Table of Contents

[Chapter 1 41](#_Toc6510947)

[Introduction 41](#_Toc6510948)

[1.1. Project Overview 41](#_Toc6510949)

[1.2. Background study 41](#_Toc6510950)

[1.2.1. Historical perspective 42](#_Toc6510951)

[1.3. Scope 43](#_Toc6510952)

[1.3.1. Modules 43](#_Toc6510953)

[1.4. Features of Proposed System 45](#_Toc6510954)

[Chapter 2 47](#_Toc6510955)

[Requirement Analysis 47](#_Toc6510956)

[2.1. Software Requirement Specification 47](#_Toc6510957)

[2.1.1. Introduction 47](#_Toc6510958)

[2.1.2. Purpose 47](#_Toc6510959)

[2.1.3. Proposed System 48](#_Toc6510960)

[2.2. Specification Requirements 50](#_Toc6510961)

[2.2.1. Functional Requirements 50](#_Toc6510962)

[2.2.2. Non-Functional Requirements 51](#_Toc6510963)

[2.2.3. Optional Requirements 51](#_Toc6510964)

[2.3. Project Objective 51](#_Toc6510965)

[Chapter 3 53](#_Toc6510966)

[METHODOLOGY & WORKPLAN 53](#_Toc6510967)

[In this chapter, we will discuss that what are the existing methodologies and which one we have chosen for implementation of this project in an effective way, also we will discuss advantages of adopted methodology. 53](#_Toc6510968)

[What is methodology and why we need it? 53](#_Toc6510969)

[3.1 Adopted Methodology 53](#_Toc6510970)

[3.2 Roles & Responsibilities 54](#_Toc6510971)

[Chapter 4 55](#_Toc6510972)

[Software Design 55](#_Toc6510973)

[4.1 Activity Diagram 55](#_Toc6510974)

[4.2 Sequence Diagram 57](#_Toc6510975)

[4.2.1 SIMS Sequence Diagram 57](#_Toc6510976)

[4.2.2 User Login 58](#_Toc6510977)

[4.2.3 Add Department 58](#_Toc6510978)

[4.2.4 Add Class 59](#_Toc6510979)

[4.2.5 Add Course 60](#_Toc6510980)

[4.2.6 Add Student Information 60](#_Toc6510981)

[4.2.7Add Teacher Information 61](#_Toc6510982)

[4.2.8Add Student Result 61](#_Toc6510983)

[4.2.9Add Student Attendance 62](#_Toc6510984)

[4.2.10Add Student Fee Details 62](#_Toc6510985)

[4.3 Use Case Diagram 63](#_Toc6510986)

[4.3.1 Use Case Description 63](#_Toc6510987)

[*Use Case UC1: Login User* 63](#_Toc6510988)

[Main Success scenario: 64](#_Toc6510989)

[Extension: 64](#_Toc6510990)

[*Use Case UC2: Add Department* 65](#_Toc6510991)

[Main Success scenario: 65](#_Toc6510992)

[Extension: 65](#_Toc6510993)

[*Use Case UC2: Add Class* 66](#_Toc6510994)

[Main Success scenario: 66](#_Toc6510995)

[Extension: 66](#_Toc6510996)

[*Use Case UC2: Add Course* 67](#_Toc6510997)

[Main Success scenario: 67](#_Toc6510998)

[Extension: 67](#_Toc6510999)

[*Use Case UC2: Add Student Information* 69](#_Toc6511000)

[Main Success scenario: 69](#_Toc6511001)

[Extension: 69](#_Toc6511002)

[*Use Case UC2: Add Student Result* 70](#_Toc6511003)

[Main Success scenario: 71](#_Toc6511004)

[Extension: 71](#_Toc6511005)

[*Use Case UC2: Add Teacher Information* 71](#_Toc6511006)

[Main Success scenario: 72](#_Toc6511007)

[Extension: 72](#_Toc6511008)

[*Use Case UC2: Add Student Attendance* 72](#_Toc6511009)

[Main Success scenario: 73](#_Toc6511010)

[Extension: 73](#_Toc6511011)

[Main Success scenario: 74](#_Toc6511012)

[Extension: 74](#_Toc6511013)

[*Use Case UC2: Logout* 74](#_Toc6511014)

[Main Success scenario: 75](#_Toc6511015)

[Extension: 75](#_Toc6511016)

[4.4 Class Diagram 75](#_Toc6511017)

[4.5 ERD Diagram 77](#_Toc6511018)

[Chapter 5 78](#_Toc6511019)

[Implementation 78](#_Toc6511020)

[5.1 Introduction 78](#_Toc6511021)

[5.2 Tools and Technology Used 78](#_Toc6511022)

[5.2.1 Tools 78](#_Toc6511023)

[5.2.2 Technology 79](#_Toc6511024)

[Chapter 6 80](#_Toc6511025)

[Database Design 80](#_Toc6511026)

[6.1 Database Design 80](#_Toc6511027)

[6.2 Database 80](#_Toc6511028)

[6.3 Database tables 84](#_Toc6511029)

[Chapter 7 89](#_Toc6511030)

[User Interface 89](#_Toc6511031)

[7.1 Introduction 89](#_Toc6511032)

[7.2 User Interface Front End 90](#_Toc6511033)

[7.2.1 Login 90](#_Toc6511034)

[7.2.2 Dashboard Page 90](#_Toc6511035)

[7.2.3 Department Page 90](#_Toc6511036)

[7.2.4 Course Page 91](#_Toc6511037)

[7.2.5 Course Mapping Page 91](#_Toc6511038)

[Chapter 8 93](#_Toc6511039)

[SYSTEM TESTING 93](#_Toc6511040)

[8.1 Introduction 93](#_Toc6511041)

[8.2 Testing Plan 93](#_Toc6511042)

[8.2.1 Unit Testing 93](#_Toc6511043)

[8.2.2 System Testing 94](#_Toc6511044)

[8.2.3 Integration Testing 94](#_Toc6511045)

[8.2.4 User Acceptance Testing 94](#_Toc6511046)

[8.3 Test Cases 94](#_Toc6511047)

[8.4 Testing Results 96](#_Toc6511048)

[Chapter 9 98](#_Toc6511049)

[Conclusion & Future Work 98](#_Toc6511050)

[9.1 Conclusion 98](#_Toc6511051)

[9.2 Future Work 98](#_Toc6511052)

[REFERENCES: 99](#_Toc6511053)

# Chapter 1

# Introduction

## 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.

## 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.

### Historical perspective

The Student Information Management System is the SIMS that will hold the business in its goal to a paperless office environment. From the acronym itself, the SIMS 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.

## 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.

### 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

#### 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.

#### 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 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 adds new Course, delete, and update and view the course detail with ease.

#### Student Result Management Module

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

#### Student Attendance Module

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

#### Student Fee Module

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

#### 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.

#### 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.

## 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)

# Chapter 2

# Requirement Analysis

## Software Requirement Specification

### 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.

### 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.

### Proposed System

The proposed SIMS system provides detail general information about the students along with Educational, SIMS is used for adding, viewing information and updating students’ details. SIMS 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.

#### 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 user account. User who has valid login id and password can only login into their respective 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 user to add course detail like course name, course code, 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 user to mark the result of student in each subject/course.

#### Student Attendance Module

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

#### Student Fee Module

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

#### 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 student details by using teacher id.

## Specification Requirements

### 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).

### Non-Functional Requirements

#### 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.

#### 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.

#### 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.

### 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

## 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.

1. Student information management
2. Student result management
3. Department management
4. Class management
5. Courses Management
6. Student Fee Management
7. Student Attendance Management
8. Teacher Information Management

# Chapter 3

# METHODOLOGY & WORKPLAN

In this chapter, we will discuss that what are the existing methodologies and which one we have chosen for implementation of this project in an effective way, also we will discuss advantages of adopted methodology.

What is methodology and why we need it?

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.

## 3.1 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.

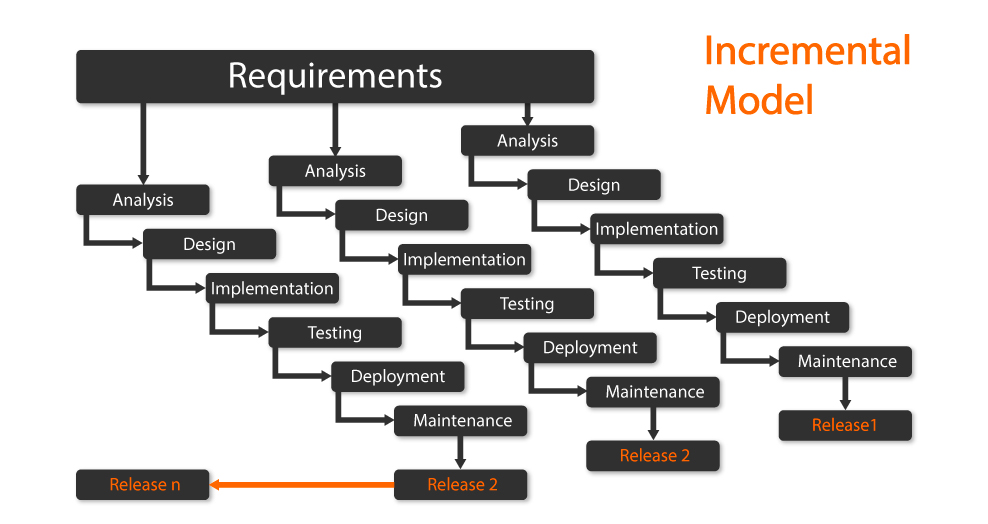
****

Figure 0.1: Adopted Methodology

## 3.2 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.

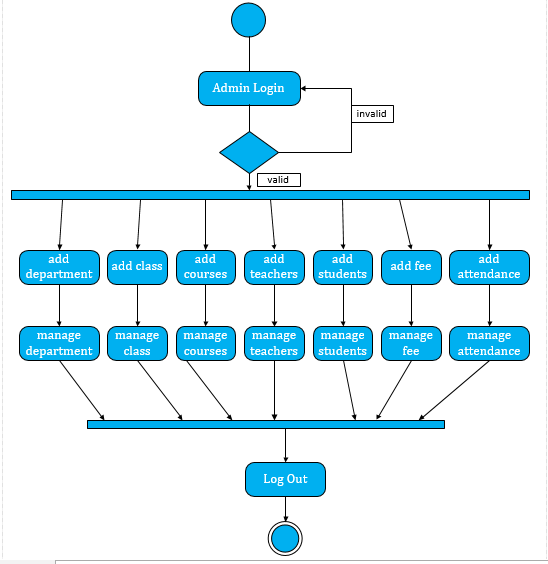
# Chapter 4

# 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.

## Activity Diagram

Activity diagrams are graphical representations of [workflows](https://en.wikipedia.org/wiki/Workflow) of stepwise activities and actions with support for choice, iteration and concurrency. In the [Unified Modeling Language](https://en.wikipedia.org/wiki/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.

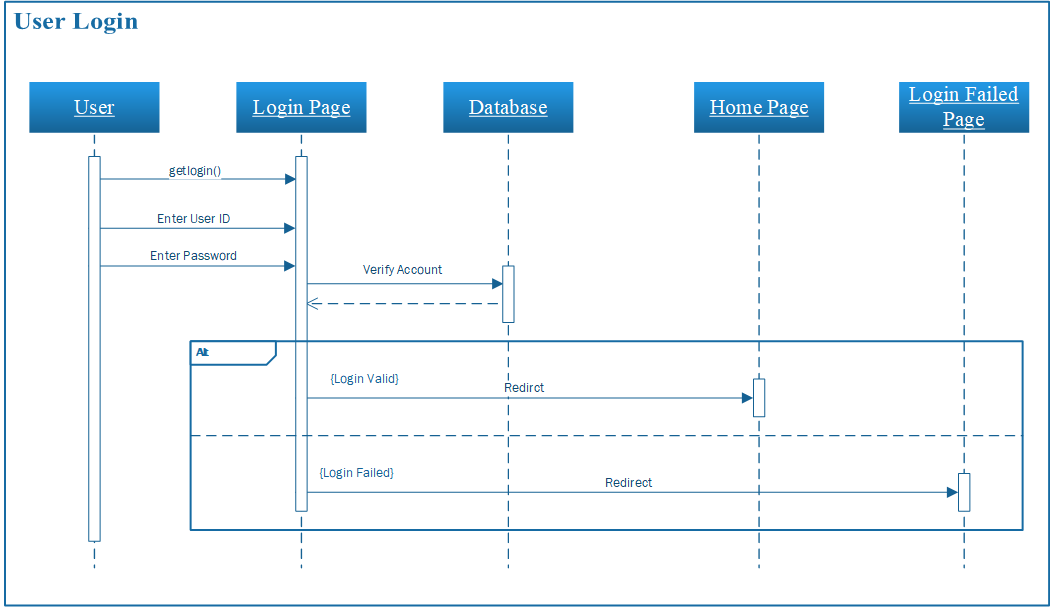
**Figure 4.1 Activity Diagram for Student Information System**

## Sequence Diagram

### 4.2.1 SIMS Sequence Diagram

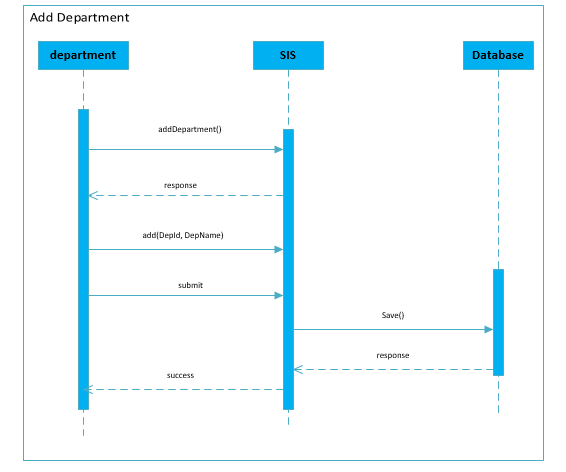
**Figure 4.2.1 Sequence Diagram**

### 4.2.2 User Login



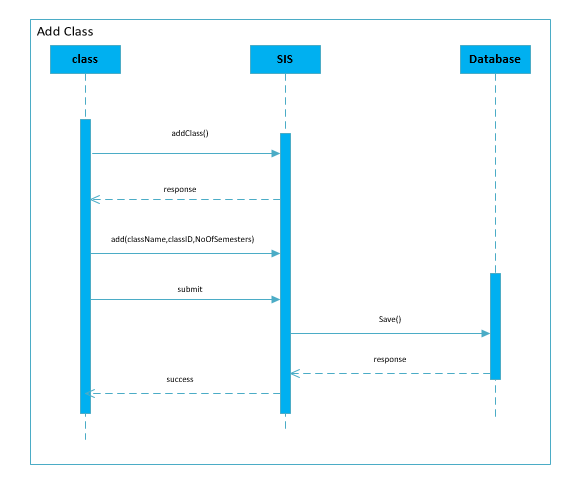
**Figure 4.2.2 User Login Sequence Diagram**

### 4.2.3 Add Department



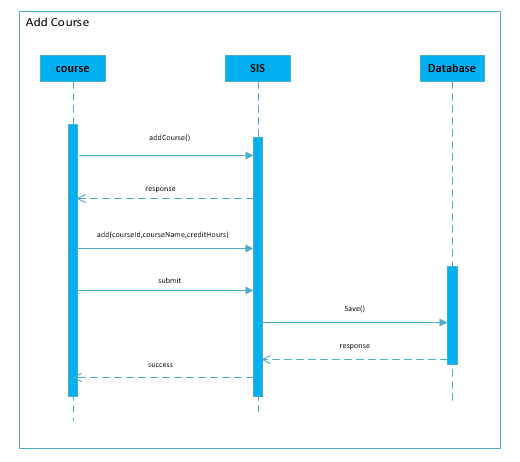
**Figure 4.2.3 Add Department Sequence Diagram**

### 4.2.4 Add Class



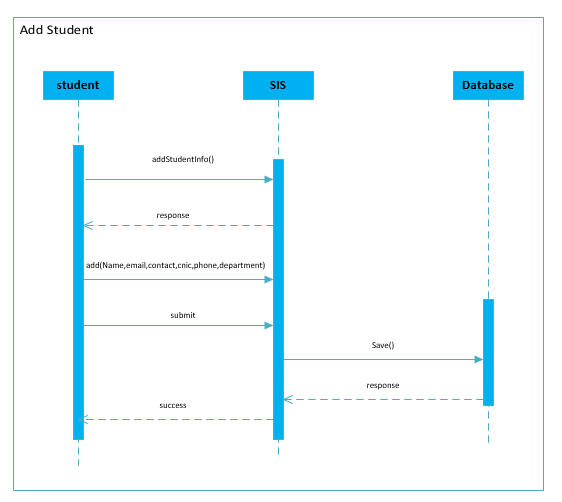
**Figure 4.2.4 Add Class Sequence Diagram**

### 4.2.5 Add Course



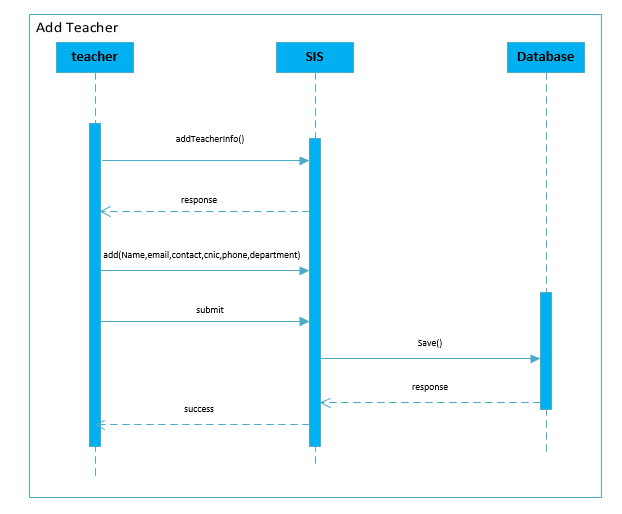
**Figure 4.2.5 Add Course Sequence Diagram**

### 4.2.6 Add Student Information



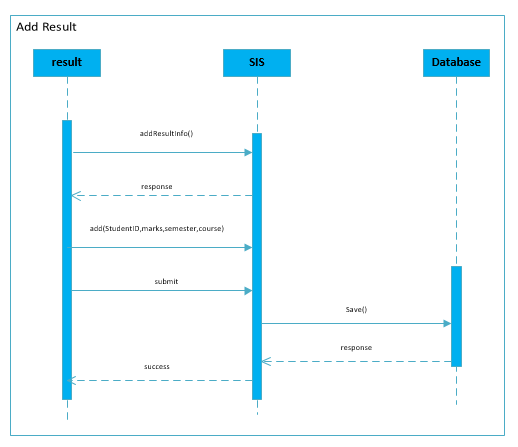
**Figure 4.2.6 Student Sequence Diagram**

### 4.2.7Add Teacher Information



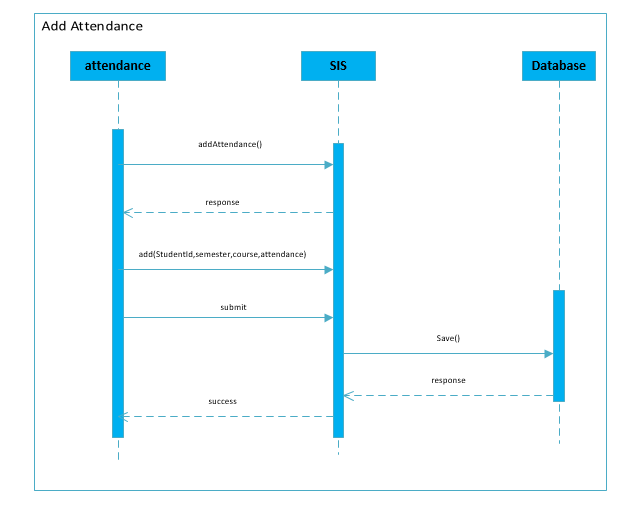
**Figure 4.2.7 Teacher Sequence Diagram**

### 4.2.8Add Student Result



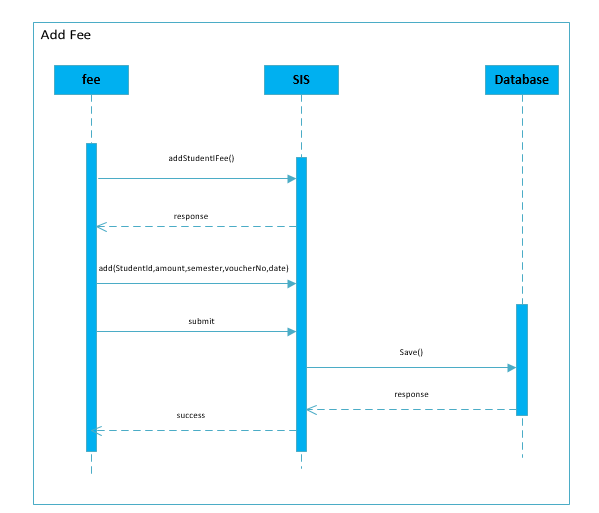
**Figure 4.2.8 Student Result Sequence Diagram**

### 4.2.9Add Student Attendance



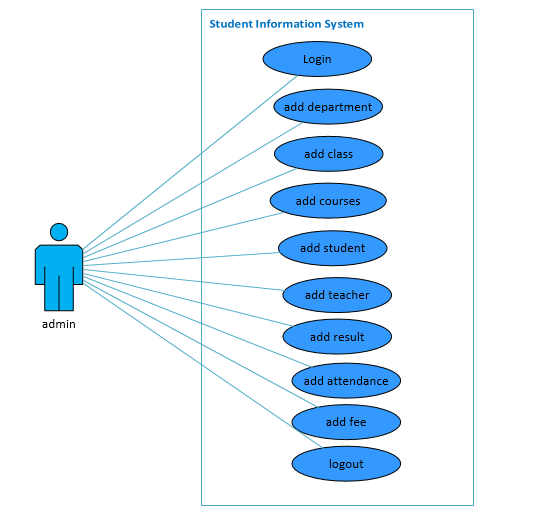
**Figure 4.2.9 Student Attendance Sequence Diagram**

### 4.2.10Add Student Fee Details



**Figure 4.2.10 Student Fee Sequence Diagram**

## 4.3 Use Case Diagram



Use case Diagram1

### 4.3.1 Use Case Description

|  |  |
| --- | --- |
| *Use Case UC1: Login User* | |
| Full Name | Login Admin |
| Level: | Administration |
| Primary Actor: | Administration |
| Secondary Actor | Administration |
| Description: | This use case describes how an actor logs into his/her SIMS account. |
| Precondition: | Actor must have an active login account |
| Post condition: | Actor is log in and can use the system |
| Main Success scenario: | This use case starts when actor wishes to login his/her account.   1. The actor click smart login link. 2. The actor types his/her ID and Password. 3. The actor hits the login button 4. 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 | |

|  |  |
| --- | --- |
| *Use Case UC2: Add Department* | |
| Full Name | Add Department |
| Level: | Admin |
| Primary Actor: | Admin |
| Secondary Actor | Admin |
| Description: | The Admin add Department in SIMS |
| Precondition: | 1. Admin must be log into system |
| Post condition: | Department id, name inserted into database |
| Main Success scenario: | This use case starts when admin wishes to add department in the SIMS.   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. |
| Extension: | System will generate and error and provide some suggestion if the requirement will not fulfill. |
| Use Case Diagram 1:Add\_Department | |

|  |  |
| --- | --- |
| *Use Case UC2: Add Class* | |
| Full Name | Add Class |
| Level: | Admin |
| Primary Actor: | Admin |
| Secondary Actor | Admin |
| Description: | The Admin add Class in SIMS |
| Precondition: | 1. Admin must be log into system |
| Post condition: | Class id, name, no of semesters inserted into database |
| Main Success scenario: | This use case starts when admin wishes to add class in the SIMS.   1. The admin click add Class link. 2. The Admin type Class name, Alias and status. 3. The Admin hits the save button 4. The system validates the Class attributes, if the data meets the requirements it will be inserted into database. |
| Extension: | System will generate and error and provide some suggestion if the requirement will not fulfill. |
| Use Case Diagram 1:Add\_Class | |

|  |  |
| --- | --- |
| *Use Case UC2: Add Course* | |
| Full Name | Add Course |
| Level: | Admin |
| Primary Actor: | Admin |
| Secondary Actor | Admin |
| Description: | The Admin add Course in SIMS |
| Precondition: | 1. Admin must be log into system |
| Post condition: | Course id, code and cr. hours inserted into database |
| Main Success scenario: | This use case starts when admin wishes to add Course in the SIMS.   1. The admin click add Course link. 2. The Admin type Course id, code and cr. hours. 3. The Admin hits the save button 4. The system validates the course attributes, if the data meets the requirements it will be inserted into database. |
| Extension: | System will generate and error and provide some suggestion if the requirement will not fulfill. |
| Use Case Diagram 1:Add\_Course | |

|  |  |
| --- | --- |
| *Use Case UC2: Add Student Information* | |
| Full Name | Add Student Information |
| Level: | Admin |
| Primary Actor: | Admin |
| Secondary Actor | Admin |
| Description: | The Admin add Student Information in SIMS |
| Precondition: | 1. 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: | This use case starts when admin wishes to add Student Information in the SIMS.   1. The admin click add Student Information link. 2. The Admin type Student Information Detail. 3. The Admin hits the save button 4. The system validates the Student Information attributes, if the data meets the requirements it will be inserted into database. |
| Extension: | System will generate and error and provide some suggestion if the requirement will not fulfill. |
| Use Case Diagram 1:Add\_Student | |

|  |  |
| --- | --- |
| *Use Case UC2: Add Student Result* | |
| Full Name | Add Student Result |
| Level: | Admin |
| Primary Actor: | Admin |
| Secondary Actor | Admin |
| Description: | The Admin add Student Result in SIMS |
| Precondition: | 1. 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: | This use case starts when admin wishes to add Student Result in the SIMS.   1. The admin click add Student Result link. 2. The Admin type Student Result Detail. 3. The Admin hits the save button 4. The system validates the Student Result attributes, if the data meets the requirements it will be inserted into database. |
| Extension: | System will generate and error and provide some suggestion if the requirement will not fulfill. |
| Use Case Diagram 1:Add\_Result | |

|  |  |
| --- | --- |
| *Use Case UC2: Add Teacher Information* | |
| Full Name | Add Student Information |
| Level: | Admin |
| Primary Actor: | Admin |
| Secondary Actor | Admin |
| Description: | The Admin add Teacher Information in SIMS |
| Precondition: | 1. Admin must be log into system |
| Post condition: | Teacher information (name, father name, CNIC, cell no., email, Address, gender, qualification, designation, type, hiring date) inserted into database. |
| Main Success scenario: | This use case starts when admin wishes to add Student Information in the SIMS.   1. The admin click add Teacher Information link. 2. The Admin type teacher Information Detail. 3. The Admin hits the save button 4. The system validates the teacher Information attributes, if the data meets the requirements it will be inserted into database. |
| Extension: | System will generate and error and provide some suggestion if the requirement will not fulfill. |
| Use Case Diagram 1:Add\_Teacher | |

|  |  |
| --- | --- |
| *Use Case UC2: Add Student Attendance* | |
| Full Name | Add Student Attendance |
| Level: | Admin |
| Primary Actor: | Admin |
| Secondary Actor | Admin |
| Description: | The Admin add Student Attendance in SIMS |
| Precondition: | 1. 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: | This use case starts when admin wishes to add Student Attendance Information in the SIMS.   1. The admin click add Student Attendance link. 2. The Admin type Student Attendance Detail. 3. The Admin hits the save button 4. The system validates the Student Attendance Information attributes, if the data meets the requirements it will be inserted into database. |
| Extension: | System will generate and error and provide some suggestion if the requirement will not fulfill. |
| Use Case Diagram 1:Add\_Student\_Attendance | |
| **Use Case UC2: Add Student Fee** | |
| Full Name | Add Student Fee Information |
| Level: | Admin |
| Primary Actor: | Admin |
| Secondary Actor | Admin |
| Description: | The Admin add Student Fee Information in SIMS |
| Precondition: | 1. 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: | This use case starts when admin wishes to add Student Fee Information in the SIMS.   1. The admin click add Student Fee Information link. 2. The Admin type Student Fee Detail. 3. The Admin hits the save button 4. The system validates the Student Fee Information attributes, if the data meets the requirements it will be inserted into database. |
| Extension: | System will generate and error and provide some suggestion if the requirement will not fulfill. |
| Use Case Diagram 1:Add\_Fee | |

|  |  |
| --- | --- |
| *Use Case UC2: Logout* | |
| Full Name | Admin Logout |
| Level: | Admin |
| Primary Actor: | Admin |
| Secondary Actor | Admin |
| Description: | The Admin Logout. |
| Precondition: | 1. Admin must be log into system |
| Main Success scenario: | This use case starts when admin wishes to add Student Information in the SIMS.   1. The admin click add Student Logout link. 2. The system logout the admin. |
| Extension: | System will generate and error and provide some suggestion if the requirement will not fulfill. |
| Use Case Diagram 1:LogOut | |

## Class Diagram

Classes are the work-horses of the design effort they actually perform the real work of the system. The other design elements subsystems, packages and collaborations simply describe how classes are grouped or how they interoperate. Capsules are also stereotyped classes, used to represent concurrent threads of execution in real-time systems. In such cases, other design classes are 'passive' classes, used within the execution context provided by the 'active' capsules. When the software architect and designer choose not to use a design approach based on capsules, it is still possible to model concurrent behavior using 'active' classes.



**Figure 4.3 Class Diagram**

## 4.5 ERD Diagram

**Figure 4.5 Entity Relation Diagram**

# Chapter 5

# Implementation

## 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.

## Tools and Technology Used

### Tools

#### Eclipse

Eclipse is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), and is the most widely used Java IDE. It contains a base [workspace](https://en.wikipedia.org/wiki/Workspace) and an extensible [plug-in](https://en.wikipedia.org/wiki/Plug-in_(computing)) system for customizing the environment. Eclipse is written mostly in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) and its primary use is for developing Java applications, but it may also be used to develop applications in other [programming languages](https://en.wikipedia.org/wiki/Programming_language) via plug-ins, including [Ada](https://en.wikipedia.org/wiki/Ada_(programming_language)), [ABAP](https://en.wikipedia.org/wiki/ABAP), [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)" \o "C Sharp (programming language)),[JavaScript](https://en.wikipedia.org/wiki/JavaScript),  [Perl](https://en.wikipedia.org/wiki/Perl), [PHP](https://en.wikipedia.org/wiki/PHP), [Prolog](https://en.wikipedia.org/wiki/Prolog), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [R](https://en.wikipedia.org/wiki/R_(programming_language)), [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)), [Rust](https://en.wikipedia.org/wiki/Rust_(programming_language)), [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)), and [Scheme](https://en.wikipedia.org/wiki/Scheme_(programming_language)). 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](https://en.wikipedia.org/wiki/Software_development_kit) (SDK) is [free and open-source software](https://en.wikipedia.org/wiki/Free_and_open-source_software), released under the terms of the [Eclipse Public License](https://en.wikipedia.org/wiki/Eclipse_Public_License), although it is incompatible with the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License).

#### 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).

### Technology

#### Language: Java

**Java** is a [general-purpose](https://en.wikipedia.org/wiki/General-purpose_language) [computer-programming language](https://en.wikipedia.org/wiki/Programming_language) that is [concurrent](https://en.wikipedia.org/wiki/Concurrent_computing), [class-based](https://en.wikipedia.org/wiki/Class-based_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), and specifically designed to have as few implementation [dependencies](https://en.wikipedia.org/wiki/Dependency_(computer_science)) as possible. It is intended to let [application developers](https://en.wikipedia.org/wiki/Application_developer) "[write once, run anywhere](https://en.wikipedia.org/wiki/Write_once,_run_anywhere)" (WORA), meaning that [compiled](https://en.wikipedia.org/wiki/Compiler) Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to ["bytecode"](https://en.wikipedia.org/wiki/Java_bytecode) that can run on any [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) regardless of the underlying [computer architecture](https://en.wikipedia.org/wiki/Computer_architecture). The language derives much of its original features from [SmallTalk](https://en.wikipedia.org/wiki/SmallTalk" \o "SmallTalk), with a [syntax](https://en.wikipedia.org/wiki/Syntax_(programming_languages)) similar to [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B), but it has fewer [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) facilities than either of them. As of 2018, Java was according to [Github](https://en.wikipedia.org/wiki/Github" \o "Github) one of the most [popular programming languages in use](https://en.wikipedia.org/wiki/Measuring_programming_language_popularity), particularly for [client-server](https://en.wikipedia.org/wiki/Client%E2%80%93server) [web applications](https://en.wikipedia.org/wiki/Web_applications), with a reported 9 million developers.

Java was originally developed by a Canadian [James Gosling](https://en.wikipedia.org/wiki/James_Gosling) at [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) ([which has since been acquired by Oracle](https://en.wikipedia.org/wiki/Sun_acquisition_by_Oracle)) and released in 1995 as a core component of Sun Microsystems' [Java platform](https://en.wikipedia.org/wiki/Java_(software_platform)).

# 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](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
* [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
* [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Uniquely identifies a row/record in another table
* [CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that all values in a column satisfies a specific condition
* [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column when no value is specified
* [INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - 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:

1. Primary Key
2. Secondary Key
3. Foreign key
4. Composite key
5. **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.

1. **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.

1. **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.

1. **Composite Key**

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

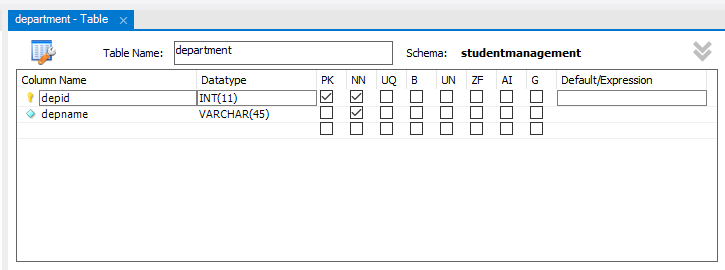
### 6.3 Database tables

#### 6.3.1 Admin Table

#### 

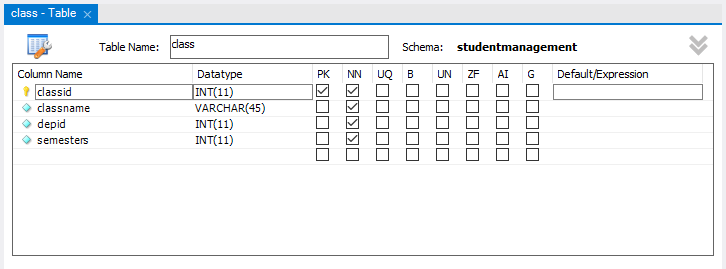
**Figure 6.3.1 Admin Database**

#### 6.3.2 Department Table



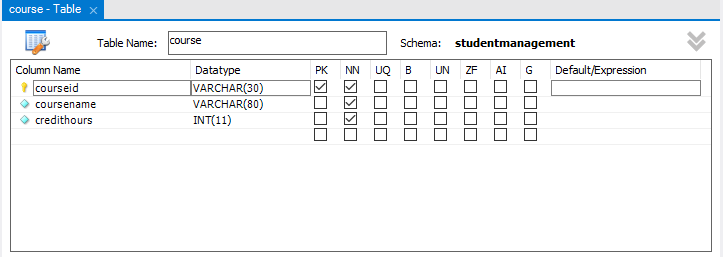
**Figure 6.3.2** **Department Database**

#### 6.3.3 Class Table



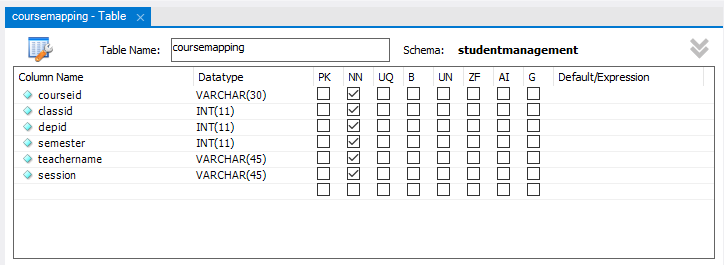
**Figure 6.3.3 Class Database**

#### 6.3.4 Course Table



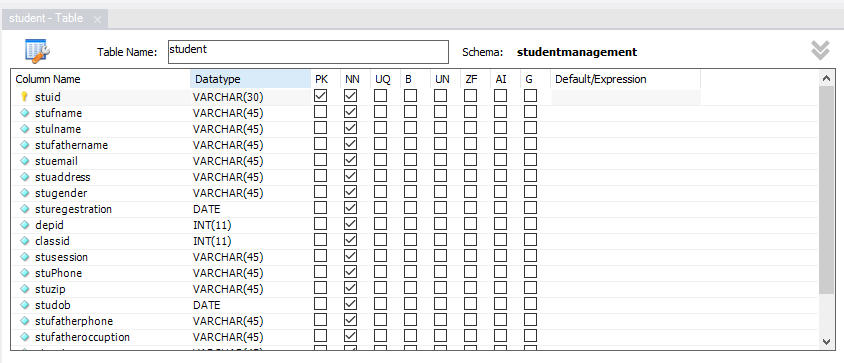
**Figure 6.3.4 Course Database**

#### 6.3.5 Course Mapping Table



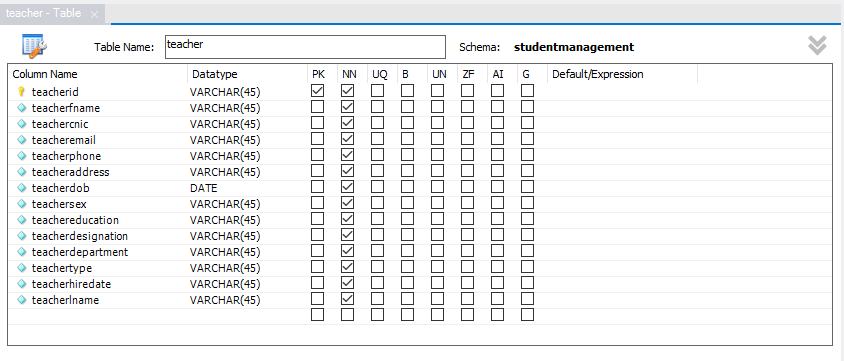
**Figure 6.3.5 Course Mapping Database**

#### 6.3.6 Student Table



**Figure 6.3.6 Student Database**

#### 6.3.7 Teacher Table



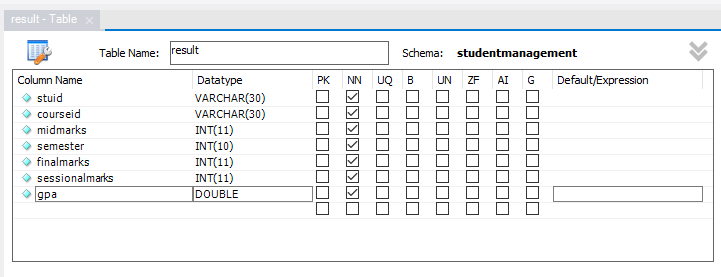
**Figure 6.3.7 Teacher Database**

#### 6.3.8 Student Attendance Table

#### 

**Figure 6.3.8 Student Attendance Database**

#### 6.3.9 Student Result Table



**Figure 6.3.9 Student Result Database**

#### 6.3.10 Student Fee Table

#### 

**Figure 6.3.10 Student Fee Database**

# 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](https://techterms.com/definition/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.

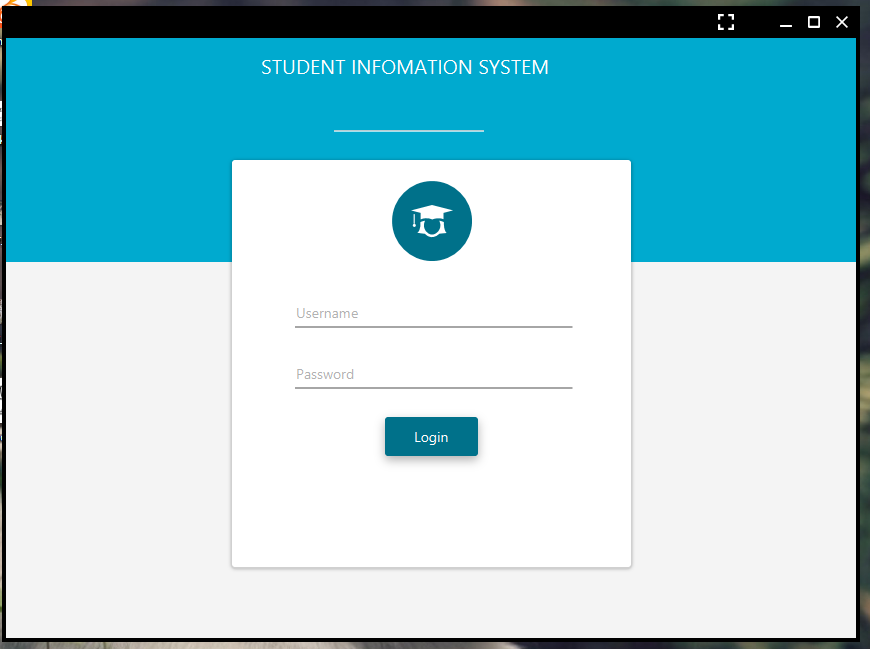
Nearly all software programs have a graphical user interface, or [GUI.](https://techterms.com/definition/gui) This means the program includes graphical controls, which the user can select using a mouse or keyboard. A typical GUI of a software program includes a [menu bar,](https://techterms.com/definition/menu_bar) [toolbar,](https://techterms.com/definition/toolbar) [windows,](https://techterms.com/definition/window) 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,](https://techterms.com/definition/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](https://techterms.com/definition/wordprocessor) and [Web browsers](https://techterms.com/definition/web_browser) all have rather similar interfaces, providing a consistent user experience across multiple programs.

Most [hardware](https://techterms.com/definition/hardware) devices also include a user interface, though it is typically not as complex as a [software](https://techterms.com/definition/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 of software, most are a combination of both. For example, to control a software program, you typically need to use a [keyboard](https://techterms.com/definition/keyboard) and [mouse,](https://techterms.com/definition/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

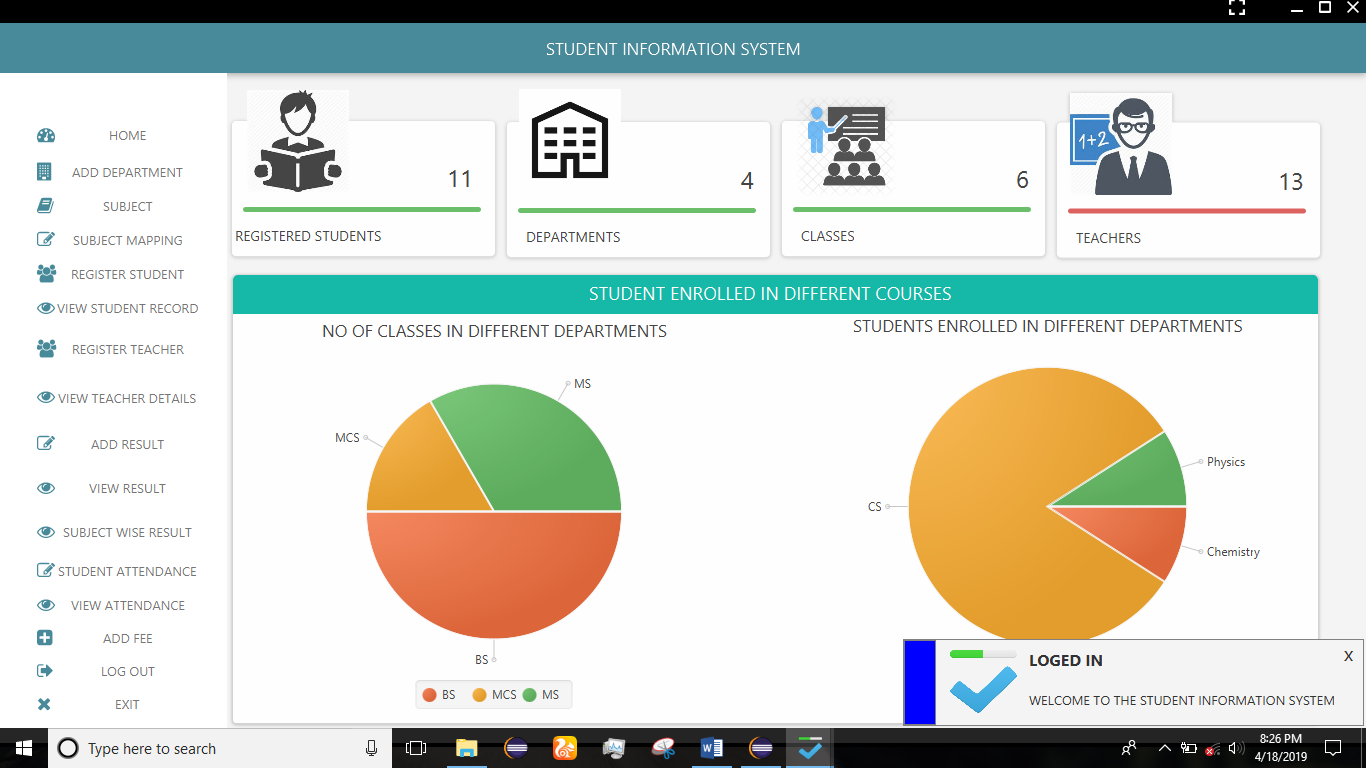


**Figure 7.2.1 Login**

**Description:**

This is the home page of the EPOS Cash and Carry System which can be seen when the user visits into the system.

## 7.2.2 Dashboard Page

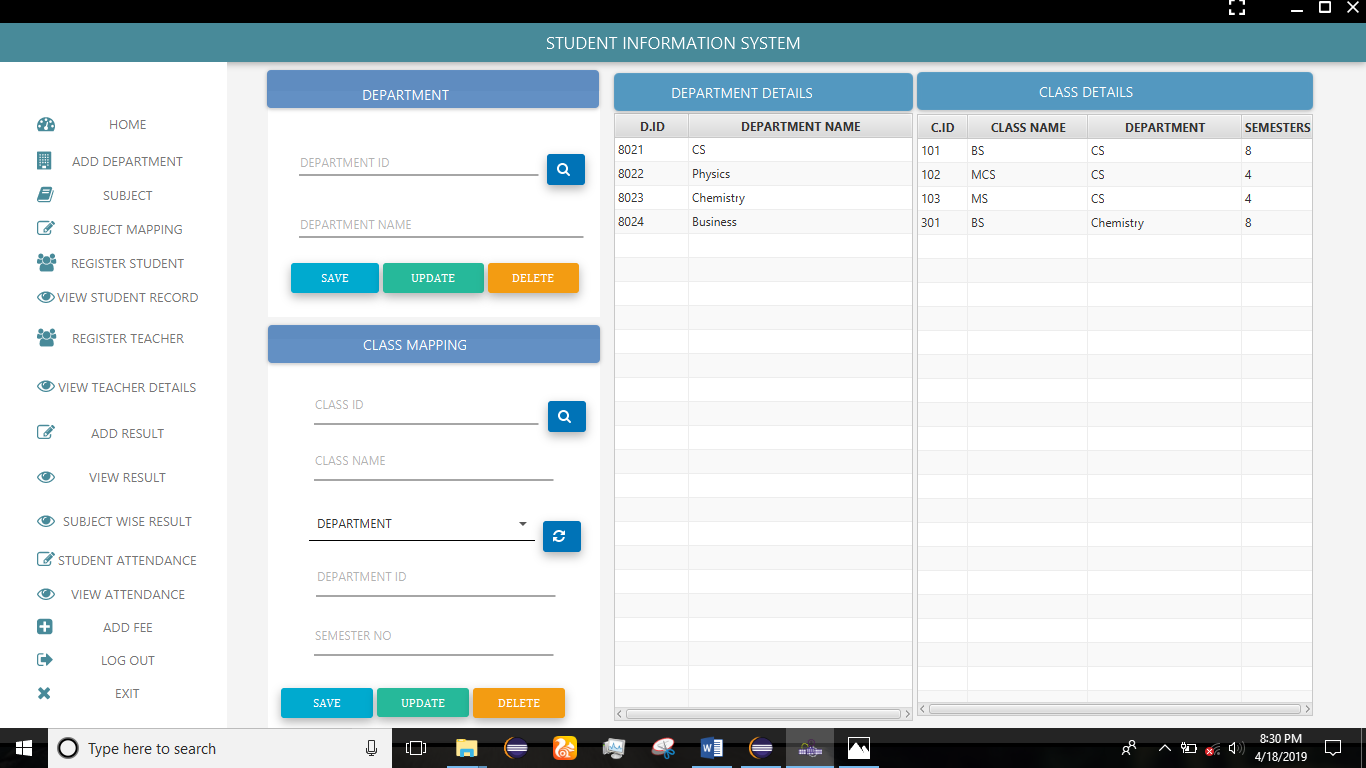


**Figure 7.2.2 Dashboard**

**Description:**

This is the login page of EPOS Cash and Carry System which can be seen when the user visits into the system. They can login into the system through it.

## 7.2.3 Department Page

****

**Figure 7.2.3 Department Page Description:**

This is the Register page of the EPOS Cash and Carry System which can be buy when user registered into the system.

## 7.2.4 Course Page

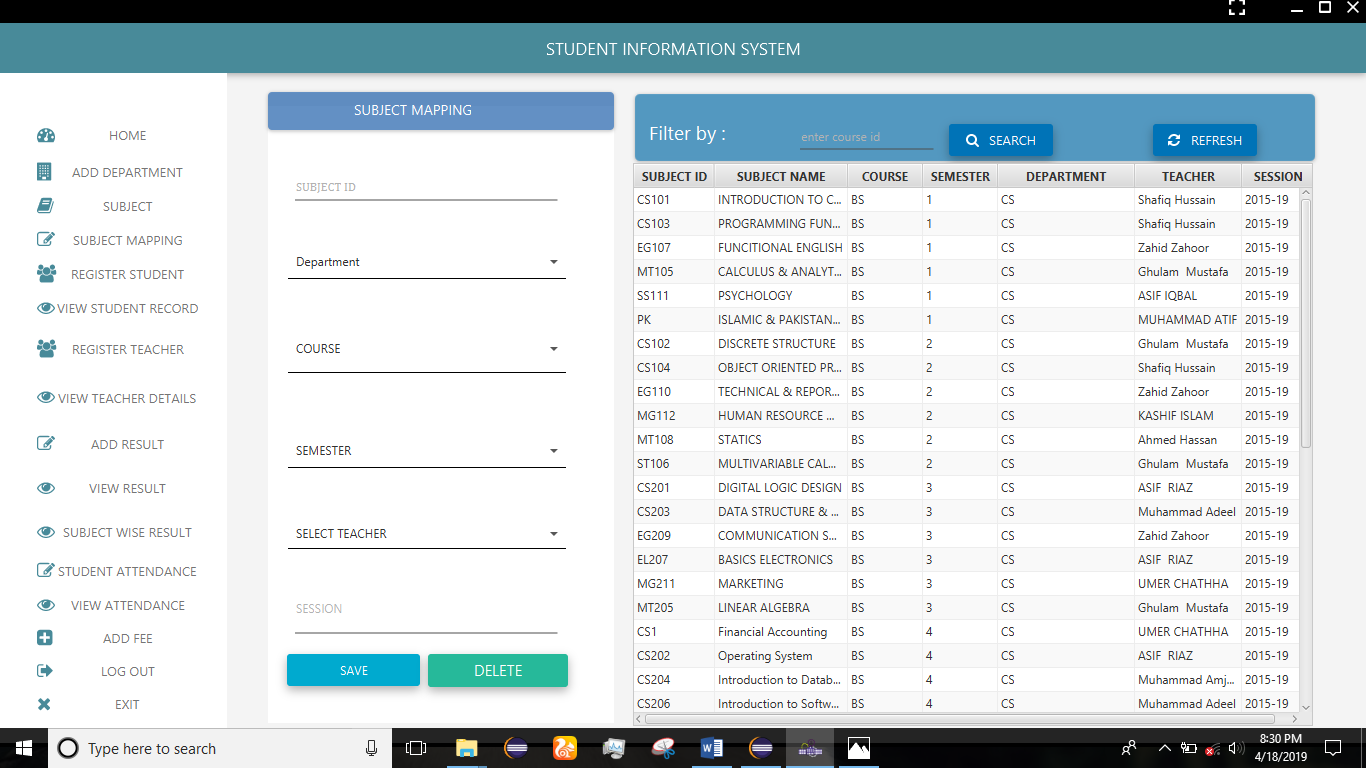
**Figure 7.2.4 Course Page**

**Description:**

This is the Services page of the EPOS Cash and Carry System which can be seen when the User visits into the system.

## 

## 7.2.5 Course Mapping Page

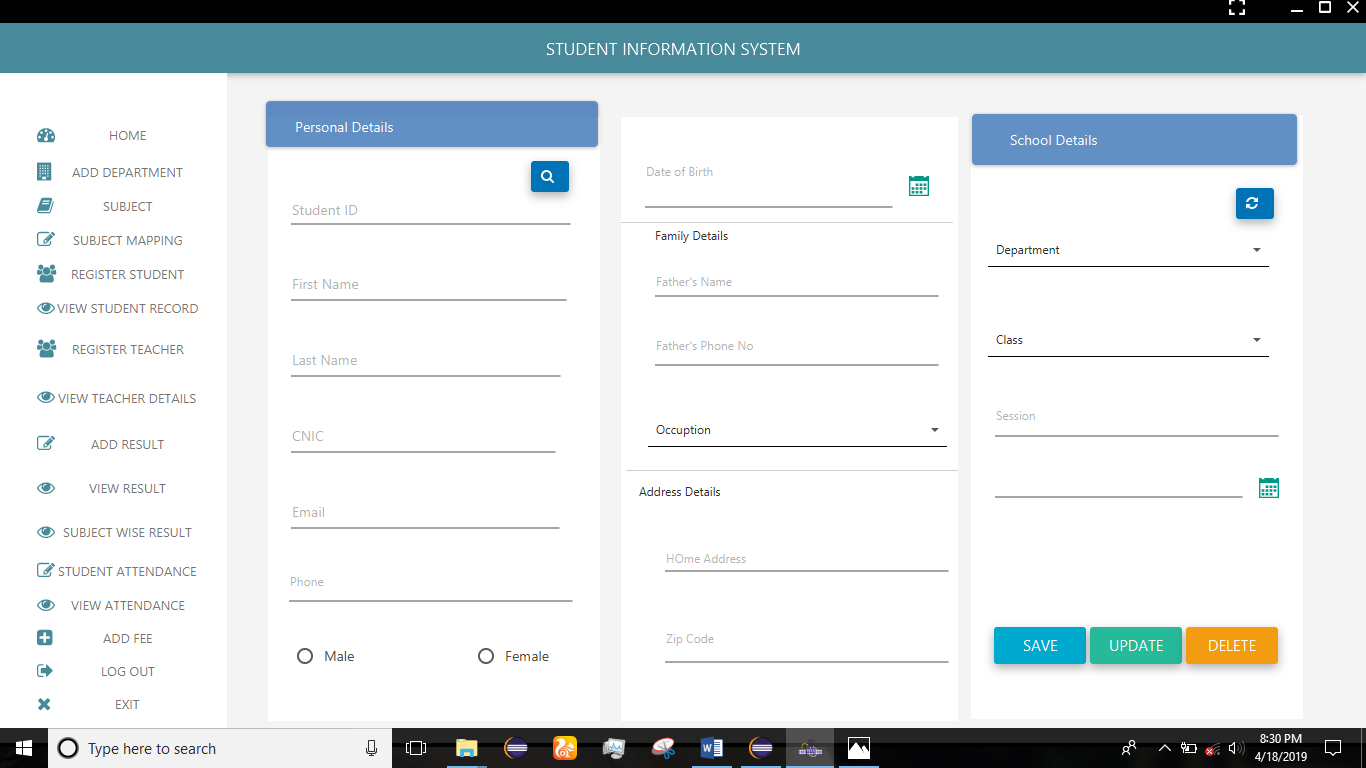


**Figure 7.2.5 Course Mapping Page**

**Description:**

This is the contact page of the EPOS Cash and Carry System which can be seen when the User visits into the system.

**7.2.6 Student Register Page**

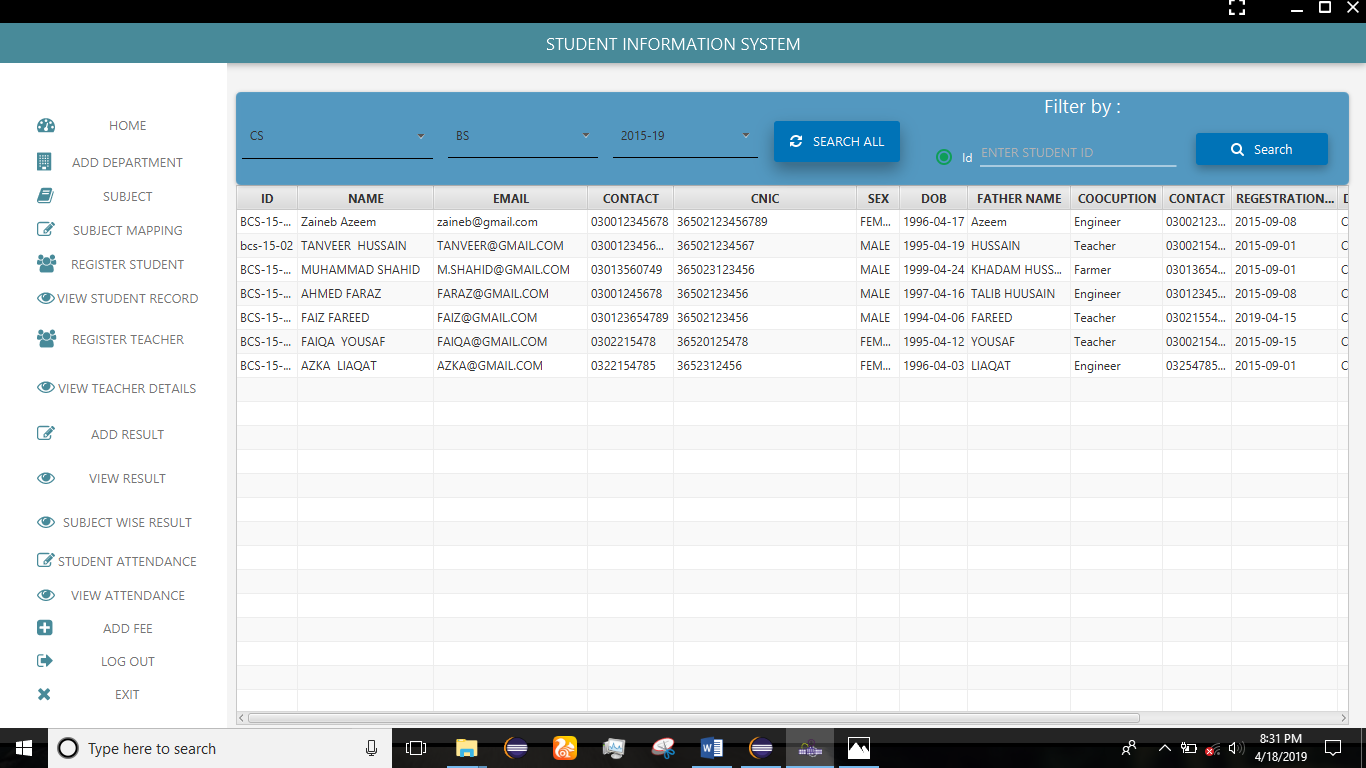
****

**Figure 7.2.6 Student Register Page**

**Description:**

This is the About page of the EPOS Cash and Carry System which can be seen when the Customer visits into the system.

**7.2.7 Student View page**

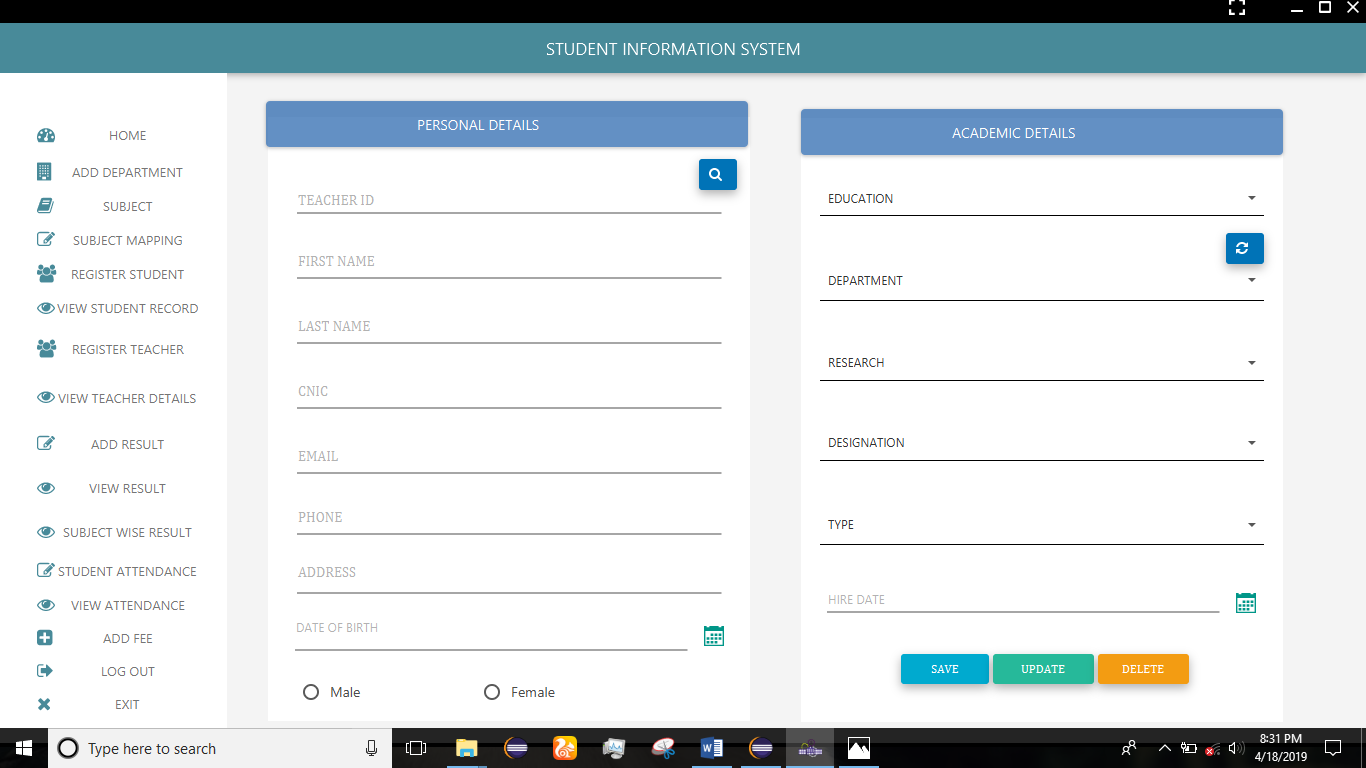
****

**Figure 7.2.7 Detail Page**

**Description:**

This is the products detail page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8 Teacher Register Page**

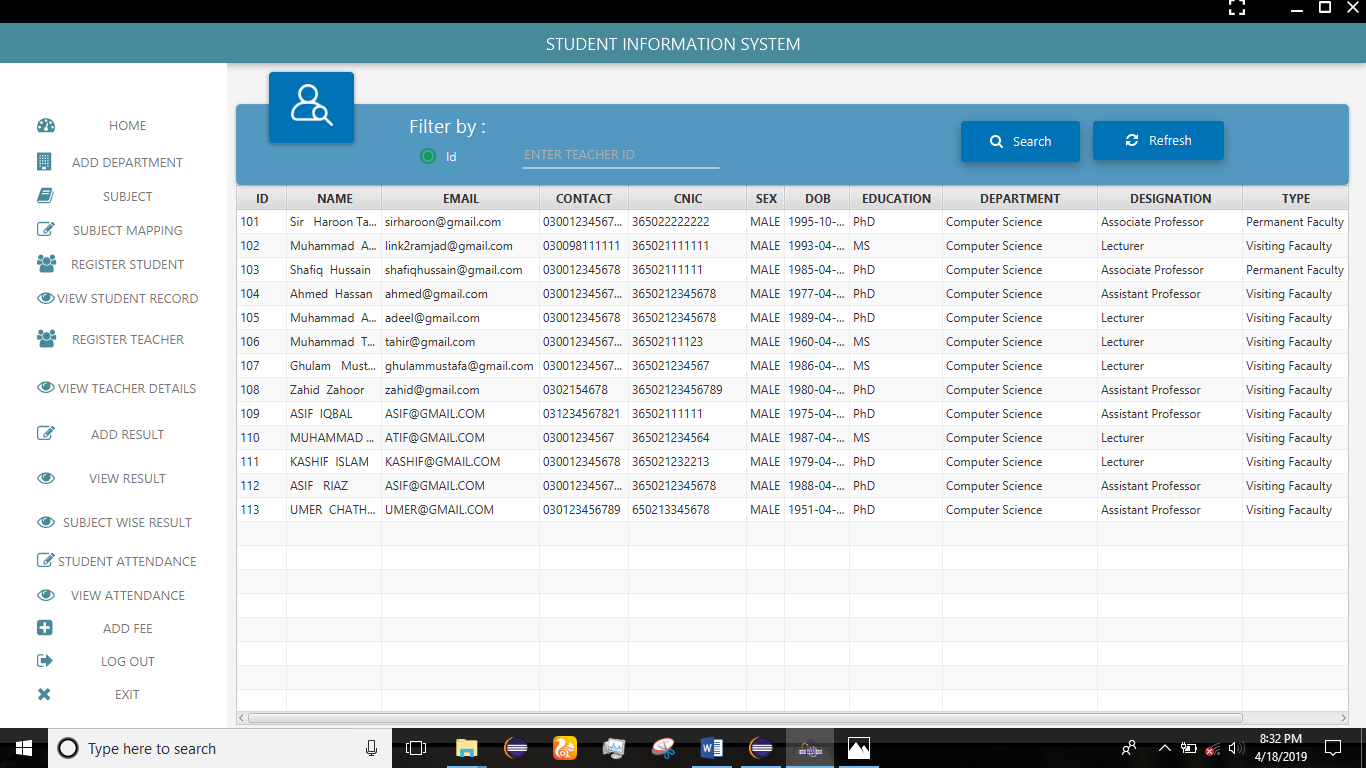
****

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8 Teacher Register Page**

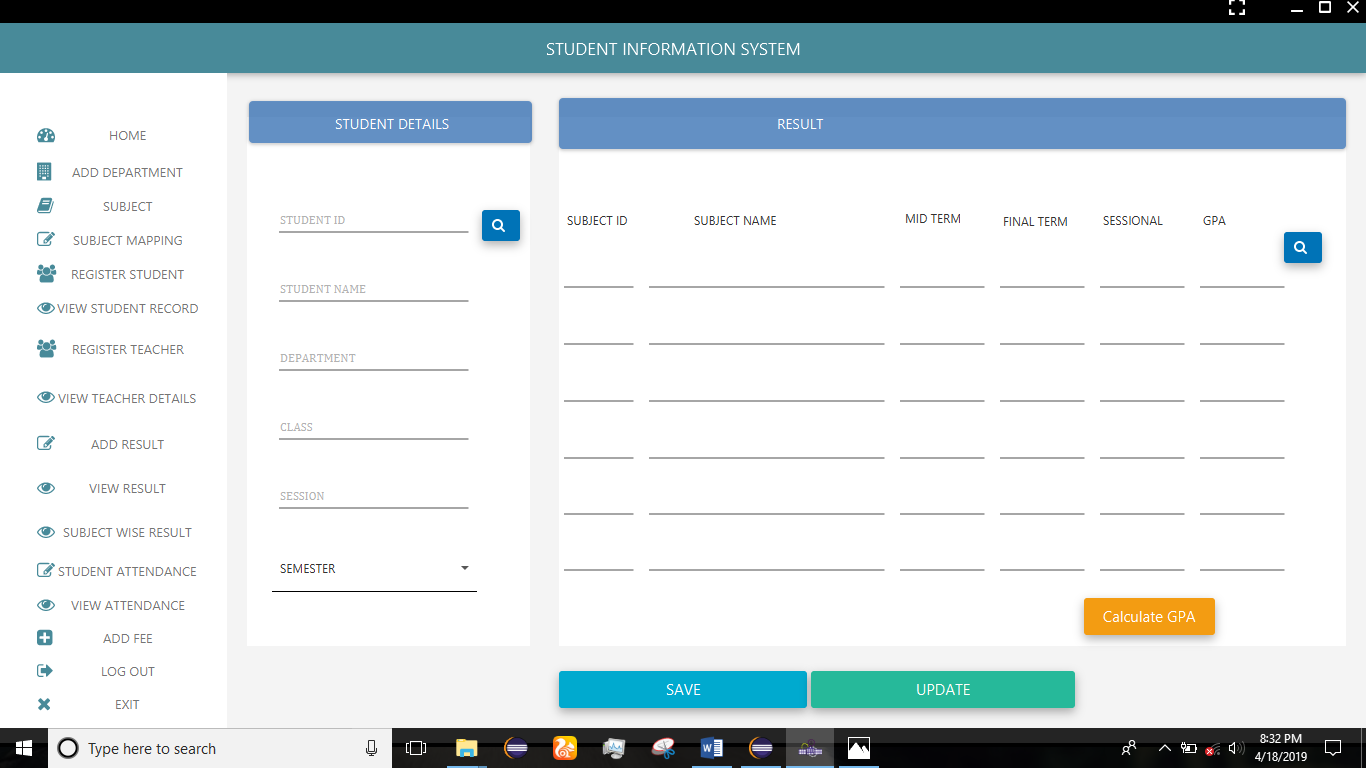
****

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8Add Result**

****

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

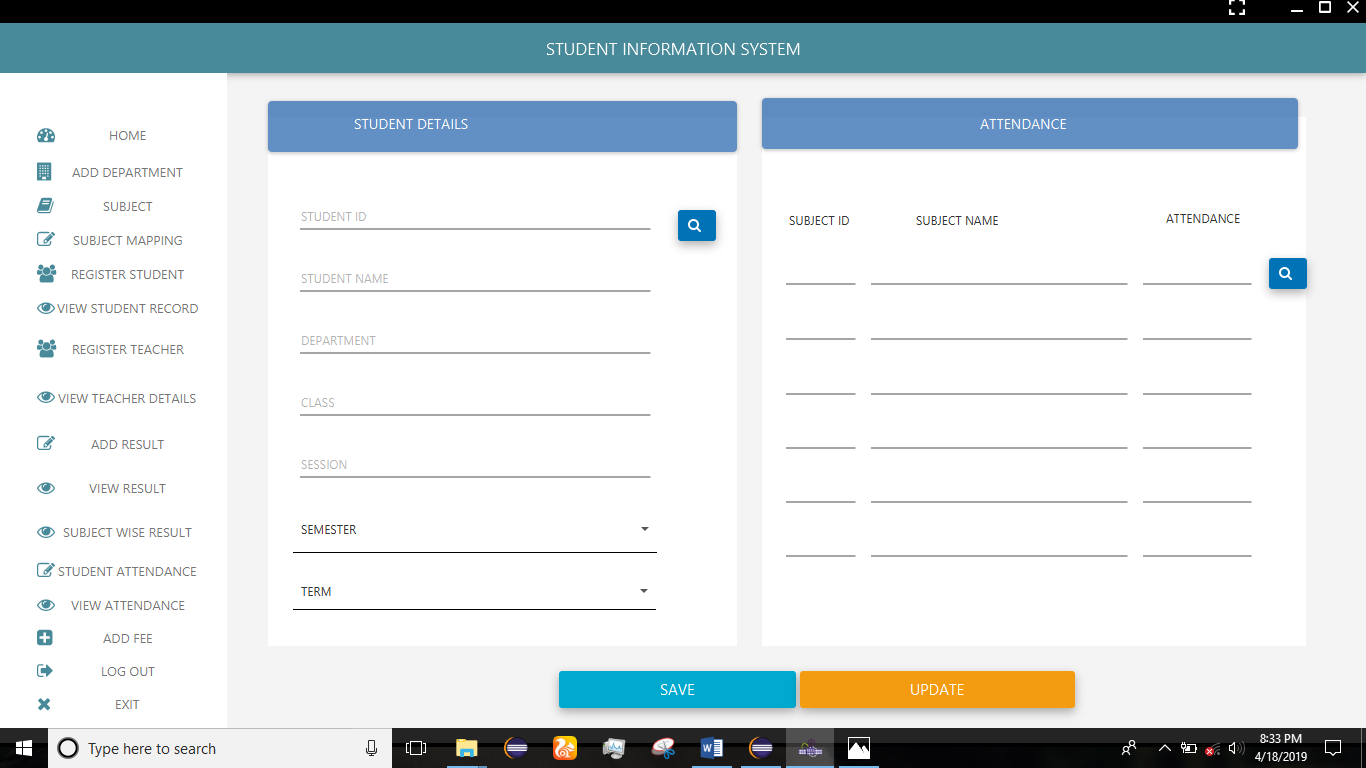
**7.2.8 View Result**

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8 Att attendance**

****

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8 Teacher Register Page**

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8 Teacher Register Page**

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8 Teacher Register Page**

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8 Teacher Register Page**

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8 Teacher Register Page**

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

**7.2.8 Teacher Register Page**

**Figure 7.2.8 Cart Page**

**Description:**

This is the Cart page of the EPOS Cash and Carry System which can be seen when you visits into the system can.

# 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 framework “**Student Information System”** with conclude remarks and will also discuss related future work of this application.

## 9.1 Conclusion

“Choose your Career Wisely (website)” is developed for provide education assistance, career guidance .This application provide a platform to student which is best for him. What opportunity they have. What they can do after inter.

This application provide such environment this is more efficient and user friendly. It will improve the performance of such organizations (schools, colleges and universities) and protect users from selecting wrong career. It is professionally beneficial.

## 9.2 Future Work

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

* All universities of Pakistan
* Provide Offline website
* Domain recommendation after Matriculation
* Job recommendation
* Fields detail
* Provide all fee structure and scholarships offered in universities
* Provide jobs prediction related to jobseeker’s qualification and interest
* Provide location of all educational institutes nearest to user’s location

# REFERENCES:

1. UML Revision Task Force. OMG Unified Modelling Language Specification, Version 1.4 (final draft). February 2001