. Course management module will help the administrator to

manage and maintain all the courses. The administrator can add new Course, delete, and update and view the course detail with ease.

1.3.1.7. Student Result Management Module

Student result management module allows the user to mark the result of student in each subject/course.

1.3.1.8. Student Attendance Module

Student attendance module allows the admin to enter, update and view the attendance of a student in each subject/course.

1.3.1.9. Student Fee Module

Student fee module allows the admin to keep record of students' fee.

1.3.1.10. Teacher Registration Module

Teacher Registration module allow the administrator to add the teacher personal detail like name, father name, data of birth, CNIC, cell number, email and student address, department, and registration hiring date. The administrator can add, delete, update and view the teacher information with ease.

1.3.1.11. Teacher View Module

Teacher view module allows the administrator to view the records of teachers registered. Administration can also view the single teacher details by using student id.

1.4. Features of Proposed System

Below is the list of main features of proposed system.

1. User login
2. Maintain student information (save, delete, update student information and also search the specific record).
3. Maintain student result (save, update student result record and also search the specific record).
4. Maintain department detail (save, delete, update department information and also search the specific record).

5. Maintain class detail (save, delete, update class information and also search the specific record).
6. Maintain course detail (save, delete, update course information and also search the specific record).
7. Maintain student attendance record(save, update and view attendance of a specific student or whole class)
8. Maintain student fee record(save, update and search the fee details of a specific student or whole class)
9. Maintain Teacher Information(save, delete, update and search the specific teacher details or can view the all teachers' record)

1.5 Tools & Techniques

Student Information System is a desktop application consists of software and hardware tools.

1.5.1 Hardware Specifications:

• Processor	Core 2 Quad (or above)
• RAM	4 GB (or above)
• Hard disk	80 GB(or above)

1.5.2 Software Specifications:

• Operating System	Window XP, Window 7 or above
• IDE	Eclipse Neon
• UI Design Tool	ScenceBuilder
• Database	MySQL
• Language	Java, JavaFX

Chapter 2

Requirement Analysis

2.1. Software Requirement Specification

2.1.1. Introduction

Student data are the main element for any educational institute. “Student Information Management System” provides us a simple interface for maintenance of student information. It can be used by educational institutes or colleges to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using Student Information Management System project. Throughout the project the focus has been on presenting information in an easy and intelligible manner. The project is very useful for school, colleges, academies and universities. The project provides facilities like department management, class management and course management, registration and profile creation of student as well as managing the student academic record and result thus reducing paperwork and automating the record generation process in an educational institution.

2.1.2. Purpose

The current problem in the standard school is did not have systematic data arrangement in the student management. When the staff of administrator wants to record the data of the student (student academic record as well as personal information), they need to fill out by use the manual system, In this case data might be lost when several problem is occur. Other problem that can be happen is hard to search and update the student information and class arrangement. The systematic requirement is required so that all data is stored into the database for future reference and enhancement. Below is the specific problem statement that occurs in standard school via using manual system.

- Lack of data arrangement that is record by using manual system and using a lot of paper to record the student information, student result and performance.
- The manual system is hard to search and update about the student information, result and performance

- The manual system is not providing the security of the academic information that might be lost.

2.1.3. Proposed System

The proposed SIS system provides detail general information about the students along with Educational, SIS is used for adding, viewing information and updating students' details. SIS Emphasizes the system's functionality, database design and functional modules. Some advantages of proposed system are mentioned below:-

- It is very fast and clever.
- Not easy for data loss.
- Short time to knowledge and learning to use operate the system.
- Need short time to find any student information.
- No need for more time.
- Easy to update any data.

2.1.3.1. Modules of Proposed System

1. Login module
2. Student registration Module
3. Student Information Module
4. Department Management Module
5. Class Management Module
6. Subject/Course Management Module
7. Student Result Management Module
8. Student Attendance Module
9. Student Fee Module
10. Teacher Registration Module
11. Teacher Profile View Module

Login Module

Login module will help in authentication of admin account. Admin who has valid login id and password can only login into his accounts.

Student Registration Module

Student Registration module allow the administrator to add the student personal detail like name, father name, data of birth, CNIC, cell number, email and student address, department, class, session and registration date. The administrator can add, delete, and update the student information with ease.

Student View Module

Student view module allows the administrator to view the records of students registered in different departments. Administration can also view the single student details by using student id.

Department Management Module

Department management module will help the administrator to manage the departments in the institute or organization. The administrator can add new department, delete, and update and view the department detail.

Class Management Module

Class management module will help the administrator to manage the Class in the institute. The administrator can adds new Class, delete, and update and view the Class detail with ease.

Subject/Course Management Module

Course management module allows the admin to add course detail like course name, course code, and course credit hours. Course management module will help the administrator to manage and maintain all the courses. The administrator can adds new course, delete, and update and view the course detail with ease.

Student Result Management Module

Student result management module allows the admin to mark the result of student in each subject/course. Admin can also view and update student result.

Student Attendance Module

Student attendance module allows the admin to enter the attendance of a student in each subject/course. Admin can also view and update student attendance.

Student Fee Module

Student fee module allows the admin to keep record of students' fee. It also allows admin to view and update fee record.

Teacher Registration Module

Teacher Registration module allow the administrator to add the teacher personal detail like name, father name, data of birth, CNIC, cell number, email and teacher address, department, class, session and registration date. The administrator can add, delete, update and view the teacher information with ease.

Teacher View Module

Teacher view module allows the administrator to view the records of teacher registered. Administration can also view the single teacher details by using teacher id.

2.2. Specification Requirements

2.2.1. Functional Requirements

1. Administrator shall have rights to maintain student information (save, delete, update student information and also search the specific record).
2. Administrator shall have rights to maintain student result (save, delete, update student result record and also search the specific record).
3. Administrator shall have rights to maintain department detail (save, delete, update department information and also search the specific record).
4. Administrator shall have rights to maintain class detail (save, delete, update class information and also search the specific record).
5. Administrator shall have rights to maintain course detail (save, delete, update course information and also search the specific record).
6. Administrator shall have rights to maintain student attendance detail (save, delete, update course attendance information and also search the specific record).
7. Administrator shall have rights to maintain students' fee detail (save, delete, update fee information and also search the specific record).
8. Administrator shall have rights to maintain teacher detail (save, delete, update teacher information and also search the specific record).

2.2.2. Non-Functional Requirements

2.2.2.1. Performance Requirements

The proposed system that we are going to develop will be used as the chief performance system for providing help to the organization in managing the whole database of the student studying in the institute. Therefore, it is expected that the database would perform functionally all the requirements that are specified.

2.2.2.2. Safety Requirements

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup.

2.2.2.3. Security Requirements:

We are going to develop such system that will secure only administrator access the system after authentication and maintain and manage the student record and perform some other activity.

2.2.3. Optional Requirements

1. Performance requirement
2. User interface consideration
3. Global access and functionality
4. Application and database management
5. User and system documentation and training

2.3. Project Objective

The current problem in the standard school is did not have systematic data arrangement in the student management. When the staff of administrator wants to record the data of the student (student academic record as well as personal information), they need to fill out by use the manual system, In this case data might be lost when several problem is occur. Other problem that can be happen is hard to search and update the student information and class arrangement. The systematic requirement is required so that all data is stored into the database for future reference and enhancement.

Chapter 3

METHODOLOGY & WORKPLAN

3.1 What is Mythology and why we need it?

Project development team is consisting of two members. In order to accomplish a goal, documentation and development is equally distributed among them and each member work on parallel to avoid wastage of time.

Whenever a small or large project has started to develop, first thing all of programmers required is methodology. Methodology is a way of developing a project, in which all of the programmers gather the user's requirements, design the project, implement it, and after all this testing and maintenance of the project, in a satisfaction of user and according to the project requirements.

3.2 Adopted Methodology

Incremental model is used to develop this project, in which we divided our work in multiple modules. All these modules are further divided into more easily managed modules which made up the actual implementation of the requirements.

Reason behind using this model is:

- It is easy to test and debug the product during iterations.
- Software released in increments over time is more likely to satisfy changing user requirements than if it were planned as a single overall release at the end of the same period.
- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it'd iteration.

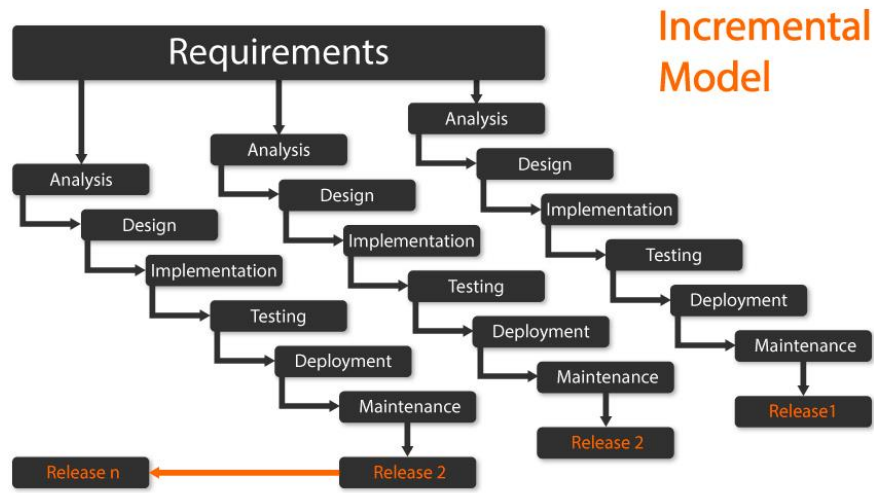


Figure 3.1: Adopted Methodology

Requirement Analysis	<ul style="list-style-type: none"> Requirement and specification of the software are collected
Design	<ul style="list-style-type: none"> Some high-end function are designed during this stage
Code	<ul style="list-style-type: none"> Coding of software is done during this stage
Test	<ul style="list-style-type: none"> Once the system is deployed, it goes through the testing phase

3.3 Roles & Responsibilities

Project development team is consisting of two members. In order to accomplish a goal, documentation and development is equally distributed among them and each member work on parallel to avoid wastage of time.

Chapter 4

Software Design

In this chapter requirements analysis, feasibility study, planning, forecasting, modeling, scheduling and design of the project is discussed. For developing any project, the major problem is requirement gathering. Asking questions from clients is straightforward than collecting requirements. We will also focus on functional and non-functional requirements.

The procedure for gathering requirements has its own defined procedure according to the complexity of the application. To define project schedule and processing, different models and techniques also focused on this chapter.

4.1 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities.

4.1.1 Login Activity Diagram for Admin

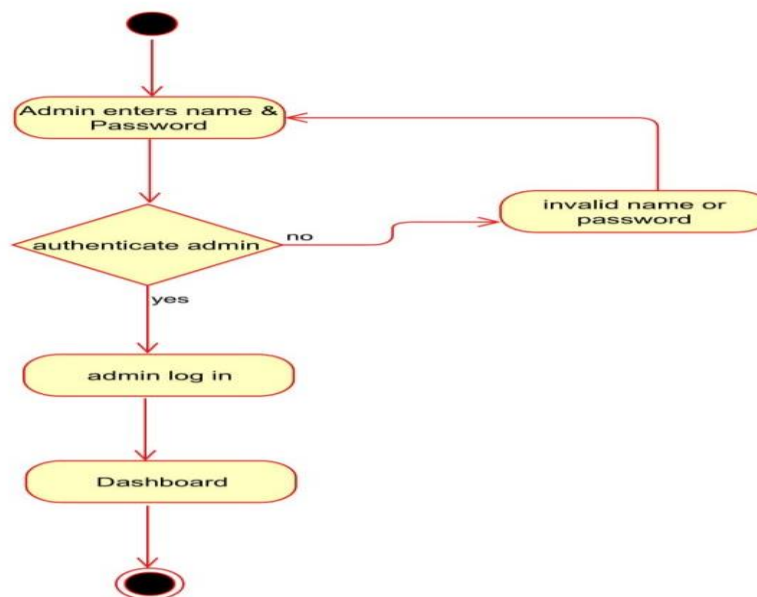


Figure 4.1.1 Admin Login Activity Diagram

4.1.2 Activity Diagram for Student Information System

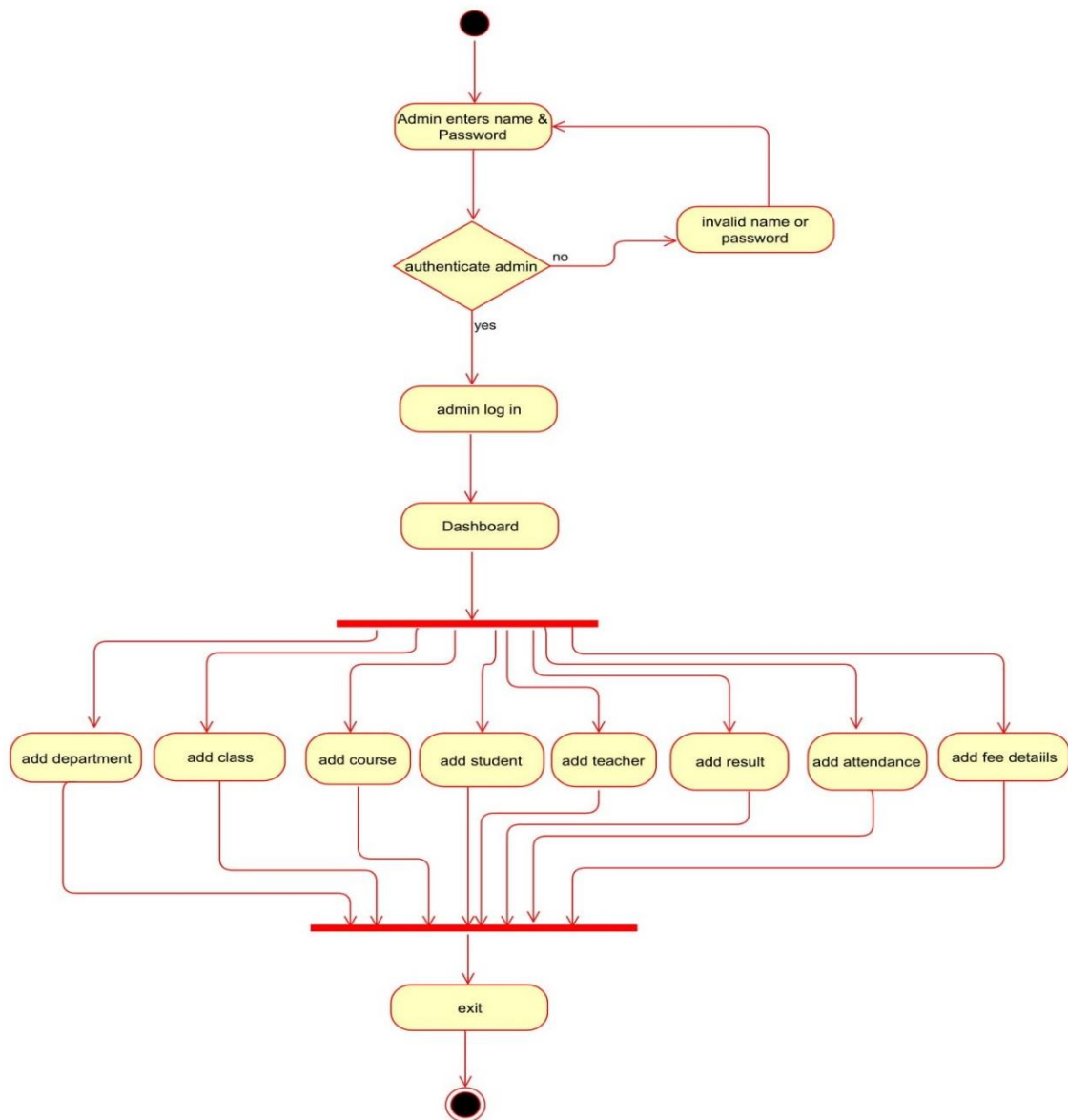


Figure 4.1.2 Student Information System Activity Diagram

4.2 Sequence Diagram

4.2.1 SIS Sequence Diagram

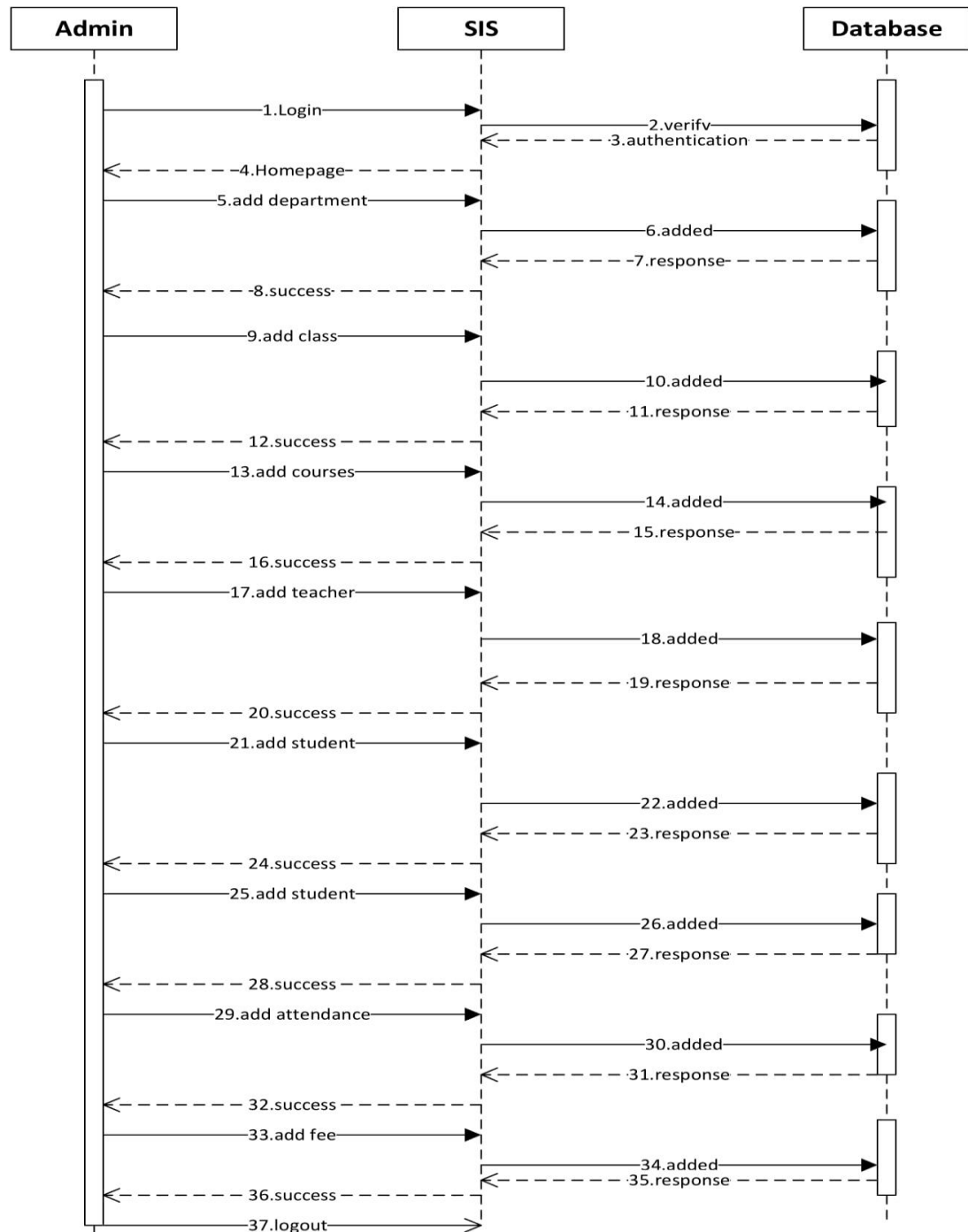


Figure 4.2.1 Sequence Diagram

4.2.2 User Login

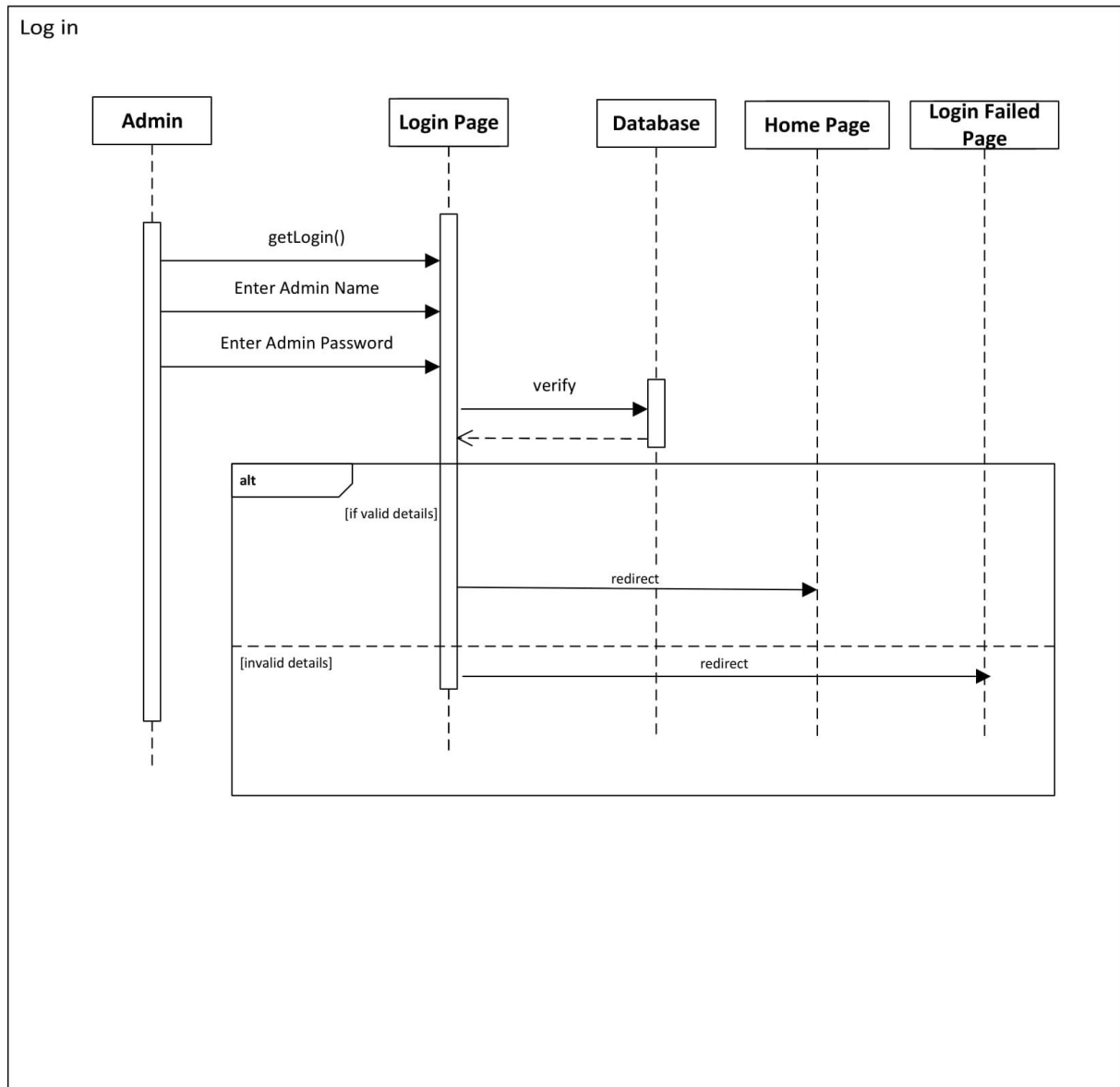


Figure 4.2.2 User Login Sequence Diagram

4.2.3 Add Department

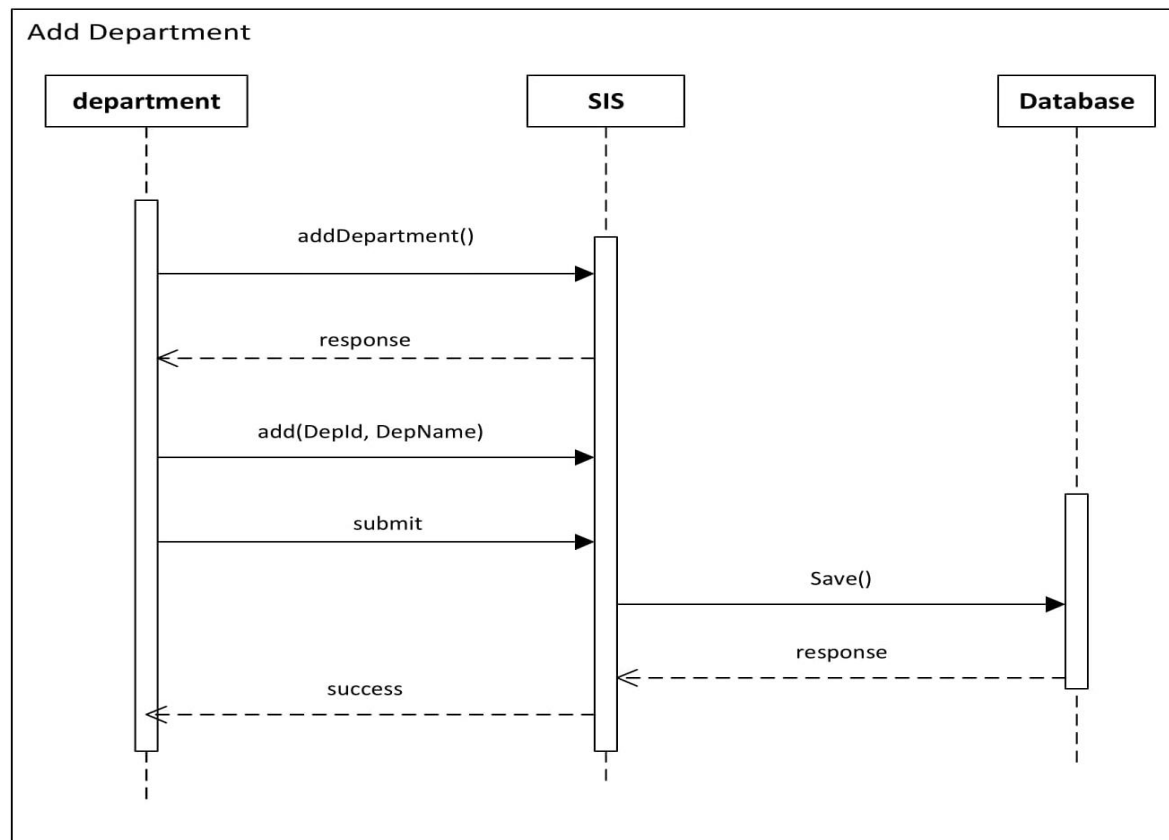


Figure 4.2.3 Add Department Sequence Diagram

4.2.4 Add Class

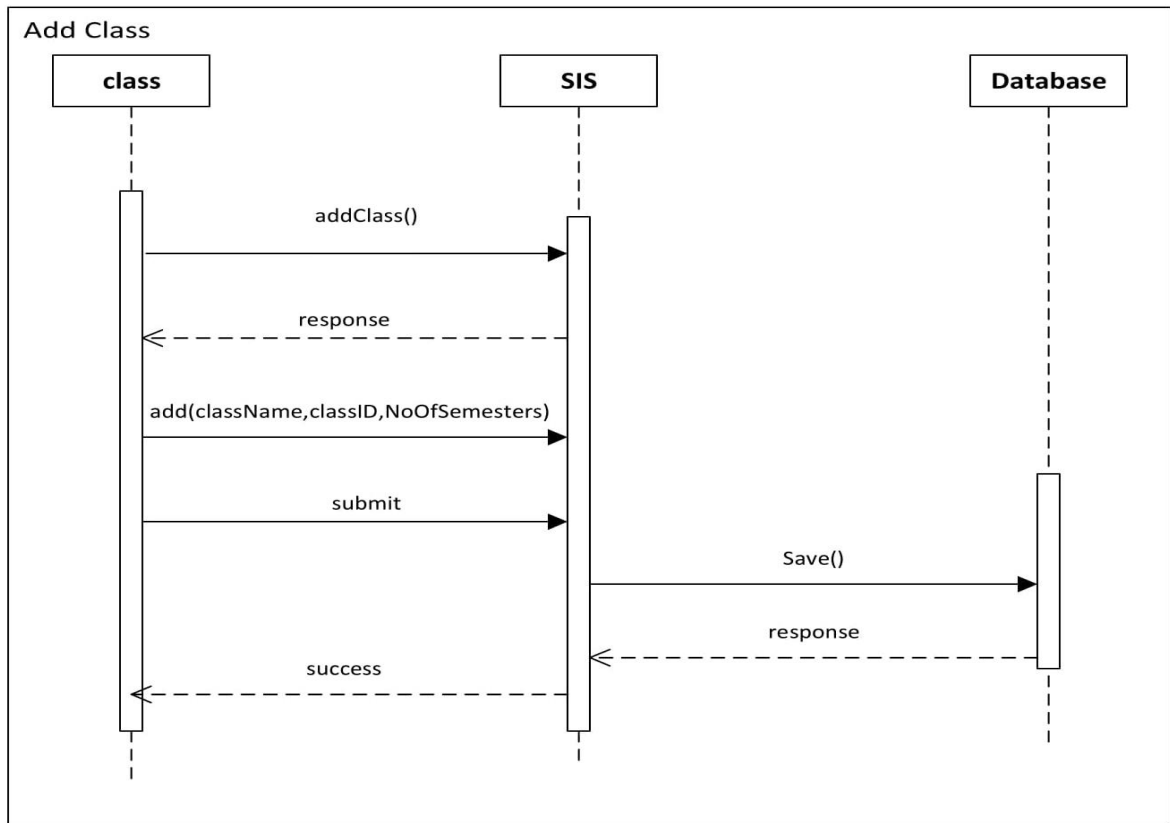


Figure 4.2.4 Add Class Sequence Diagram

4.2.5 Add Course

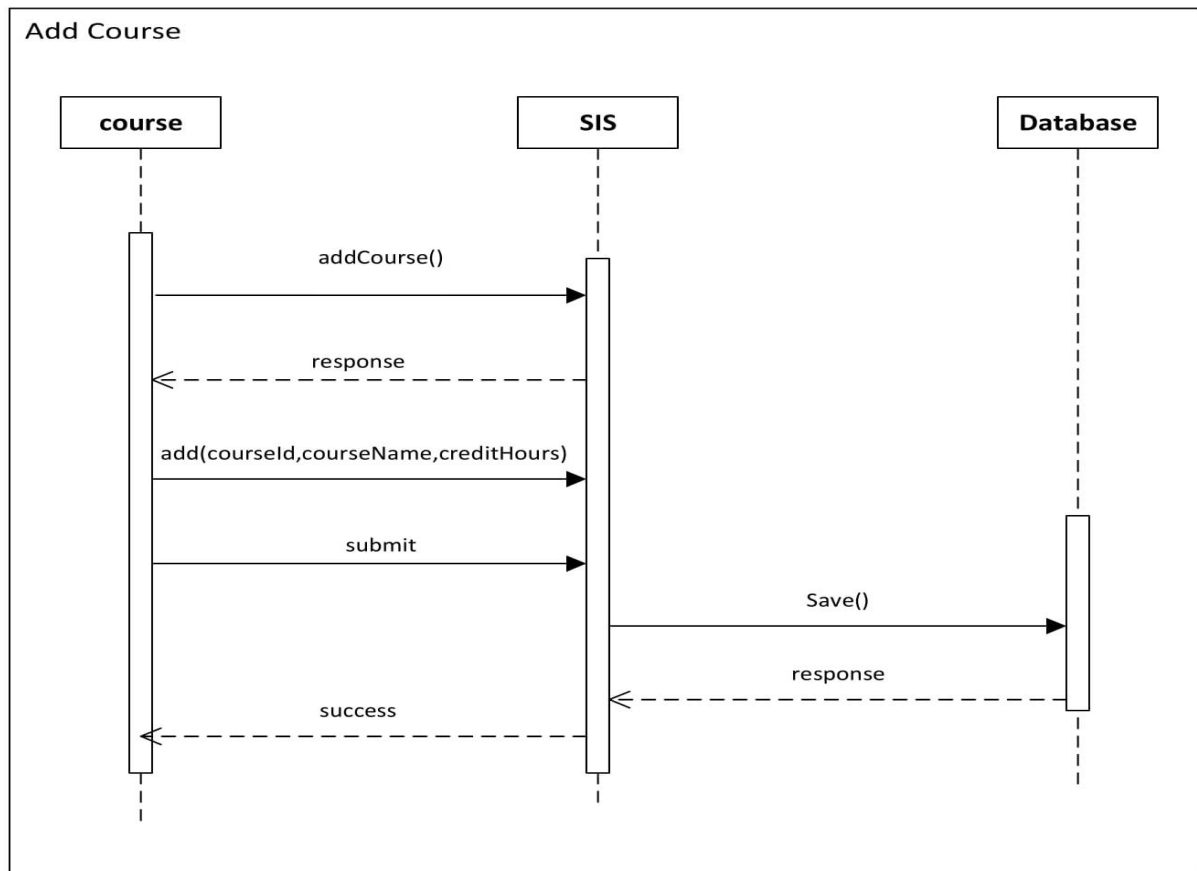


Figure 4.2.5 Add Course Sequence Diagram

4.2.6 Add Student Information

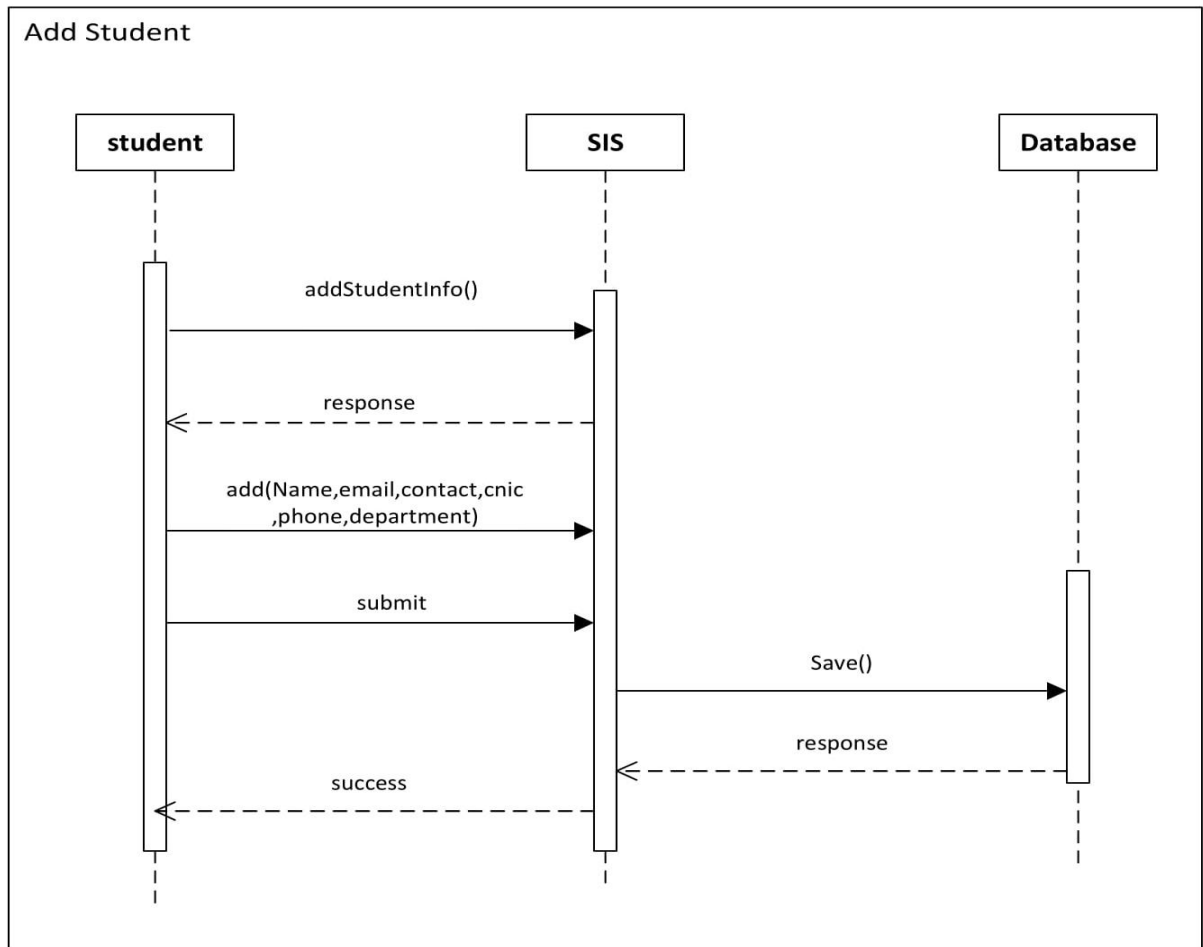


Figure 4.2.6 Student Sequence Diagram

4.2.7 Add Teacher Information

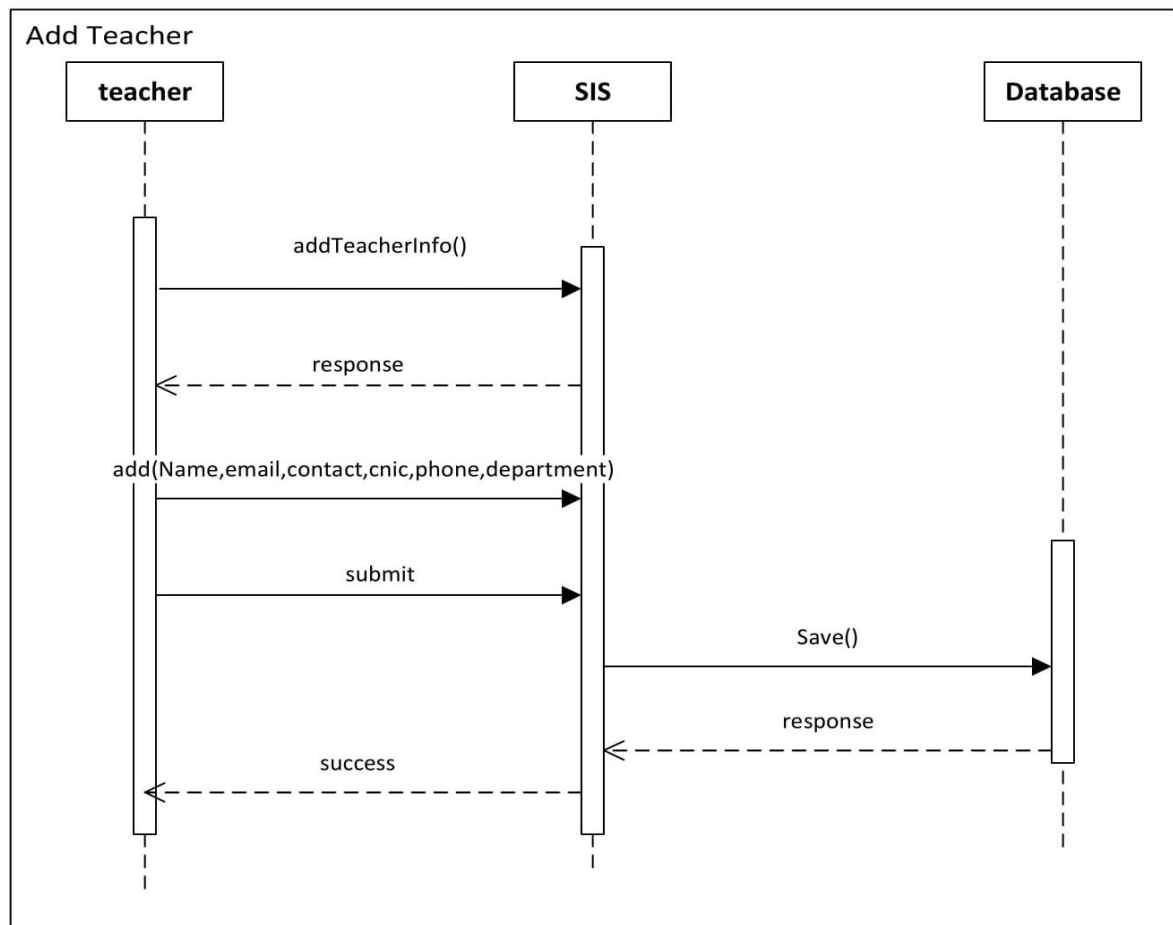


Figure 4.2.7 Teacher Sequence Diagram

4.2.8 Add Student Result

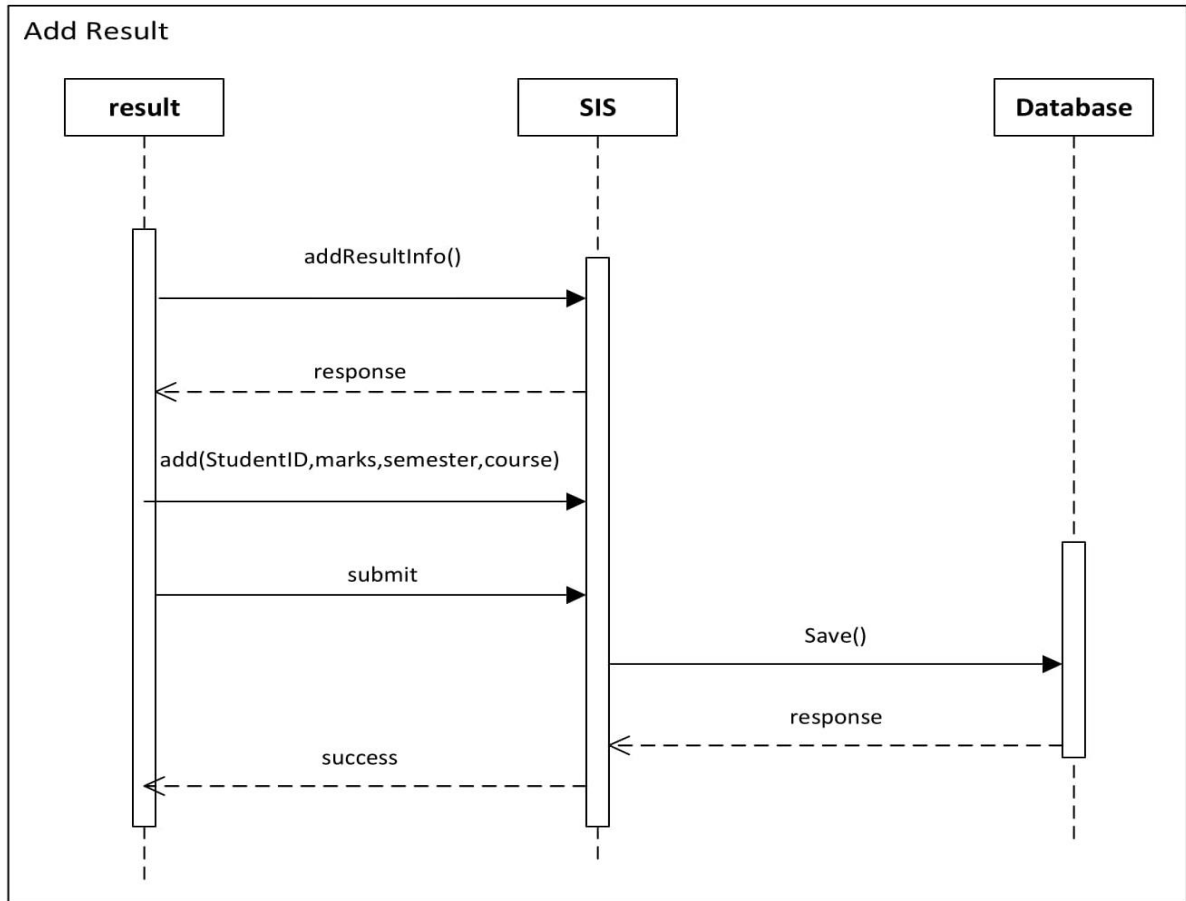


Figure 4.2.8 Student Result Sequence Diagram

4.2.9 Add Student Attendance

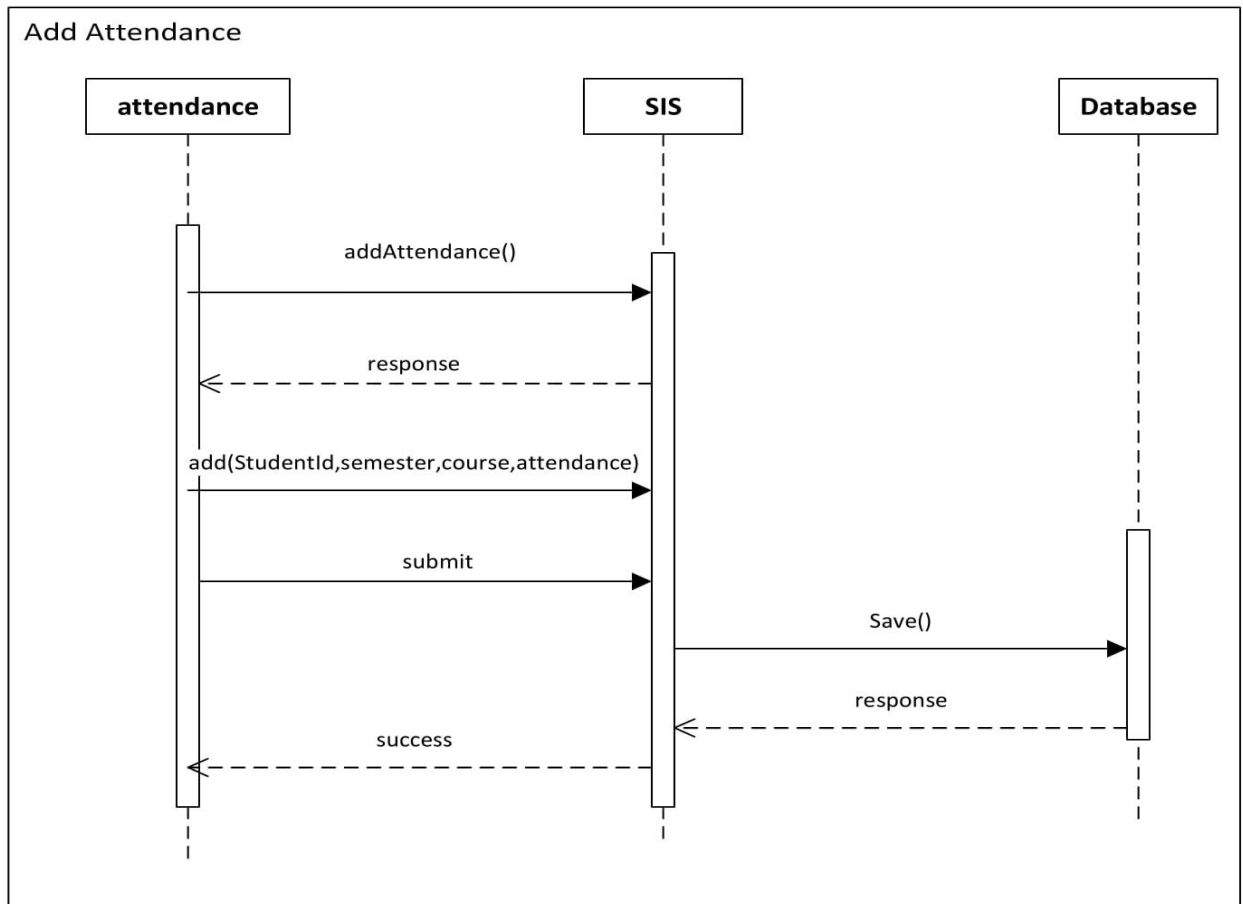


Figure 4.2.9 Student Attendance Sequence Diagram

4.2.10 Add Student Fee Details

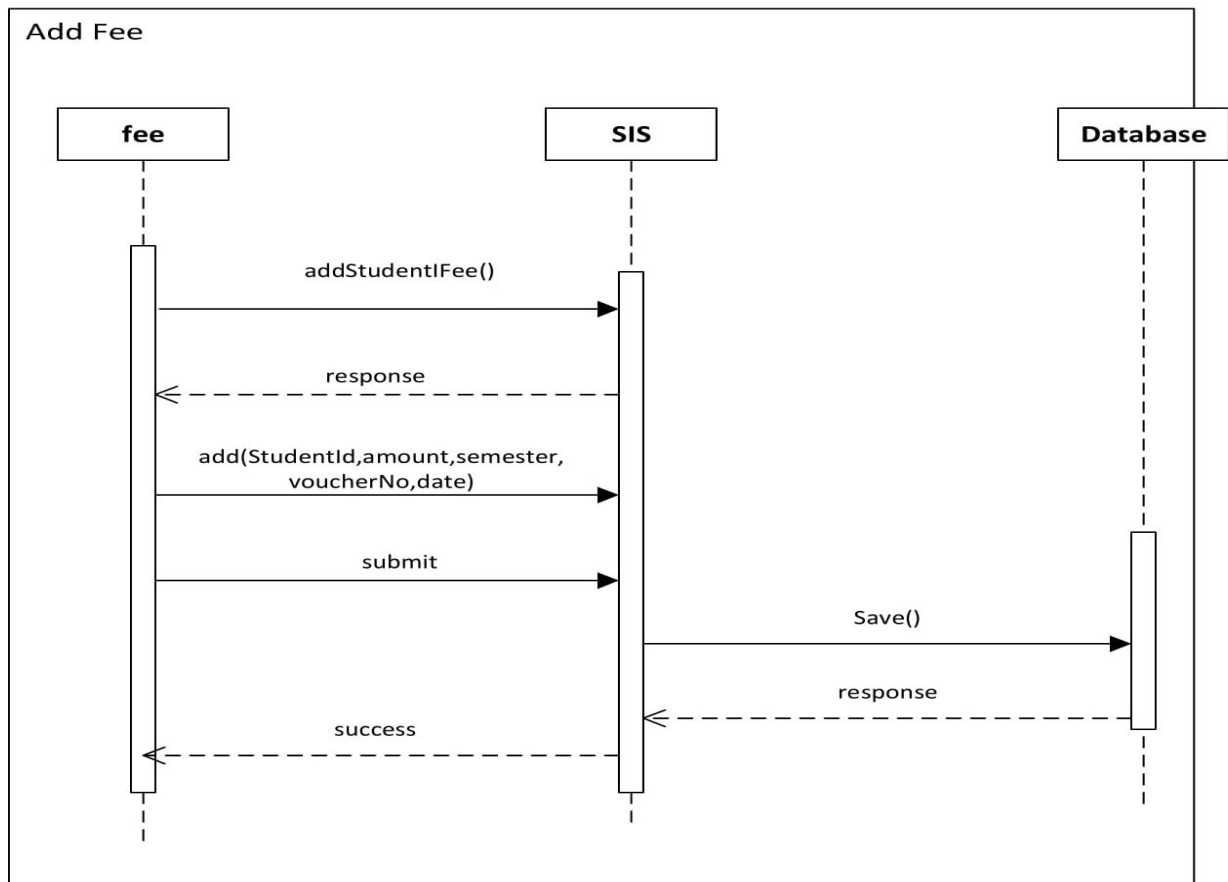


Figure 4.2.10 Student Fee Sequence Diagram

4.3 Use Case Diagram

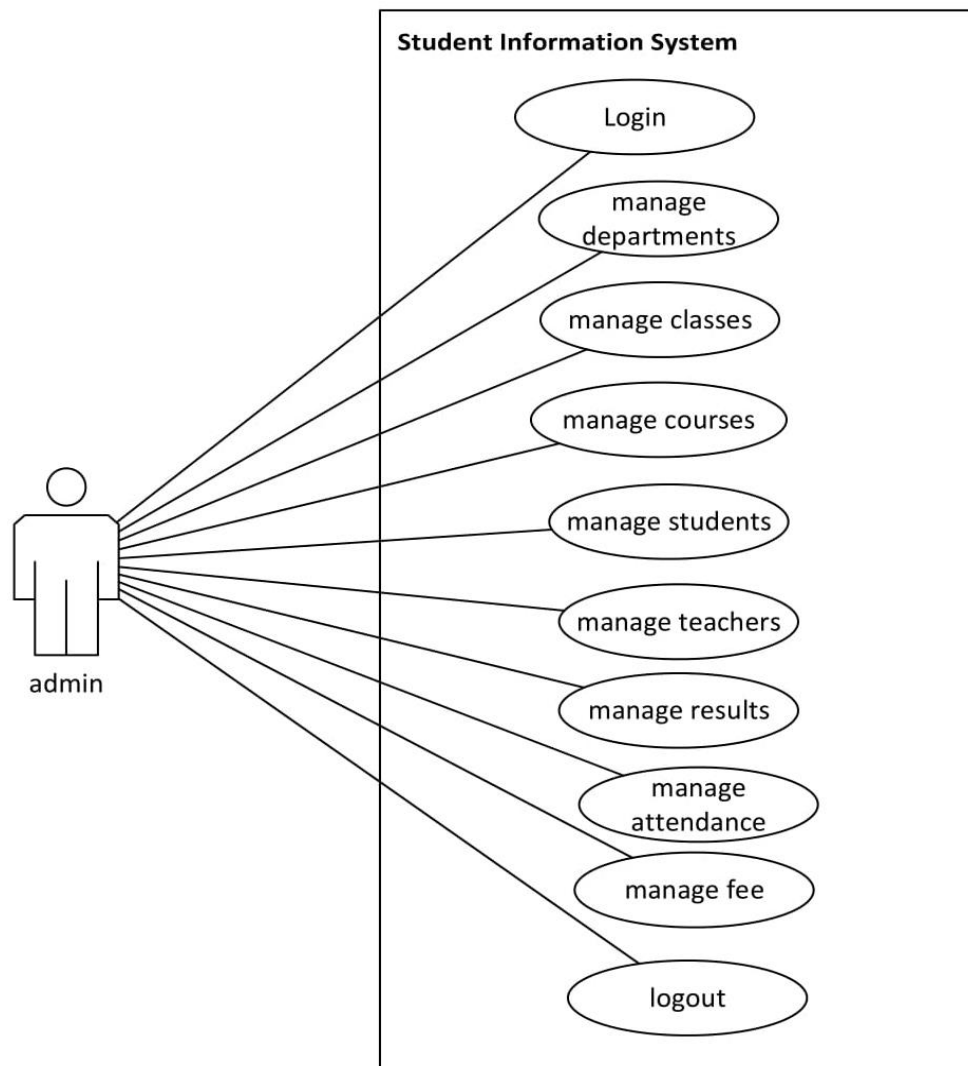
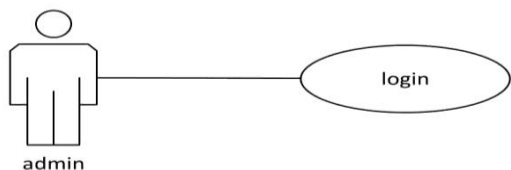


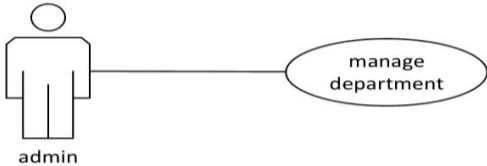
Figure 4.3 Use case Diagram

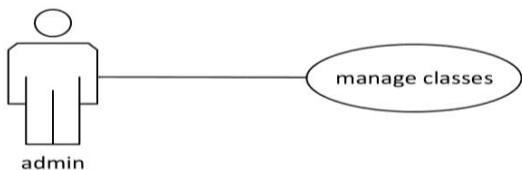
4.3.1 Use Case Description

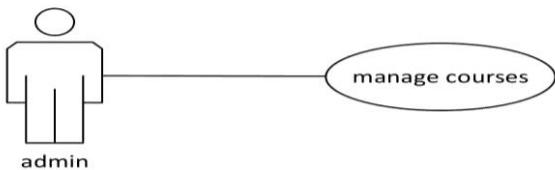
<i>Use Case UC1: Login</i>	
Full Name	Login Admin
Level:	Administration
Primary Actor:	Administration
Secondary Actor	Administration
Description:	This use case describes how an admin logs into his/her SIS account.
Precondition:	Admin/actor must have an active login account
Post condition:	Admin/actor is log in and can use the system
Main Success scenario:	<p>This use case starts when actor wishes to login his/her account.</p> <ol style="list-style-type: none">1. The actor click smart login link.2. The actor types his/her ID and Password.3. The actor hits the login button4. The system validates the actors password and logs him/her into the system
Extension:	There is no extension points associated with this use case.

Use Case Diagram 1:Login



<i>Use Case UC2: Manage Department</i>	
Full Name	Manage Department
Level:	Admin
Primary Actor:	Admin
Secondary Actor	Admin
Description:	The Admin add, view, update and delete Departments in SIS
Precondition:	1. Admin must be log into system
Post condition:	Department id, name inserted into database
Main Success scenario:	<p>This use case starts when admin wishes to add department in the SIS.</p> <ol style="list-style-type: none"> 1. The admin click add department link. 2. The Admin type department name. 3. The Admin hits the save button 4. The system validates the department attributes, if the data meets the requirements it will be inserted into database. 5. The admin can also view, update and delete the departments.
Extension:	System will generate and error and provide some suggestion if the requirement will not fulfill.
<p>Use Case Diagram 2:Manage_Department</p>  <pre> graph LR admin((admin)) --- manage_department([manage department]) </pre>	

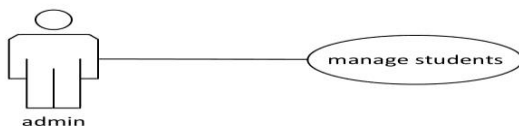
<i>Use Case UC3: Manage Class</i>	
Full Name	Manage Class
Level:	Admin
Primary Actor:	Admin
Secondary Actor	Admin
Description:	The Admin add Class in SIS.
Precondition:	2. Admin must be log into system
Post condition:	Class id, name, no of semesters inserted into database
Main Success scenario:	<p>This use case starts when admin wishes to add class in the SIS.</p> <ol style="list-style-type: none"> 6. The admin click add Class link. 7. The Admin type Class name, id and no. of semesters. 8. The Admin hits the save button 9. The system validates the Class attributes, if the data meets the requirements it will be inserted into database. 10. The admin can also view, update and delete the classes.
Extension:	System will generate and error and provide some suggestion if the requirement will not fulfill.
<p>Use Case Diagram 3:Manage_Class</p>  <pre> graph LR admin[admin] --- manage_classes((manage classes)) </pre>	

<i>Use Case UC4: Manage Course</i>	
Full Name	Manage Course
Level:	Admin
Primary Actor:	Admin
Secondary Actor	Admin
Description:	The Admin add Course in SIS
Precondition:	3. Admin must be log into system
Post condition:	Course id, code and cr. hours inserted into database
Main Success scenario:	<p>This use case starts when admin wishes to add Course in the SIS.</p> <ol style="list-style-type: none"> 11. The admin click add Course link. 12. The Admin type Course id, code and cr. hours. 13. The Admin hits the save button 14. The system validates the course attributes, if the data meets the requirements it will be inserted into database. 15. The admin can also view, update and delete the course.
Extension:	System will generate and error and provide some suggestion if the requirement will not fulfill.
<p>Use Case Diagram 4:Manage_Course</p>  <pre> graph LR admin[admin] --- manage_courses((manage courses)) </pre>	

Use Case UC5: Manage Student Information

Full Name	Manage Student Information
Level:	Admin
Primary Actor:	Admin
Secondary Actor	Admin
Description:	The Admin add Student Information in SIS
Precondition:	4. Admin must be log into system
Post condition:	Student information (name, father name, CNIC, cell no., email, Address, gender, date of birth, registration date, session, department, class) inserted into database.
Main Success scenario:	<p>This use case starts when admin wishes to add Student Information in the SIS.</p> <p>16. The admin click add Student Information link.</p> <p>17. The Admin type Student Information Detail.</p> <p>18. The Admin hits the save button</p> <p>19. The system validates the Student Information attributes, if the data meets the requirements it will be inserted into database.</p>
Extension:	System will generate and error and provide some suggestion if the requirement will not fulfill.

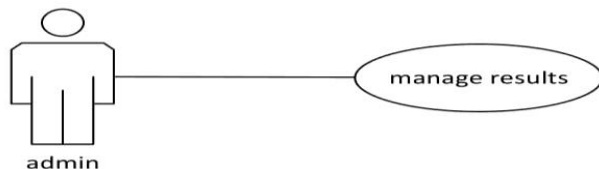
Use Case Diagram 5:Manage_Student



Use Case UC6: Manage Student Result

Full Name	Manage Student Result
Level:	Admin
Primary Actor:	Admin
Secondary Actor	Admin
Description:	The Admin add Student Result in SIS
Precondition:	5. Admin must be log into system
Post condition:	Student result (student id, course id, semester, mid marks, final marks, sessional marks, gpa) inserted into database.
Main Success scenario:	<p>This use case starts when admin wishes to add Student Result in the SIS.</p> <p>20. The admin click add Student Result link.</p> <p>21. The Admin type Student Result Detail.</p> <p>22. The Admin hits the save button</p> <p>23. The system validates the Student Result attributes, if the data meets the requirements it will be inserted into database.</p> <p>24. The admin can also view, update and student result.</p>
Extension:	System will generate and error and provide some suggestion if the requirement will not fulfill.

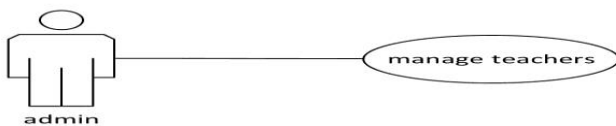
Use Case Diagram 6:Manage_Result

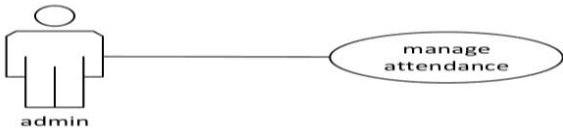


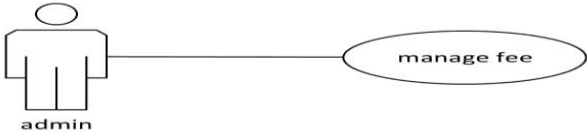
Use Case UC7: Manage Teacher Information

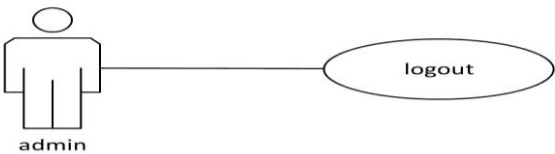
Full Name	Manage Teacher Information
Level:	Admin
Primary Actor:	Admin
Secondary Actor	Admin
Description:	The Admin add Teacher Information in SIS
Precondition:	6. Admin must be log into system
Post condition:	Teacher information (name, CNIC, cell no., email, Address, gender, qualification, designation, type, hiring date) inserted into database.
Main Success scenario:	<p>This use case starts when admin wishes to add Teacher Information in the SIS.</p> <p>25. The admin click add Teacher Information link.</p> <p>26. The Admin type teacher Information Detail.</p> <p>27. The Admin hits the save button</p> <p>28. The system validates the teacher Information attributes, if the data meets the requirements it will be inserted into database.</p>
Extension:	System will generate and error and provide some suggestion if the requirement will not fulfill.

Use Case Diagram 7:Manage_Teacher



<i>Use Case UC8: Manage Attendance</i>	
Full Name	Manage Student Attendance
Level:	Admin
Primary Actor:	Admin
Secondary Actor	Admin
Description:	The Admin add Student Attendance in SIS
Precondition:	7. Admin must be log into system
Post condition:	Student information (student id, course id, course attendance, term, and semester) inserted into database.
Main Success scenario:	<p>This use case starts when admin wishes to add Student Attendance Information in the SIS.</p> <p>29. The admin click add Student Attendance link.</p> <p>30. The Admin type Student Attendance Detail.</p> <p>31. The Admin hits the save button</p> <p>32. The system validates the Student Attendance Information attributes, if the data meets the requirements it will be inserted into database.</p> <p>33. The admin can also view, and update student attendance.</p>
Extension:	System will generate and error and provide some suggestion if the requirement will not fulfill.
<p>Use Case Diagram 8:Manage_Attendance</p>  <pre> graph LR admin[admin] --- manage_attendance((manage attendance)) </pre>	

Use Case UC9: Manage Fee	
Full Name	Manage Student Fee Information
Level:	Admin
Primary Actor:	Admin
Secondary Actor	Admin
Description:	The Admin add Student Fee Information in SIS
Precondition:	8. Admin must be log into system
Post condition:	Student Fee information (student id, semester, amount, voucher no, and paid date) inserted into database.
Main Success scenario:	<p>This use case starts when admin wishes to add Student Fee Information in the SIS.</p> <p>34. The admin click add Student Fee Information link.</p> <p>35. The Admin type Student Fee Detail.</p> <p>36. The Admin hits the save button</p> <p>37. The system validates the Student Fee Information attributes, if the data meets the requirements it will be inserted into database.</p> <p>38. The admin can also view, update and delete the teacher record.</p>
Extension:	System will generate and error and provide some suggestion if the requirement will not fulfill.
<p>Use Case Diagram 9:Manage_Fee</p>  <pre> graph LR admin[admin] --- manage_fee((manage fee)) </pre>	

<i>Use Case UC10: Logout</i>	
Full Name	Admin Logout
Level:	Admin
Primary Actor:	Admin
Secondary Actor	Admin
Description:	The Admin Logout.
Precondition:	9. Admin must be log into system
Main Success scenario:	<p>This use case starts when admin wishes to add Student Information in the SIS.</p> <p>39. The admin click add Student Logout link.</p> <p>40. The system logout the admin.</p>
Extension:	System will generate and error and provide some suggestion if the requirement will not fulfill.
<p>Use Case Diagram 10:LogOut</p>  <pre> graph LR admin[admin] --- logout((logout)) </pre>	

4.4 Class Diagram

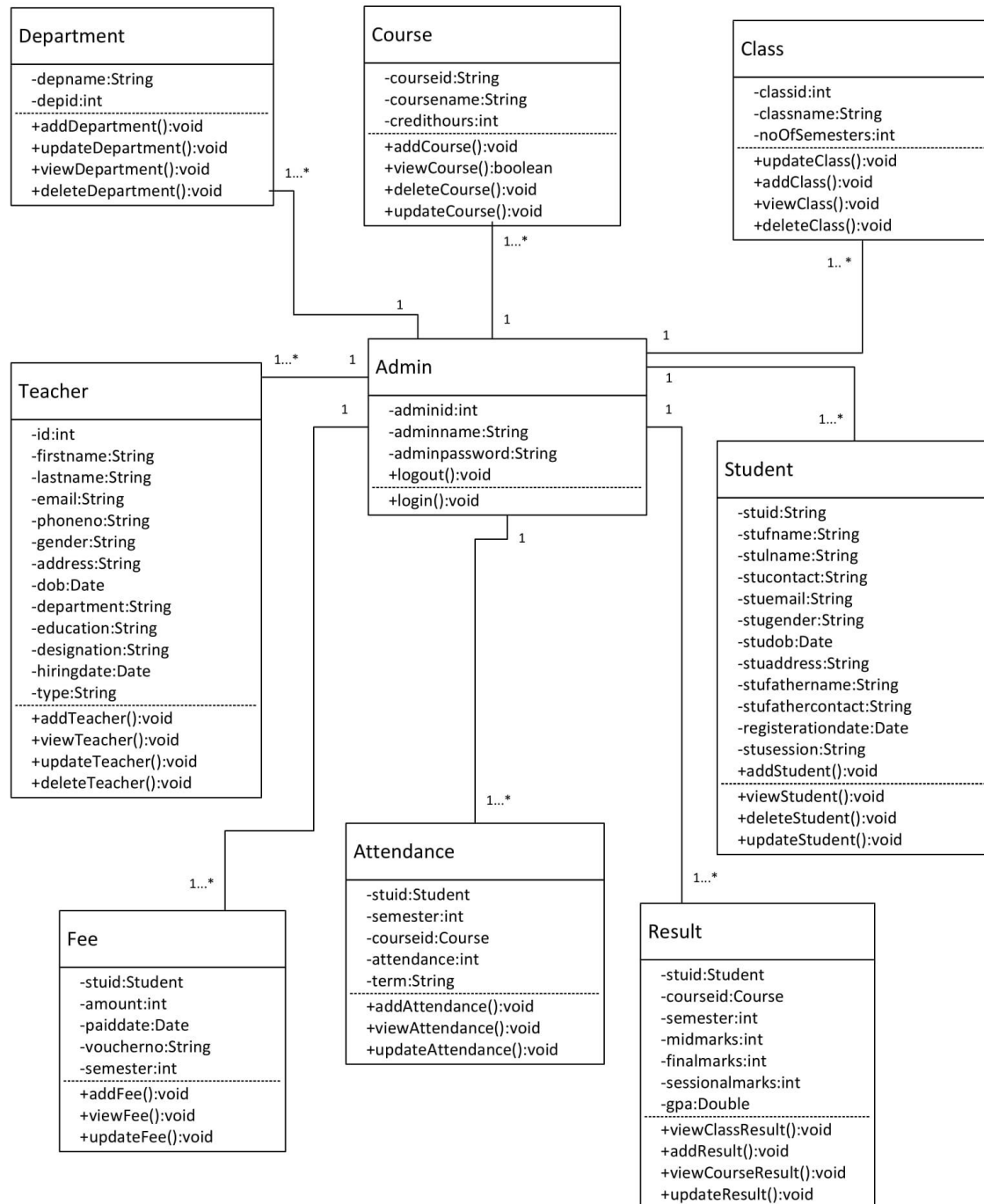


Figure 4.4 Student Information System Class Diagram

4.5 ERD Diagram

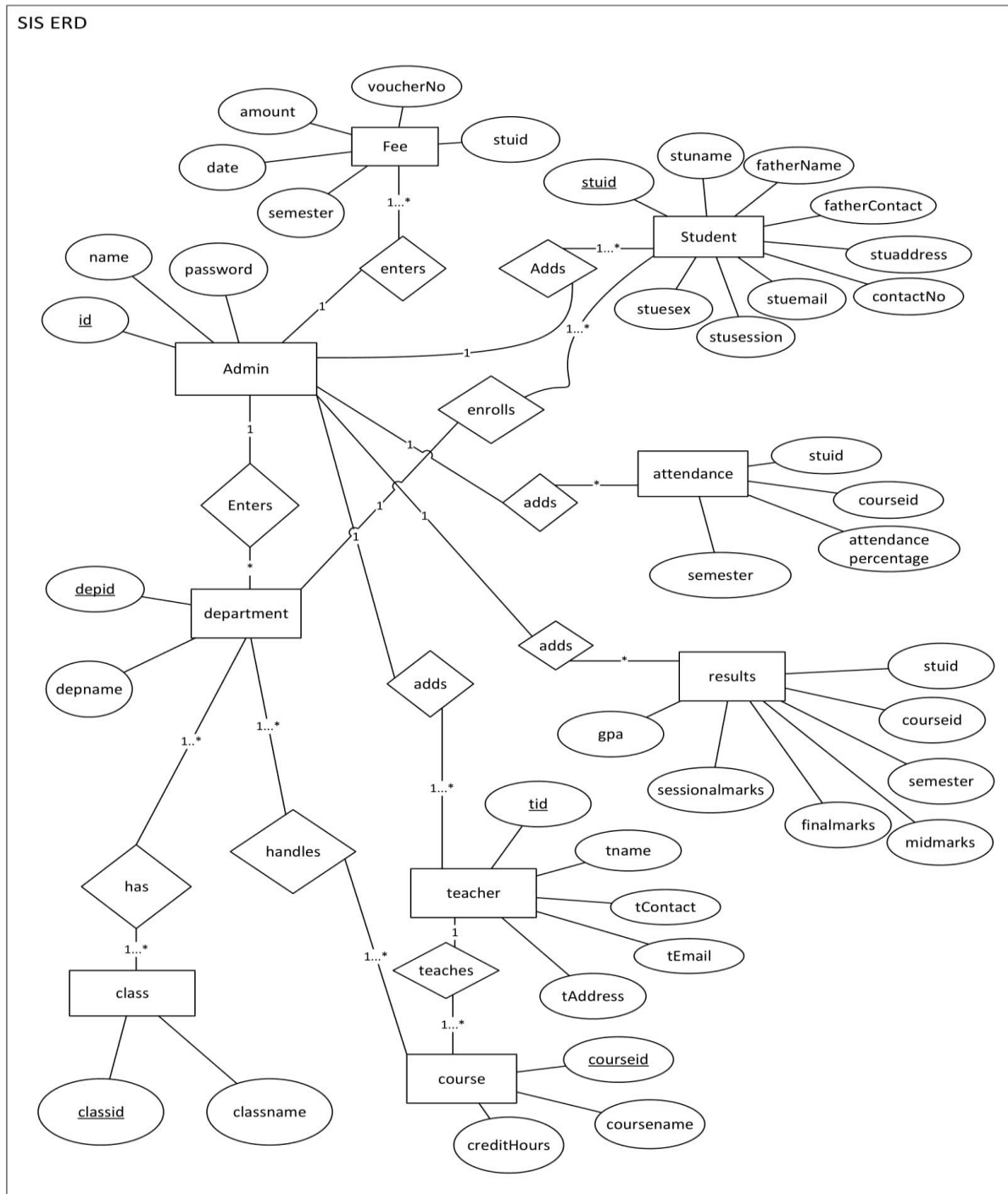


Figure 4.5 Student Information System Entity Relation Diagram

Chapter 5

Implementation

5.1 Introduction

In this chapter we will be discussing the fact that how Student Information Management System is implemented and which were the tools and technologies, that made the conceived idea a real artifact.

5.2 Tools and Technology Used

5.2.1 Tools

5.2.1.1 Eclipse

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C++, C#, JavaScript, Perl, PHP, Prolog, Python, R, Ruby, Rust, Scala, and Scheme. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

Eclipse software development kit (SDK) is free and open-source software, released under the terms of the Eclipse Public License, although it is incompatible with the GNU General Public License.

5.2.1.2 DBMS (SQL Server)

Microsoft SQL Server SQL Server is a relational database management system (RDBMS) from Microsoft that's designed for the enterprise environment. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications which may run either on the same computer or on another computer across a network (including the Internet).

5.2.2 Technology

5.2.2.1 Language: Java

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to "bytecode" that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The language derives much of its original features from SmallTalk, with a syntax similar to C and C++, but it has fewer low-level facilities than either of them. As of 2018, Java was according to Github one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers.

Java was originally developed by a Canadian James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform.

5.2.3 Planning & Scheduling

Gantt chart

A Gantt chart is a scheduling technique that is used commonly in project management. It is one of the useful and common ways of representation of activities displayed in contrast to time. On the vertical side of the chart activities list is displayed and on the top a time scale which is appropriate. A bar represents each activity's length and also shows the start date, end date and duration of the activity.

This shows following information.

- "What the different activities are?"
- "When an activity starts and ends?"
- "How much long each activity is scheduled to finish?"
- "The start and finish date of the whole project."

Key Milestones of the Project with dates

Table 5-1: Project Plan & Key Milestones

Key Milestones of the Project with dates			
S. No	Elapsed time since start of the project	Milestone	Deliverable
1.	October 21,2018 - December 15,2018	Analysis & Requirements	Complete Research Proposal
2.	December 16,2018 - January 2 ,2019	Planning, scheduling and purchase of hardware	Report
3.	January 3,2019 – January 20,2019	Modeling & design	Report
4.	January 21,2019 - April 18,2019	Coding & testing	Software System
5.	April 20, 2019	Deployment	Hardware & Software System

Chapter 6

Database Design

6.1 Database Design

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model.

Database design involves classifying data and identifying interrelationships. This theoretical representation of the data is called ontology. The ontology is the theory behind the database's design.

6.2 Database

MYSQL is an application for storing information inside a “table” structure; let’s examine the reasons why you would use a Database rather than a spreadsheet or some other program for data storage. Imagine you’re creating an application for storing sales transactions. We’ll start by saving just a few columns of information such as the Student Id, name, address, phone, email, gender. One of the first storage options to consider is saving this information in a large text file. There are benefits to text file saves such as quick write times. The problem with text files is during a read, if the text file is large, it can take quite a bite of time to open and scan the contents of the file looking for what we want. Also, if we wanted to see all the record of a specific student, the entire text file would have to be read, and every line occurrence of the student name would need to be saved in some temporary place until we had them all. If we saved to a spreadsheet instead of a text file, we would have a Sort feature built in. So we may be able to find all the sales to a specific student quicker, but again, if the file was large, opening the spreadsheet could take a great deal of time.

Why Not Use a Database?

There are some problems with using a database. First, time must be taken to learn the new system. A database is not as intuitive as a spreadsheet. In addition, if there is only a small amount of data that doesn’t need to be changed over time, it’s probably simpler to save it in a file. Unfortunately, most business problems are neither simple nor small, so a database is usually the best tool for the job.

6.2.1 Advantages of Database

- An organized and comprehensiveness of recording the result of the firms activities.
- A receiver of data to be used in meeting the information requirement of the MIS users.
- Reduced data redundancy.
- Reduced updating errors and increased consistency.
- Greater data integrity and independence from applications programs.
- Improved data access to users through use of host and query languages.
- Improved data security.
- Reduced data entry, storage, and retrieval costs.
- Facilitated development of new applications program.
- Standard can be enforced: Standardized stored data format is particularly desirable as an old data to interchange or migration (change) between the systems.
- Conflicting requirement can be handled.

6.2.2 Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition

- **DEFAULT** - Sets a default value for a column when no value is specified
- **INDEX** - Used to create and retrieve data from the database very quickly

6.2.3 Entity

An entity is any object in the system that we want to model and store information about. Entities are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events which have relevance to the database. Entities are represented by rectangles (either with round or square corners):

Examples:

Customer, Staff, and Account.

6.2.4 Attributes

An attribute is an item of information which is stored about an entity. For example, the entity 'Student' could have attributes such as student id, surname, forename, date of birth, telephone number, etc. An attribute can only appear in one entity, unless it is the key attribute in another entity. In a traditional filing system an attribute equates to a **field** in a record.

6.2.5 Keys

A key is a data item that allows us to uniquely identify individual occurrences or an entity type. You can sort and quickly retrieve information from a database by choosing one or more fields (i.e. attributes) to act as *keys*. For instance, in a Student table you could use a combination of the last name and first name fields (or perhaps last name, first name and birth dates to ensure you identify each student uniquely) as a key field.

There are several types of key field:

- Primary Key
 - Secondary Key
 - Foreign key
 - Composite key
- a. **Primary Key**

A **primary key** consists of one or more attributes that distinguishes a specific record from any other. For each record in the table the primary key acts like a driver's license number or a national insurance number, only one number exists for each person.

For example, your student number

A primary key is **mandatory**. That is, each entity occurrence must have a value for its primary key.

b. Secondary Key

An entity may have one or more choices for the primary key. Collectively these are known as candidate keys. One is selected as the primary key. Those not selected are known as **secondary keys**.

For example, a student has student roll no, CNIC number and an email address. If the student roll no is chosen as the primary key then the CNIC number and email address are secondary keys. However, it is important to note that if any student does not have a CNIC number or email address (i.e. the attribute is not mandatory) then it cannot be chosen as a primary key.

c. Foreign Key

A **foreign key** is one or more attribute in one entity, which enables a link (or relationship) to another entity. That is, a foreign key in one entity links to a primary key in another entity. However, if the business rules permit, a foreign key may be optional.

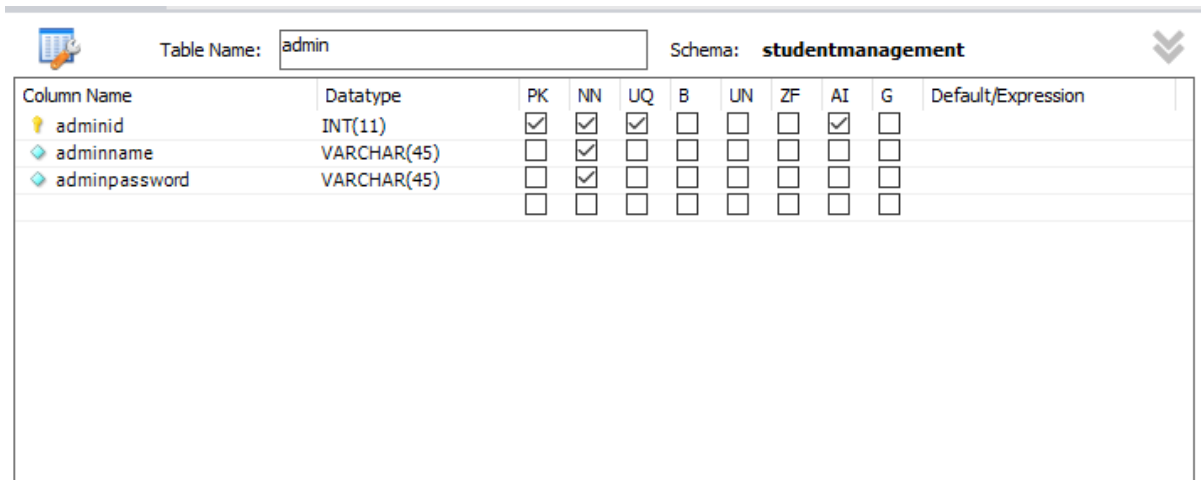
For example, a student enrolls in a department. The department number column in the student entity is a foreign key, which links to the department entity.

d. Composite Key

A **composite key** consists of more than one attribute to uniquely identify an entity occurrence.

6.3 Database tables

6.3.1 Admin Table

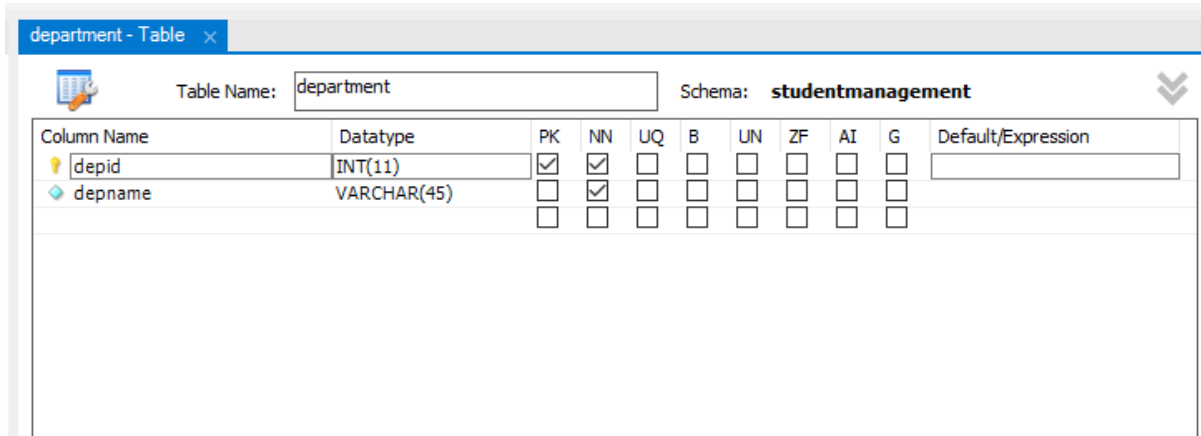


The screenshot shows the configuration for the 'admin' table in the 'studentmanagement' schema. The table has three columns: 'adminid' (INT(11)), 'adminname' (VARCHAR(45)), and 'adminpassword' (VARCHAR(45)). The 'adminid' column is marked as the primary key (PK), not null (NN), and unique (UQ). The 'adminname' and 'adminpassword' columns are marked as not null (NN). The 'adminid' column is also marked as indexed (AI) and has a default value (G). The 'adminname' and 'adminpassword' columns are also marked as indexed (AI) and have default values (G).

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
adminid	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
adminname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
adminpassword	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 6.3.1 Admin Database Table

6.3.2 Department Table



The screenshot shows the configuration for the 'department' table in the 'studentmanagement' schema. The table has two columns: 'depid' (INT(11)) and 'depname' (VARCHAR(45)). The 'depid' column is marked as the primary key (PK), not null (NN), and unique (UQ). The 'depname' column is marked as not null (NN). The 'depid' column is also marked as indexed (AI) and has a default value (G). The 'depname' column is also marked as indexed (AI) and has a default value (G).

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
depid	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
depname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 6.3.2 Department Database Table

6.3.3 Class Table

class - Table x

Table Name: Schema: **studentmanagement**





Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 classid	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
 classname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 depid	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 semesters	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 6.3.3 Class Database Table

6.3.4 Course Table

course - Table x

Table Name: Schema: **studentmanagement**




Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 courseid	VARCHAR(30)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
 coursename	VARCHAR(80)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 credithours	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 6.3.4 Course Database Table

6.3.5 Course Mapping Table

coursemapping - Table										
Table Name: <input type="text" value="coursemapping"/>		Schema: studentmanagement								
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
courseid	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
classid	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
depid	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
semester	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
teachername	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
session	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 6.3.5 Course Mapping Database Table

6.3.6 Student Table

student - Table										
Table Name: <input type="text" value="student"/>		Schema: studentmanagement								
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
stuid	VARCHAR(30)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stufname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stulname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stufathname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stuemail	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stuaddress	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stugender	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
sturegistration	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
depid	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
classid	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stusession	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stuPhone	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stuzip	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
studob	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stufatherphone	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stufatherocuppion	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 6.3.6 Student Database Table

6.3.7 Teacher Table














teacher - Table											
Table Name: <input type="text" value="teacher"/>		Schema: studentmanagement									
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression	
 teacherid	VARCHAR(45)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teachername	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teacherid	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teacheremail	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teacherphone	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teacheraddress	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teacherdob	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teachersex	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teachereducation	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teacherdesignation	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teacherdepartment	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teacherhiredate	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 teachername	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Figure 6.3.7 Teacher Database Table

6.3.8 Student Attendance Table







attendance - Table											
Table Name: <input type="text" value="attendance"/>		Schema: studentmanagement									
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression	
 stuid	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 courseid	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 semester	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 session	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 term	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
 attendance	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Figure 6.3.8 Student Attendance Database Table

6.3.9 Student Result Table

result - Table x										
Table Name: <input type="text" value="result"/>		Schema: studentmanagement								
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
◆ stuid	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◆ courseid	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◆ midmarks	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◆ semester	INT(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◆ finalmarks	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◆ sessionalmarks	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◆ gpa	DOUBLE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

Figure 6.3.9 Student Result Database Table

6.3.10 Student Fee Table

fee - Table x										
Table Name: <input type="text" value="fee"/>		Schema: studentmanagement								
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
◆ stuid	VARCHAR(40)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
◆ semester	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◆ challanno	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◆ amount	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◆ paiddate	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 6.3.10 Student Fee Database Table

Chapter 7

User Interface

7.1 Introduction

A user interface, also called a "UI" or simply an "interface," is the means in which a person controls a software application or hardware device. A good user interface provides a "user friendly" experience, allowing the user to interact with the software or hardware in a natural and intuitive way.

Nearly all software programs have a graphical user interface, or GUI. This means the program includes graphical controls, which the user can select using a mouse or keyboard. A typical GUI of a software program includes a menu bar, toolbar, windows, buttons, and other controls. The Macintosh and Windows operating systems have different user interfaces, but they share many of the same elements, such as a desktop, windows, icons, etc. These common elements make it possible for people to use either operating system without having to completely relearn the interface. Similarly, programs like word processors and Web browsers all have rather similar interfaces, providing a consistent user experience across multiple programs.

Most hardware devices also include a user interface, though it is typically not as complex as a software interface. A common example of a hardware device with a user interface is a remote control. A typical TV remote has a numeric keypad, volume and channel buttons, mute and power buttons, an input selector, and other buttons that perform various functions. This set of buttons and the way they are laid out on the controller makes up the user interface. Other devices, such as digital cameras, audio mixing consoles, and stereo systems also have a user interface.

While user interfaces can be designed for either hardware or software, most are a combination of both. For example, to control a software program, you typically need to use a keyboard and mouse, which each have their own user interface. Likewise, to control a digital camera, you may need to navigate through the on-screen menus, which is a software interface. Regardless of the application, the goal of a good user interface is to be user-friendly. After all, we all know how frustrating it can be to use a device that doesn't work the way we want it to.

7.2 User Interface Front End

7.2.1 Login

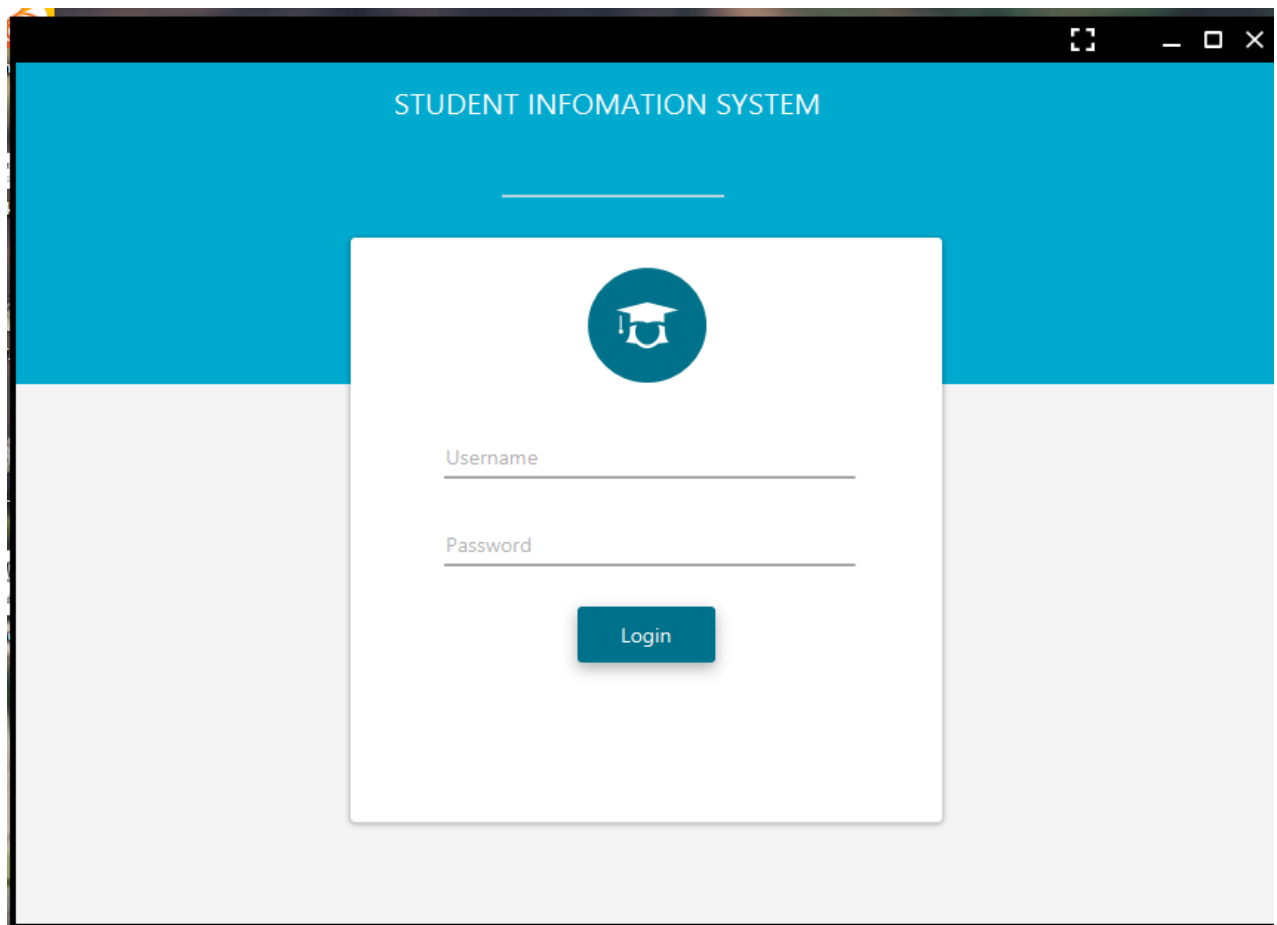


Figure 7.2.1 Login Page

Description:

This is the Log in page of “Student Information System”. Here Admin can log by entering his name and password. If both match with the database value Admin will be logged in otherwise not.

7.2.2 Dashboard Page

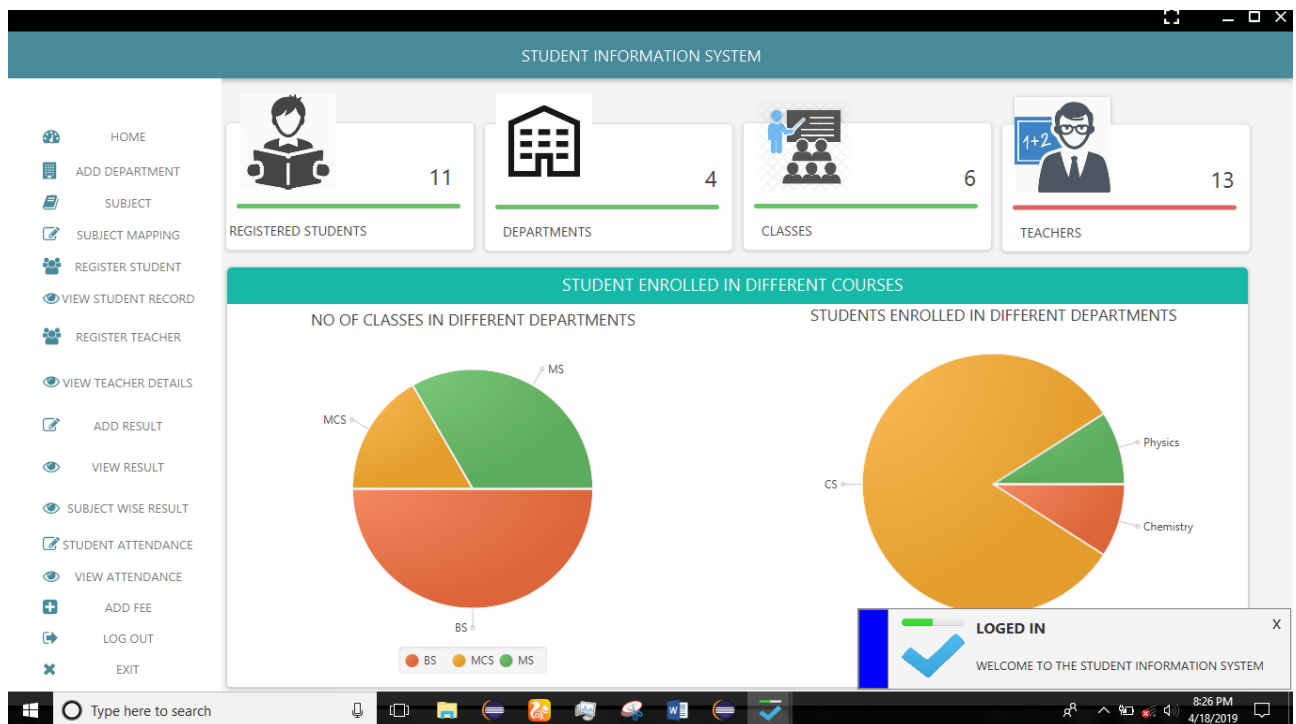


Figure 7.2.2 Dashboard

7.2.3 Department Page

Admin adds department using “add department” page.

STUDENT INFORMATION SYSTEM

DEPARTMENT

DEPARTMENT ID:

DEPARTMENT NAME:

CLASS MAPPING

CLASS ID:

CLASS NAME:

DEPARTMENT:

DEPARTMENT ID:

SEMESTER NO:

CLASS DETAILS

D.ID	DEPARTMENT NAME	C.ID	CLASS NAME	DEPARTMENT	SEMESTERS
8021	CS	101	BS	CS	8
8022	Physics	102	MCS	CS	4
8023	Chemistry	103	MS	CS	4
8024	Business	301	BS	Chemistry	8

Figure 7.2.3 Department Page

7.2.4 Course Page

Admin adds course using “add course” page.

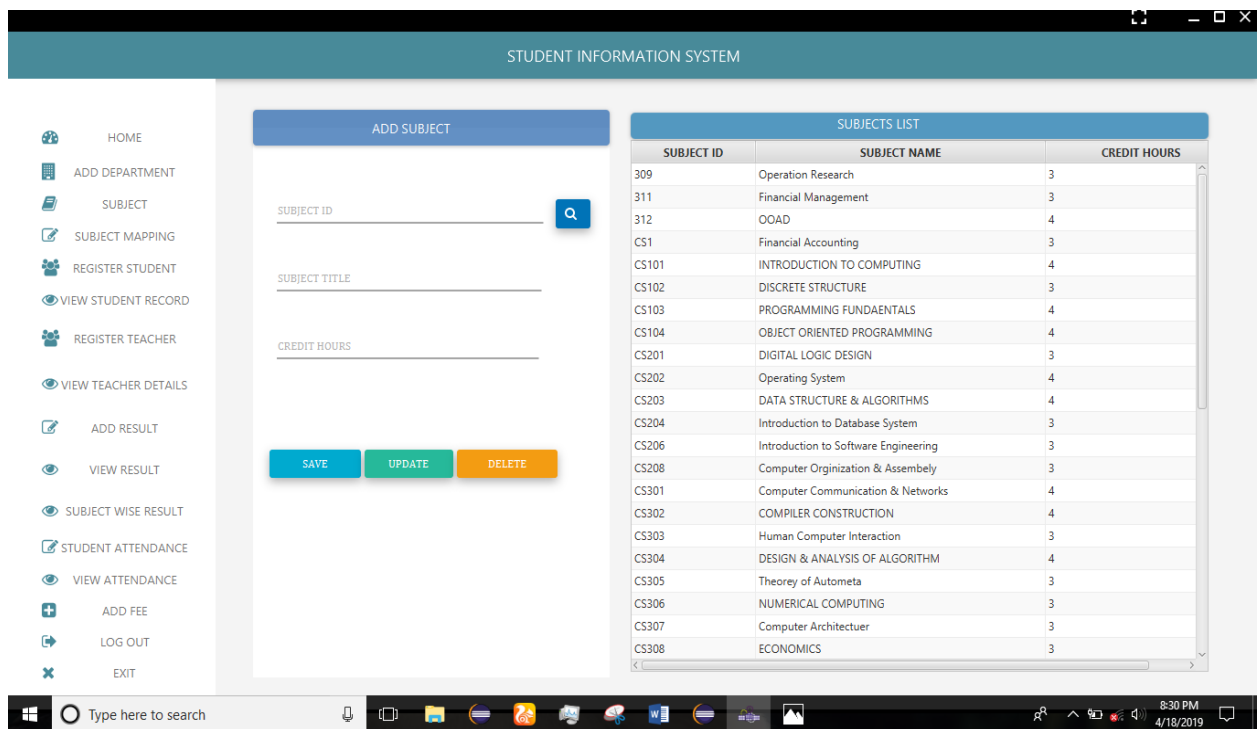


Figure 7.2.4 Course Page

7.2.5 Course Mapping Page

Admin assign courses to different classes using “course mapping” page.

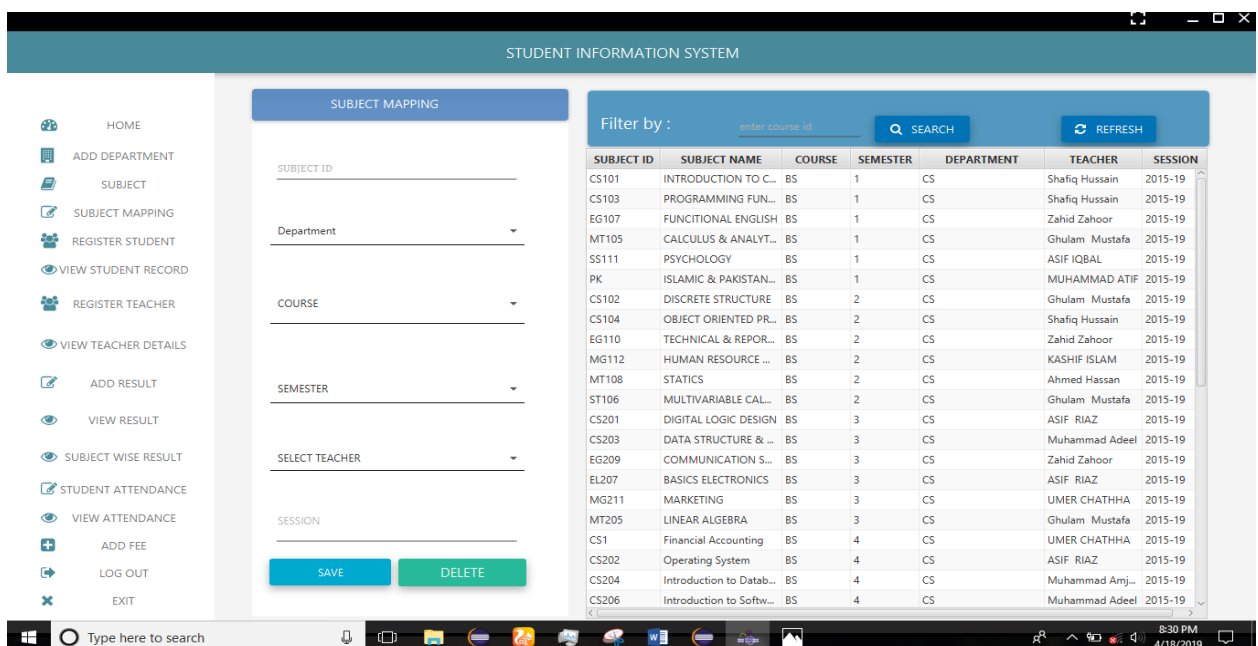


Figure 7.2.5 Course Mapping Page

7.2.6 Student Register Page

Admin adds students' details in this page.

STUDENT INFORMATION SYSTEM

HOME
ADD DEPARTMENT
SUBJECT
SUBJECT MAPPING
REGISTER STUDENT
VIEW STUDENT RECORD
REGISTER TEACHER
VIEW TEACHER DETAILS
ADD RESULT
VIEW RESULT
SUBJECT WISE RESULT
STUDENT ATTENDANCE
VIEW ATTENDANCE
ADD FEE
LOG OUT
EXIT

Personal Details

Student ID

First Name

Last Name

CNIC

Email

Phone

☐ Male ☐ Female

Family Details

Date of Birth

Father's Name

Father's Phone No

Occupation

Address Details

Home Address

Zip Code

School Details

Department

Class

Session

SAVE UPDATE DELETE

Figure 7.2.6 Student Register Page

7.2.7 Student View page

Registered Students' details can be seen in this page.

STUDENT INFORMATION SYSTEM

Filter by : CS BS 2015-19 SEARCH ALL

ENTER STUDENT ID Search

ID	NAME	EMAIL	CONTACT	CNIC	SEX	DOB	FATHER NAME	COOCCUPTION	CONTACT	REGISTRATION...
BCS-15-...	Zaineb Azeem	zaineb@gmail.com	030012345678	36502123456789	FEM...	1996-04-17	Azeem	Engineer	03002123...	2015-09-08
bcs-15-02	TANVEER HUSSAIN	TANVEER@GMAIL.COM	0300123456...	365021234567	MALE	1995-04-19	HUSSAIN	Teacher	03002154...	2015-09-01
BCS-15-...	MUHAMMAD SHAHID	M.SHAHID@GMAIL.COM	03013560749	365023123456	MALE	1999-04-24	KHADAM HUSS...	Farmer	03013654...	2015-09-01
BCS-15-...	AHMED FARAZ	FARAZ@GMAIL.COM	03001245678	36502123456	MALE	1997-04-16	TALIB HUUSAIN	Engineer	03012345...	2015-09-08
BCS-15-...	FAIZ FAREED	FAIZ@GMAIL.COM	030123654789	36502123456	MALE	1994-04-06	FAREED	Teacher	03021554...	2019-04-15
BCS-15-...	FAIQA YOUSAF	FAIQA@GMAIL.COM	0302215478	36520125478	FEM...	1995-04-12	YOUSAF	Teacher	03002154...	2015-09-15
BCS-15-...	AZKA LIAQAT	AZKA@GMAIL.COM	0322154785	3652312456	FEM...	1996-04-03	LIAQAT	Engineer	03254785...	2015-09-01

Figure 7.2.7 Student View Page

7.2.8 Teacher Register Page

Admin add teachers' detail using this page.

STUDENT INFORMATION SYSTEM

PERSONAL DETAILS

TEACHER ID

FIRST NAME

LAST NAME

CNIC

EMAIL

PHONE

ADDRESS

DATE OF BIRTH

☐ Male ☐ Female

ACADEMIC DETAILS

EDUCATION

DEPARTMENT

RESEARCH

DESIGNATION

TYPE

HIRE DATE

Figure 7.2.8 Teacher Register Page

7.2.9 Teacher View Page

Admin can view the teachers' detail using teacher view page.

STUDENT INFORMATION SYSTEM

Filter by:

ID	NAME	EMAIL	CONTACT	CNIC	SEX	DOB	EDUCATION	DEPARTMENT	DESIGNATION	TYPE
101	Sir Haroon Ta...	sirharoon@gmail.com	03001234567...	36502222222	MALE	1995-10-...	PhD	Computer Science	Associate Professor	Permanent Faculty
102	Muhammad A...	link2ramjad@gmail.com	0300981111111	3650211111111	MALE	1993-04-...	MS	Computer Science	Lecturer	Visiting Faculty
103	Shafiq Hussain	shafiqhussain@gmail.com	030012345678	36502111111	MALE	1985-04-...	PhD	Computer Science	Associate Professor	Permanent Faculty
104	Ahmed Hassan	ahmed@gmail.com	03001234567...	3650212345678	MALE	1977-04-...	PhD	Computer Science	Assistant Professor	Visiting Faculty
105	Muhammad A...	adeel@gmail.com	030012345678	3650212345678	MALE	1989-04-...	PhD	Computer Science	Lecturer	Visiting Faculty
106	Muhammad T...	tahir@gmail.com	03001234567...	36502111123	MALE	1960-04-...	MS	Computer Science	Lecturer	Visiting Faculty
107	Ghulam Must...	ghulammustafa@gmail.com	03001234567...	365021234567	MALE	1986-04-...	MS	Computer Science	Lecturer	Visiting Faculty
108	Zahid Zahoor	zahid@gmail.com	0302154678	36502123456789	MALE	1980-04-...	PhD	Computer Science	Assistant Professor	Visiting Faculty
109	ASIF IQBAL	ASIF@GMAIL.COM	031234567821	36502111111	MALE	1975-04-...	PhD	Computer Science	Assistant Professor	Visiting Faculty
110	MUHAMMAD ...	ATIF@GMAIL.COM	03001234567	365021234564	MALE	1987-04-...	MS	Computer Science	Lecturer	Visiting Faculty
111	KASHIF ISLAM	KASHIF@GMAIL.COM	030012345678	365021232213	MALE	1979-04-...	PhD	Computer Science	Lecturer	Visiting Faculty
112	ASIF RIAZ	ASIF@GMAIL.COM	03001234567...	3650212345678	MALE	1988-04-...	PhD	Computer Science	Assistant Professor	Visiting Faculty
113	UMER CHATH...	UMER@GMAIL.COM	030123456789	650213345678	MALE	1951-04-...	PhD	Computer Science	Assistant Professor	Visiting Faculty

Figure 7.2.9 Teacher View page

7.2.10 Add Result

Admin adds students' subject wise result using add student result page.

Figure 7.2.10 Add Result Page

7.2.11 View Result

Admin can view the students' GPA using view result page.

ID	NAME	PK	CS101	CS103	EG107	MT105	SS111
BCS-15-01	Zaineb Azeem	4.0	4.0	4.0	3.4	3.0	4.0
BCS-15-02	TANVEER HUSSAIN	3.8	3.0	3.0	4.0	3.0	3.0
BCS-15-03	MUHAMMAD SHAHID	4.0	3.5	3.7	3.8	2.5	4.0
BCS-15-04	AHMED FARAZ	3.3	3.0	2.5	4.0	2.8	4.0
BCS-15-05	FAIZ FAREED	4.0	3.5	4.0	4.0	4.0	4.0

Figure 7.2.11 View Result

Admin adds students' subject wise attendance using add attendance page.

Admin adds students' subject wise attendance using add attendance page.

Figure 7.2.12 Attendance Page

Admin can view students' attendance details using view attendance page.

Admin can view students' attendance details using view attendance page.

Figure 7.2.13 Student Attendance View Page

7.2.14 Fee Module

Admin add students' fee record using add fee page.

[illegible]

Figure 7.2.14 Fee Page

Chapter 8

SYSTEM TESTING

In this chapter, we will discuss the testing phase of developed application “**Student Information System**” in different manner to know that how much efficient and effective application is.

8.1 Introduction

A process of performing as application or program with the intention of finding errors and whether the application is fulfilling user needs. It can also be defined as the ability of a program in meeting the required or desired results.

In many methodologies of software engineering, a separate phase is called phase of testing which is performed after the completion of the implementation. There is a benefit in using this approach that it is hard to see one's own mistakes, and a fresh eye can find observable errors much faster than the person who has read the material many times.

8.2 Testing Plan

A process of performing as application or program with the intention of finding errors and whether the application is fulfilling user needs.

8.2.1 Unit Testing

The software units in an application are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches to the type and size supported by Java. The various controls are tested to ensure that each performs its action as required.

Commonly used method is White-Box Testing method. Every time a component of the program is changed, it can be run for testing that is the biggest and famous benefit of this testing phase. Issues that are arise during this phase, allowing to be resolved as quickly as possible. Unit testing is familiar by software developers. It allows them to test their application units before move them to testers for formal testing.

8.2.2 System Testing

To test the complete application as a whole, system testing has been used. It is beneficial to check whether the application meets its requirements and fulfill Quality Standards.

8.2.3 Integration Testing

Integration testing allows the software developers to integrate all of the components/ units of the application within a program and then test them in a group. Basically, this testing level is used to catch the defects in the user interface between the functions/ modules. It is useful to determine how logically and efficiently all the units/ components are running together.

Here the streaming module and encoding module options are integrated and tested. This testing provides the assurance that the application is well integrated functional unit with smooth transition of data.

8.2.4 User Acceptance Testing

User acceptance of an application is the key factor for the success of any application. The application under consideration is tested for user acceptance by constantly keeping in touch with the application users at time of developing and making changes whenever required.

8.3 Test Cases

Table 8-1: Testing Cases

Test Cases	Objectives
1	To make sure that user can easily understand and can use the application
2	Make sure that user can easily login
3	Make sure that administrator can easily update, delete, save and view department.
4	Make sure that admin can save, view, update and delete class.
5	Make sure that admin can save, view, update and delete course.
6	Make sure that admin can easily assign view, and update courses to different classes.
7	Make sure that admin can save, view, update and delete Student.
8	Make sure that admin can save, view, update and delete Teacher.
9	Make sure that admin can save, view, and update student attendance.
10	Make sure that admin can save, view, and update student result.
11	Make sure that admin can save, view, and update student fee record.
12	Make sure that the application run at cross-platforms successfully.

8.4 Testing Results

Table 8-2: Testing Result

CRITERIA	Test Status	REMARKS
To make sure that user can easily understand and can use the application	Test successful	None
Make sure that user can easily login	Test successful	None
Make sure that administrator can easily update, delete, save and view department.	Test successful	None
Make sure that admin can save, view, update and delete class.	Test successful	None
Make sure that admin can save, view, update and delete course.	Test successful	None
Make sure that admin can easily assign view, and update courses to different classes.	Test successful	None
Make sure that admin can save, view, update and delete Student.	Test successful	None
Make sure that admin can save, view, update and delete Teacher.	Test successful	None
Make sure that admin can save, view, and update student attendance.	Test successful	None
Make sure that admin can save, view, and update student result.	Test successful	None
Make sure that admin can save, view, and update student fee record.	Test successful	None
Make sure that the application run at cross-platforms successfully.	Test successful	None

Chapter 9

Conclusion & Future Work

In this chapter, we will discuss the results and discussions of this system “**Student Information System**” with conclude remarks and will also discuss related future work of this application.

9.1 Conclusion

“Student Information System” is developed to provide assistance to clerical staff of the “University of Sahiwal”. This application provide ease to clerical staff in managing students’ records.

This application provide such environment that is more efficient and user friendly. It will improve the performance of clerical staff and decrease their efforts in managing the manual records.

9.2 Future Work

In next my first preference is to enhance this application by providing more new features that are as follows:

- Result cards of the student can be printed
- The timetable of every class can be printed
- The students result can be inserted after each term
- Teacher attendance and pay details can be added
- The performance of every student can be monitored and reports can be given to parents.

REFERENCES:

1. UML Revision Task Force. OMG Unified Modelling Language Specification, Version 1.4 (final draft). February 2001
2. https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm
3. <https://www.javatpoint.com/javafx-tutorial>
4. <https://www.tutorialspoint.com/java/>
5. <https://www.w3schools.com/java/>
6. <https://www.w3schools.com/sql/default.asp>
7. https://www.youtube.com/playlist?list=PLhs1urmduZ29LNYi_MaoU60JemQ6Aei6A