

INTERDISCIPLINARY PROJECT REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and
Engineering

By
PODUTUR LAHARI
(REG NO. 40731138)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

APRIL 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

www.sathyabamauniversity.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **PODUTUR LAHARI (Reg. No: 40731138)** who carried out the "Student Information Management System" project under my supervision from FEBRUARY 2023 to APRIL 2023.

Internal Guide

Dr.Cruz Antony, M.C.A., Ph.D

Head of the Department

Dr.S.Vigneshwari, M.E., Ph.D

17-04-2023

Submitted for Viva-voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **PODUTUR LAHARI (Reg.No.40731138)** hereby declare that the professional training report entitled “**Student Information Management System**” done by me under the guidance of **Dr.Cruz Antony, M.C.A., Ph.D**, Professor, Department of CSE at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 17-04-2023

PLACE: Chennai

A handwritten signature in blue ink, appearing to read 'Podutur Lahari', with a horizontal line underneath.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

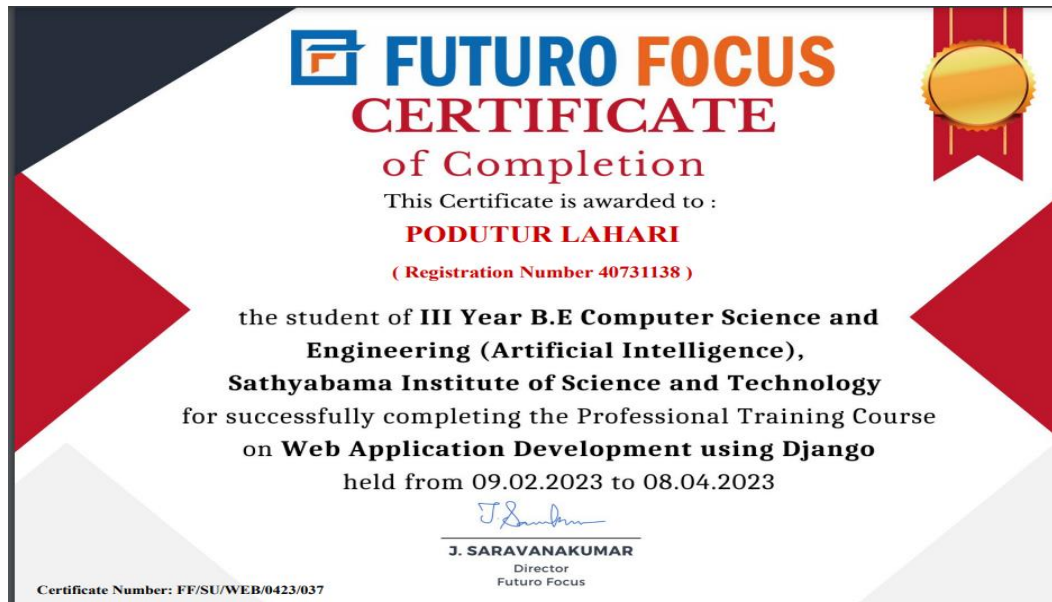
I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean, School of Computing , Dr,S.Vigneshwari, M.E., Ph.D and Dr.L.Lakshmanan M.E., Ph.D., Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.Cruz Antony, M.C.A., Ph.D** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



ABSTRACT

Student Information Management System can be used by education institutes to maintain the records of students easily. This project is entitled Online Student Result Management System. This is a web-based application project developed using Python and Django Framework. The application provides an online platform for certain school students or student parents to view the student's academic results. This project is a Django version of the project source code I published before which is the Online Student Result System using PHP/MySQL with Source Code. The application has a pleasant dark mode user interface using Django Material Dashboard Template. This project consists of user-friendly features and functionalities. 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 this project.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	i
	List of Figures	ii
1	INTRODUCTION	1
	1.1 Introduction and Purpose	1
	1.2 Definitions, Acronyms, and Abbreviations	1
	1.3 Technologies	1
	1.4 Methodology	2
	1.5 Advantages and Disadvantages of Student Information System	3
	1.6 Applications of Student Information Management System	5
2	AIM AND SCOPE OF THE PRESENT INVESTIGATION	7
	2.1 Aim	7
	2.2 Scope	7
	2.3 Product Functions	8
	2.4 Use classes and Characteristics	9
	2.5 Operating Environment	9
	2.6 Constraints	9
	2.7 Platform	9
	2.8 Use case model	10

	2.9 Assumptions and Dependencies	11
3	EXPERIMENTAL OR MATERIAL AND METHODS, ALGORITHMS USED	12
	3.1 Specific Requirements	12
	3.2 Functional Requirements	21
	3.3 Non-Functional Requirements	22
	3.3.1 Performance Requirements	22
	3.3.2 Safety Requirements	22
	3.3.3 Security Requirements	22
	3.4 Design Phase	22
	3.4.1 Scope and Purpose	22
	3.4.2 Overall System Design Objectives	23
	3.4.3 Structure of Design Document	23
	3.5 System Architecture Design	23
	3.5.1 System Architecture	23
	3.6 Data Design	25
	3.7 Functional Design Description	26
	3.7.1 Data Flow	26
	3.7.2 Decision Tree	27
4	RESULTS AND PERFORMANCE ANALYSIS	28
	4.1 Login Page	28
5	SUMMARY AND CONCLUSIONS	30

REFERNCE	30
APPENDIX	31
Project Description	37
Screenshots and Outputs	45
Source Code	56

LIST OF FIGURES

Fig No	Fig Title	Page No
1.5.1	Advantages of Student Information Management	3
2.8.1	Use model of student	11
3.1.1	Use Case Report-Login into the website	13
3.1.2	Use Case Report-Display Student Details	14
3.1.3	Use Case Report-Edit Student Detail into the website	16
3.1.4	Use Case Report-Register Student on Website	18
3.1.5	Use Case Report-Login into the System	19
3.1.6	Use Case Report-Edit Student Details into Database	21
3.5.1	Architecture Diagram	24
3.6.1	Entity Relationship Diagram	25
3.7.1	Data Flow Diagram	26
3.7.2	Decision Tree	27

CHAPTER – 1

INTRODUCTION

1.1 Introduction and Purpose

The objective of Student Information Management System (SIMS) is to allow the administrator of any organization to edit and find out the personal details of a student and allows the student to keep up to date with his profile. 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 and the student of any organization. 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 personal data.

1.2 Definitions, Acronyms, and Abbreviations :

- **Personal details:** Details of the student such as user id, phone number, address, image, resume, e-mail address, etc.
- **Contact details:** Details of contact associated with the student.
- **SRS:** System requirement Specification
- **WWW:** World Wide Web
- **Administrator:** A Login Id representing the user is an administrator & can access all the records details.

1.3 Technologies :

- PHP
- MYSQL
- JAVASCRIPT

- HTML
- CSS

1.4 METHODOLOGY

The methodology for developing a Student Information Management System (SIMS) typically involves the following steps:

Requirement gathering: This involves understanding the needs of the educational institution and identifying the data that needs to be stored and managed in the system. This can be done through interviews with stakeholders and a review of existing data systems.

System design: Based on the requirements gathered, the system architecture and design are created. This involves designing the database schema, creating wireframes for user interface design, and developing the technical specifications for the system.

Development: Once the system design is finalized, the actual development of the SIMS begins. This includes writing the code, configuring the database, developing the user interface, and integrating any necessary third-party software.

Testing: Once the development is complete, the system is tested to ensure that it works as intended. This involves testing for bugs, functionality, and performance.

Deployment: After testing, the system is deployed in the production environment. This involves installing the system on the appropriate servers, configuring the system, and ensuring that it is available to users.

Training and support: Once the system is deployed, the staff and users are trained to use the system effectively. Ongoing support and maintenance are provided to ensure that the system continues to function correctly.

Iteration: As the system is used, feedback is gathered from users to identify areas for improvement. This feedback is used to iterate and improve the system over time.

1.5 ADVANTAGES AND DISADVANTAGES OF STUDENT MANAGEMENT SYSTEM

Advantages :-

- i. **Centralized information management:** SIMS allows for the centralized management of student information, including academic records, attendance, personal information, and more. This improves the accuracy of data and makes it easier to access and analyze.

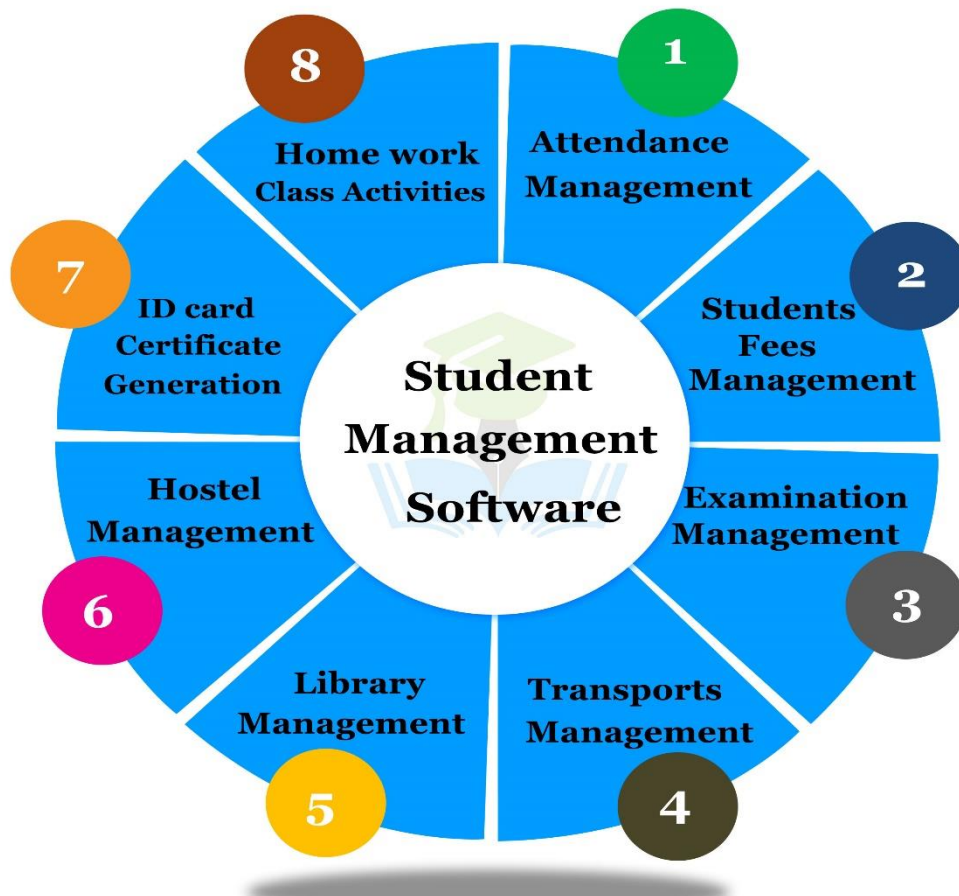


Fig 1.5.1 Advantages of Student Information Management

- ii. **Improved communication:** SIMS can improve communication between teachers, parents, and students by providing a platform for sharing information and updates.
- iii. **Efficient data collection and analysis:** With SIMS, data collection and analysis become more efficient, enabling schools to identify trends and make data-driven decisions.
- iv. **Automated processes:** SIMS can automate certain processes such as generating reports, sending notifications, and managing attendance. This reduces the workload of teachers and administrative staff.
- v. **Enhanced security:** SIMS can help protect student data by implementing security features such as user authentication, access control, and data encryption.
- vi. **Better resource management:** SIMS can help schools manage resources such as classrooms, staff, and equipment, ensuring that they are used effectively.
- vii. **Integration with other systems:** SIMS can be integrated with other systems such as learning management systems, student information systems, and financial systems.

Disadvantages :-

- i. **Implementation costs:** Implementing a SIMS can be expensive, particularly for smaller educational institutions. This includes not just the cost of purchasing and installing the system, but also the cost of training staff to use it.

- ii. **Technical issues:** SIMS can encounter technical issues such as software bugs, system crashes, and hardware failures. This can disrupt school operations and require additional IT support.
- iii. **Data privacy concerns:** As SIMS store sensitive student data, there are concerns about data privacy and security. Educational institutions must ensure that the system complies with data protection laws and implement measures to protect against data breaches.
- iv. **User resistance:** Some staff members may be resistant to using a SIMS, particularly if they are not technologically savvy. This can lead to low adoption rates and reduced system effectiveness.
- v. **Maintenance costs:** SIMS require ongoing maintenance to ensure that they continue to function properly. This includes software updates, hardware upgrades, and database backups.
- vi. **Complexity:** SIMS can be complex to use and administer, particularly for smaller educational institutions that may not have dedicated IT staff.

1.6 Applications of Student Information Management System :

- **Admissions management:** SIMS can manage the admissions process by storing applicant information, tracking application status, and managing enrollment.
- **Academic management:** SIMS can manage academic records, including student schedules, grades, transcripts, and other academic information.
- **Attendance management:** SIMS can track student attendance and generate reports to monitor attendance patterns and identify students who are at risk of falling behind.

- **Resource management:** SIMS can manage educational resources such as classrooms, textbooks, and equipment.
- **Reporting:** SIMS can generate reports on a range of academic and administrative data, including attendance, grades, enrollment, and student demographics.

CHAPTER - 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM

The website Student Information System is aimed towards recording a considerable number of student records and needs online assistance for managing records of students. The website should be user-friendly, 'quick to learn' and reliable website for the above purpose.

Student Information System is intended to be a stand-alone product and should not depend on the availability of other websites. The system will also have an administrator who has full-fledged rights with regard to performing all actions related to the control and management of the website.

2.2 Scope

Without a Student information System, managing and maintaining the details of the student is a tedious job for any organization.

Student Information system will store all the details of the students including their background information, educational qualifications, personal details and all the information related to their resume.

- a. **Login module:** Login module will help in the authentication of user accounts. Users who have a valid login id and password can only login into their respective accounts.
- b. **Search module:** Suppose there are hundreds of students and from this, we have to search for a particular student and we know the name of the student. In the manual system it is a tedious task though we know the name of the student, using this module we can easily search the student by specifying the name of the student in the search criteria. Thus this module will help the administrator in search the student with various criteria easily.

- c. **Registration Module and Account Management:** This module will help the student get registered from anywhere if internet is present .This module will really simplify the task of on paper registration. Also after successful registration the user can update information and change their password as and when required.
- d. **User Management:** This module will help the administrator in enabling/disabling a user account and updating user information as required.

Purpose of the project is to maintain details of the students such as storing information about:

- Student id
- Student password
- Student name
- Student DOB
- Student mailing address
- Gender
- Registration date
- Student status
- Contact no
- Qualification
- City
- Resume
- Image

2.3 Product Functions

There are two different users who will be using this product:

1. Administrator who can view and edit the details of any students.
2. Students who can view their details as well as they can edit their details.

The features that are available to the Administrator are:

- An Administrator can login into the system and perform any of the available operations.
- Can enable/disable student.

- Can edit student information to the database.
- Can make search for a specific student.
- Can access all the details of the student.

The features that are available to the student are:

- ❖ Students can log in to the system and can perform any of the available options.
- ❖ Can view his/her personal details.
- ❖ Can edit his/her personal details
- ❖ Can upload his/her resume.
- ❖ Can upload his/her image.

2.4 User Classes and Characteristics

There are mainly two kinds of users for the product. The users include:

1. Administrator
2. Student

2.5 Operating Environment

The product can run on any browser

2.6 Constraints

- Every user must be comfortable using computer.
- All operations are in English so user must have basic knowledge of English.

2.7 Platform

Operating Systems: Microsoft Windows

Technologies Used:

- Front End: HTML and Javascript
- Web designing language: PHP

- RDBMS(Back end): MySQL

Software Requirements:

- PHP 5.0
- APACHE HTTP Server
- Dreamweaver, FrontPage for Front End Programming
- Microsoft Windows or Linux

Hardware Requirements:

- Intel Pentium IV processor or equivalent or higher
- 512 MB Ram or Higher
- 20 GB HDD or Higher
- Network Connectivity

2.8 USE CASE MODEL

1. **Administrator:** Responsible for managing student records.
 - Login into the website
 - Update student details
 - Search student details
 - Display student details
 - Enable/Disable student
2. **Student:** Has the access rights to view and edit their personal details.
 - Login into the website
 - Display student details
 - Edit their details
 - Upload their images
 - Upload their resumes

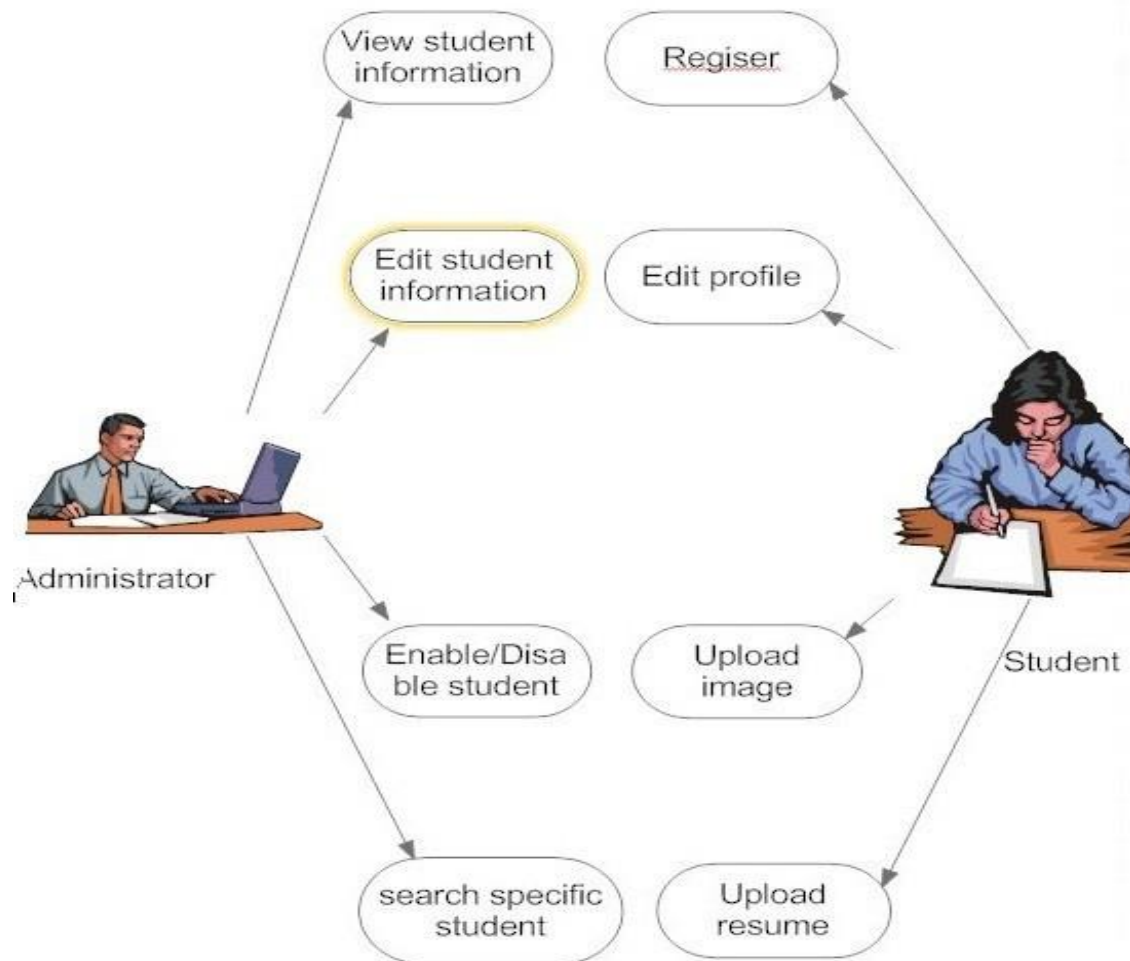


Fig 2.8.1 Use model of student

2.9 Assumptions & Dependencies

- Administrator is created in the system already.
- Roles and tasks are predefined.

CHAPTER – 3

EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS

3.1 Specific Requirements :

Use Case Report

1. Administrator: Responsible for managing student details.

- **Use-case:** Login into the website
- **Goal in context:** Gain access to the website
- **Brief Description:** This use case is used when the administrator wants to access the website to enable/disable/update the personal details of the student.
- **Preconditions:** The Administrator must be logged onto the website in order for this use case to begin.
- **Basic Flow:**
 - The Website prompts the administrator for the user name and password.
 - The Administrator enters the user name and password.
 - The Website verifies the password and sets the user's authorization.
 - The Administrator is given access to the Website to perform his tasks.
- **Alternative Flow:**
 - The administrator enters an invalid username and password then he will not be allowed to enter the website.
- **Post conditions:** The website state is unchanged by this use case.

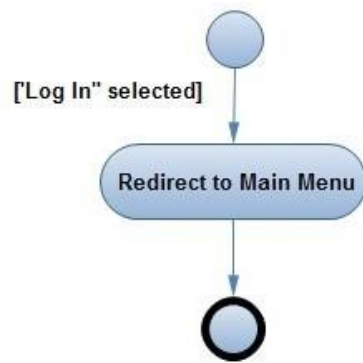


Fig 3.1.1 Use Case Report- Login into the website

- **Use Case :** Display student details
- **Goal in context:** View the details of a student.
- **Brief Description:** This use case is used when the administrator wants to view the personal details of the students already existing in the database on the screen.
- **Preconditions:**
 - The Administrator must be logged into the system in order for usecase to begin.
 - The details of the student must pre-exist in the database
 - The student id must be entered correctly.
- **Basic Flow:**
 - The Administrator logs onto the System.
 - The Administrator search the student from following keys:-
 - Student id
 - First/last name
 - Registration date
 - status

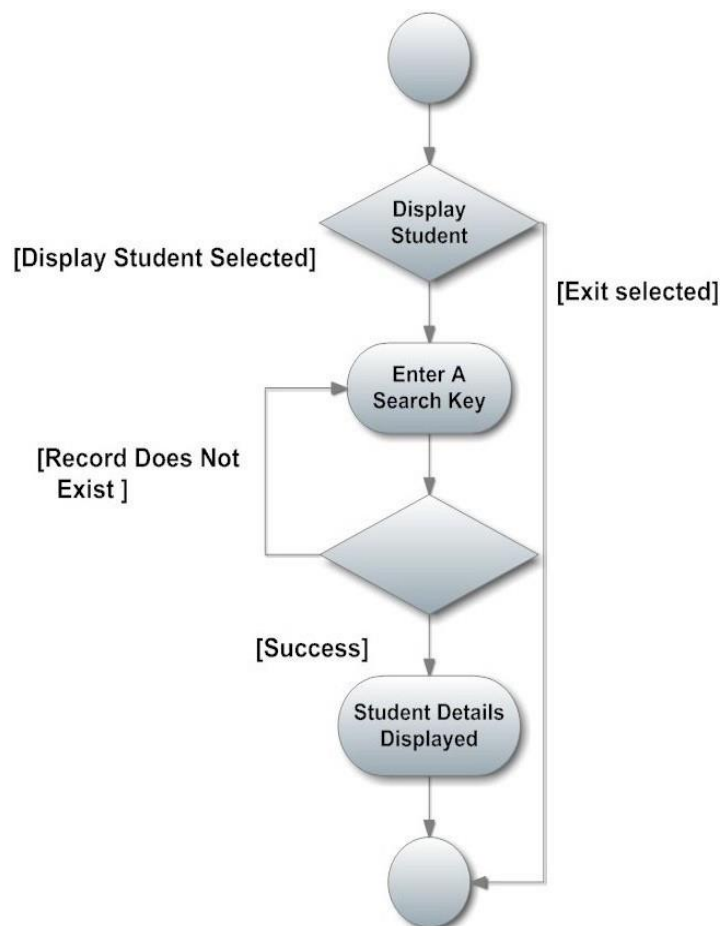
- The System prompts for the student detail from one of the above keys.
- The student details are displayed on the screen.

- **Alternative Flow:**

Student Not Found

If in the **Display a student** sub-flows, a student with the specified id number does not exist, The system displays an error message. The Administrator can then enter a different id number or cancel the operation, at which point the use case ends.

- **Post conditions:**



Use Case Report-Display Student Details

Fig 3.1.2 Use case Report-Display Student Details

- **Use Case :** Edit student details
- **Goal in context:** Edit the details of a student
- **Brief Description:** This use case is used when the administrator wants to edit the personal details of the himself/herself already existing in the database.
- **Preconditions:**
 1. The Administrator must be logged into the system in order for this usecase to begin.
 2. The details of the student must pre-exist in the database.
- **Basic Flow:**

The Administrator logs onto the System.

 - The Administrator can edit following keys:-
 - First/last name
 - Gender
 - DOB
 - Contact no
 - Qualification
 - City
 - Email1
 - Email2
 - Address
 - Description

- I. The Website updates the database according to edited details.
- II. The student details are edited in the database.

- **Alternative Flow:**

There is no alternative flow of this use case diagram.

- **Post conditions:**

The student details get updated in the database.

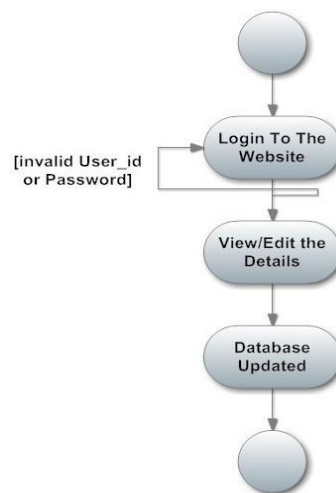


Fig 3.1.3 Use Case Report- Edit student detail into the website

2.Student

- **Use Case :** student registration
- **Goal in context:** Registration of a student
- **Brief Description:** This use case is used when the student register himself/herself in the database online.
- **Preconditions:**
 1. The Student must accessed the website in order for this use case to begin.
 2. The user id must be unique and entered correctly.

- **Basic Flow:**

1. The Student enters into the website.
2. The student fill his/her details from the following keys:-

- Student id
- password
- First/last name
- Status
- Gender
- DOB
- Contact no
- Qualification
- City
- Email1
- Email2
- Address
- Description
- Resume
- Image

- The System details are added to the database.
- The student details are displayed on the screen.

- **Alternative Flow:**

User ID not unique: if the user id entered is not unique then it will show an errormessage.

- **Post conditions**

The student get registered on the website and to login into that particular the administrator must enable it.

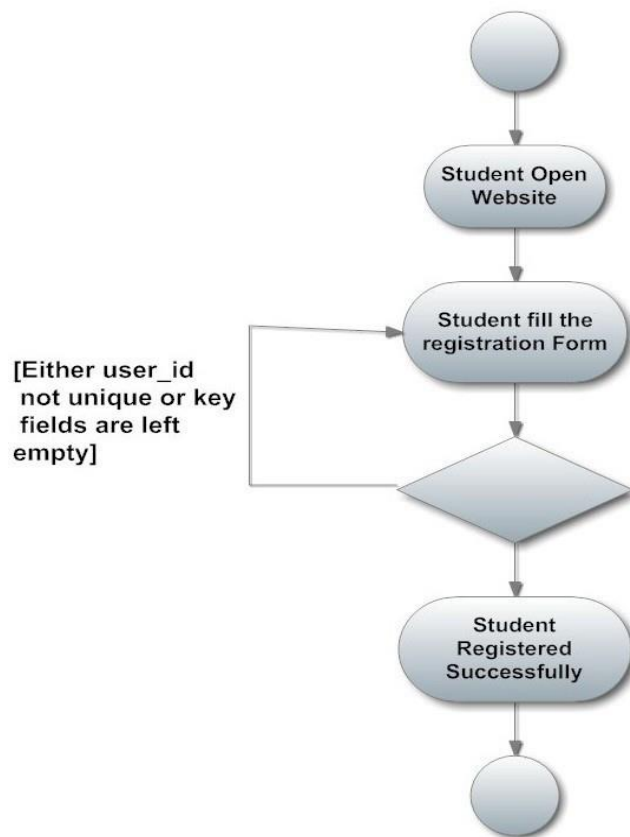


Fig 3.1.4 Use Case Report- Register student on website

- **Use-case:** Login into the website
- **Goal in context:** Gain access to the website
- **Brief Description:** This use case is used when the student wants to access the website
- **Preconditions:** The Administrator must enabled the particular student onto the website in order for this use case to begin.

- **Basic Flow:**

1. The website prompts the student for the user name and password.
2. The Student enters the user name and password.
3. The website verifies the password and sets the user's authorization.
4. The Student is given access to the website to perform his tasks.

- **Alternative Flow:**

The Student enters invalid username and password then he will not be allowed to enter the website.

- **Post conditions:** The website state is unchanged by this use case.

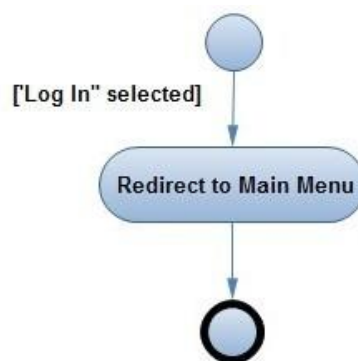


Fig 3.1.5 Use Case Report- Login into the system

- **Use Case :** Edit student details
- **Goal in context:** Edit the details of a student
- **Brief Description:** This use case is used when the student wants to edit the personal details of the himself/herself already existing in the database.

- **Preconditions:**

- The Student must be logged into the system in order for this use case to begin.
- The details of the student must pre-exist in the database
- The student must be enabled by administrator.

- **Basic Flow:**

- The Student logs onto the System.
- The Student can edit following keys:

- First/last name
- Gender
- DOB
- Contact no
- Qualification
- City
- Email1
- Email2
- Address
- Description

- The Website updates the database according to edited details.
- The student details are edited in the database.

- **Alternative Flow:**

There is no alternative flow of this use case diagram.

- **Post conditions:**

The student details get updated in the database

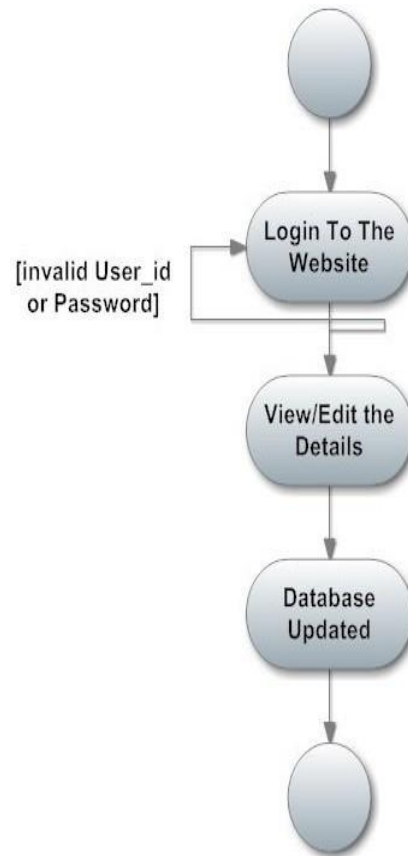


Fig 3.1.6 Use Case Report- Edit Student Details Into Database

3.2 Functional Requirements :

- The Administrator will be given more powers (enable/disable/ update) than other users.
- It will be ensured that the information entered is of the correct format. For example name cannot contain numbers. In case if incorrect form of information is added, the user will be asked to fill the information again.
- The system can be accessed anytime.

3.3 Non- Functional Requirement :

3.3.1 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 organization. Therefore, it is expected that the database would perform functionally all the requirements that are specified.

3.3.2 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.

3.3.3 Security Requirements:

We are going to develop a secured database. There are different categories of users namely Administrator, Student who will be viewing either all or some specific information from the database.

Depending upon the category of user the access rights are decided. It means if the user is an administrator then he can be able to modify the data, append etc. All other users only have the rights to retrieve the information about database.

3.4 DESIGN PHASE

3.4.1 Scope and purpose

The purpose of the design phase is to develop a clear understanding of what the developer want people to gain from his/her project. As you the developer work on the project, the test for every design decision should be "Does this feature fulfill the ultimate purpose of the project?"

A purpose statement affects the design process by explaining what the developer wants the project to do, rather than describing the project itself.

The Design Document will verify that the current design meets all of the explicit requirements contained in the system model as well as the implicit requirements desired by the customer.

3.4.2 Overall System Design Objectives

The overall system design objective is to provide an efficient, modular design that will reduce the system's complexity, facilitate change and result in an easy implementation. This will be accomplished by designing strongly cohesion system with minimal coupling. In addition, this document will provide interface design models that are consistent user friendly and will provide straight forward transition through the various system functions.

3.4.3 Structure of Design Document

System Architecture Design – The System Architecture section has detailed diagram of the system, server and client architect

Data Design – The data Design includes an ERD as well as a

Functional Design Description – This section has the functional partitioning from the SRS, and goes into great detail to describe each function.

3.5 System Architecture Design

3.5.1 System Architecture

The SIMS is a system that contains a major part which includes: student Detail, Student image, and resume.

The user selects one of the available options as input to the system. According to the input by the user the system acts and the rest of the

functions are performed accordingly. The administrator can operate on any student details. But the normal student or users can only access their details of all the functionalities.

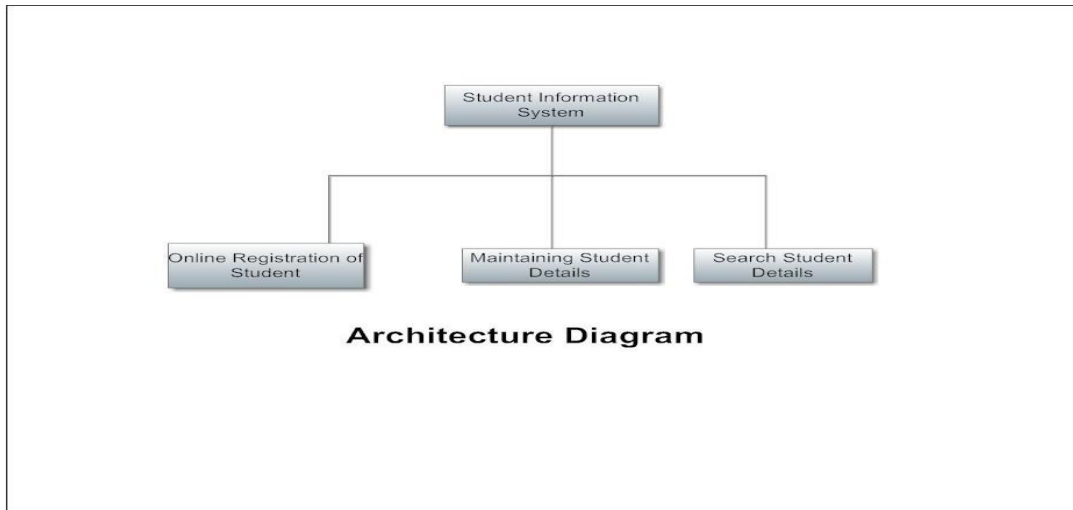


Fig 3.5.1 Architecture Diagram

3.6 Data Design

3.6.1 Entity Relationship Diagram :

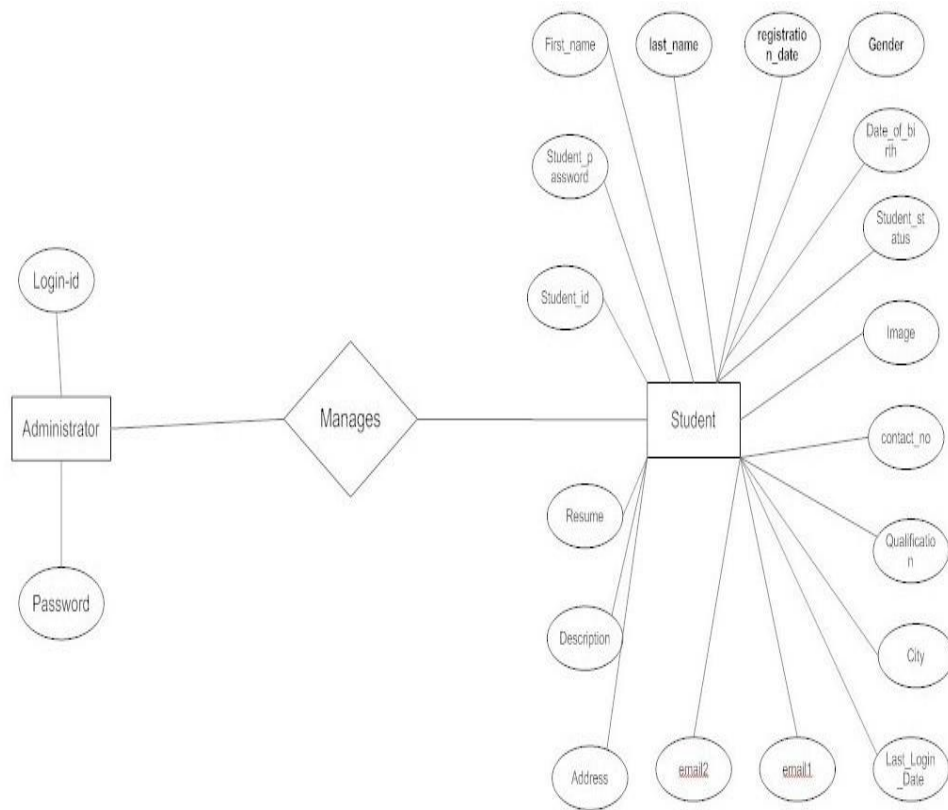


Fig 3.6.1 Entity Relationship Diagram

3.7 Functional Design Description

3.7.1 Data Flow Diagram :

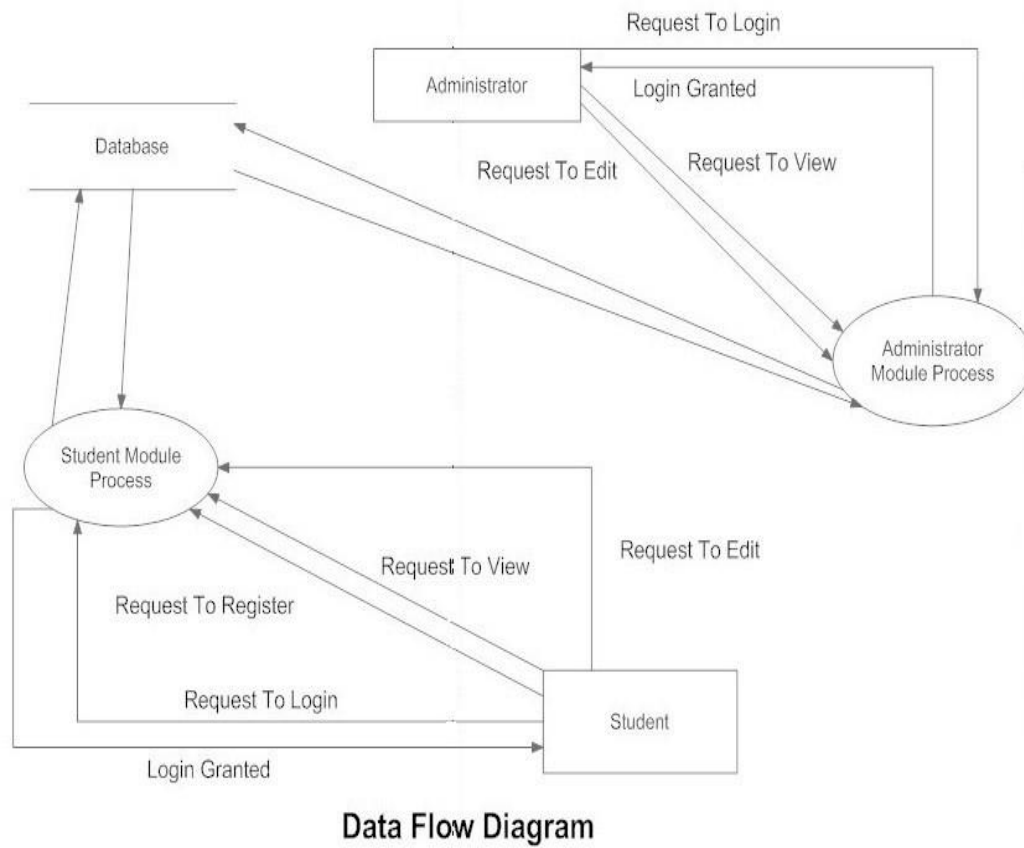


Fig 3.7.1 Data Flow Diagram

3.7.2 Decision Tree :

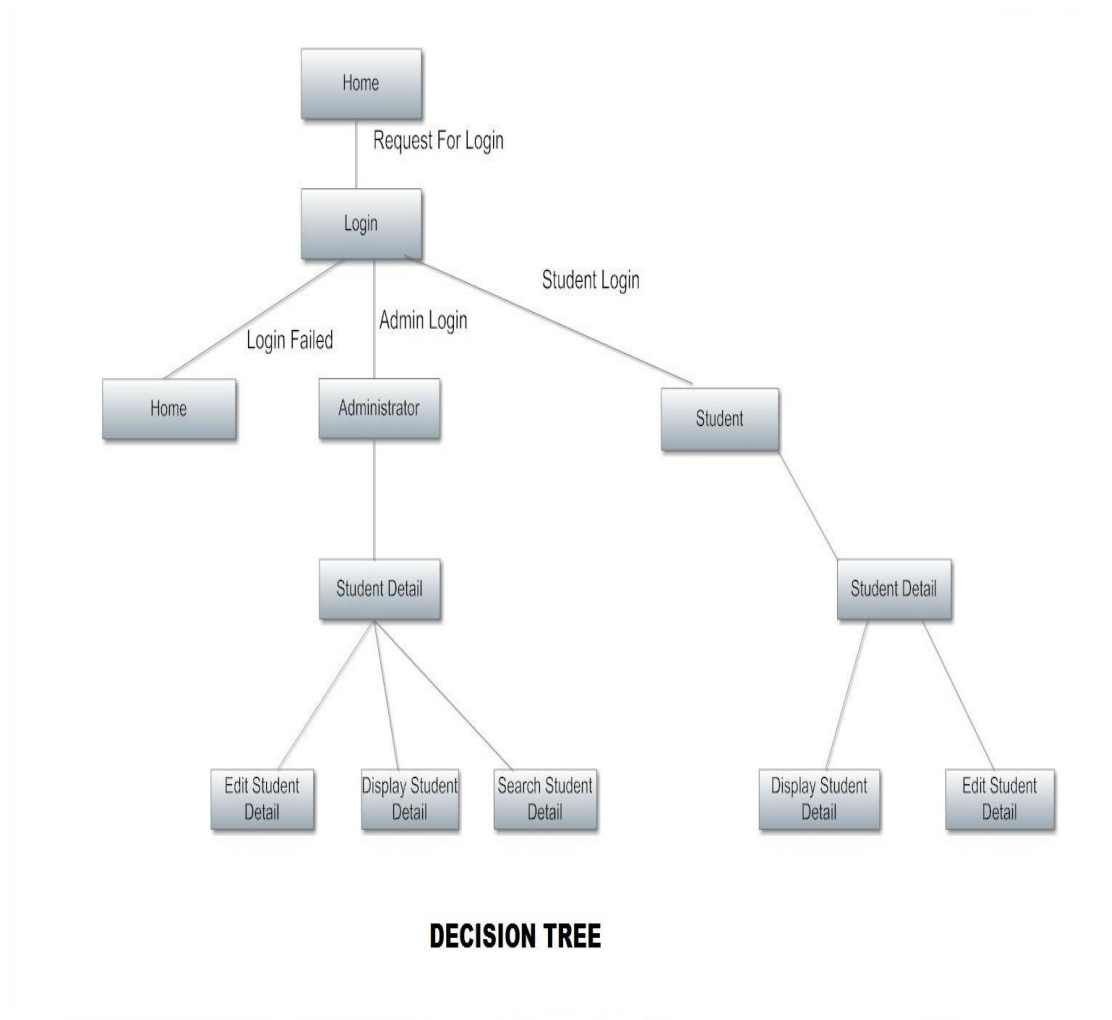


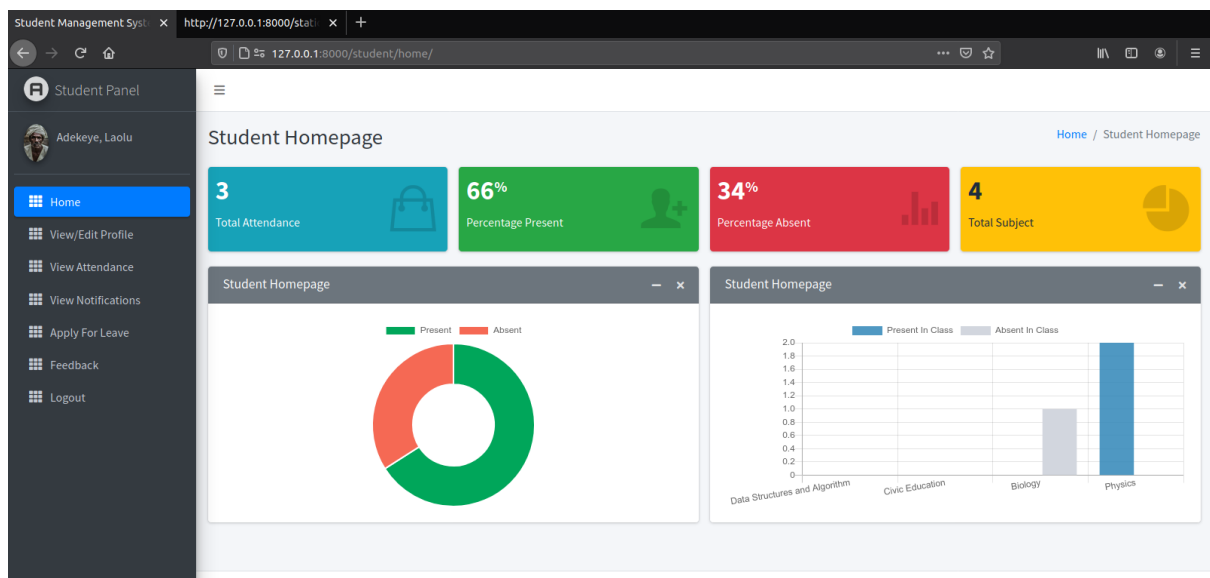
Fig 3.7.2 Decision Tree

CHAPTER – 4

RESULTS AND PERFORMANCE ANALYSIS

4.1 Login Page

This is the homepage/index page of the site.



The student can view their profile. The below page shows the user profile. The link to the resume is shown if a resume was uploaded during registration otherwise its not shown.

The screenshot shows a web browser window with the address bar displaying `http://127.0.0.1:8000/student/view/profile/`. The page title is "View/Edit Profile". On the left, there is a "Student Panel" sidebar with a user profile for "Adekeye, Laolu" and a list of navigation items: Home, View/Edit Profile (highlighted), View Attendance, View Notifications, Apply For Leave, Feedback, and Logout. The main content area has a blue header "View/Edit Profile" and a red error message "Invalid Data Provided". Below this, the form contains the following fields:

- First name:** Text input field containing "Laolu".
- Last name:** Text input field containing "Adekeye".
- Email:** Text input field containing "jobowonubi@gmail.com". Below this field is a red error message: "The given email is already registered".
- Gender:** Dropdown menu with "Male" selected.
- Password:** Text input field with a placeholder text: "Fill this only if you wish to update password".

CHAPTER – 5

SUMMARY AND CONCLUSIONS

The Student Information Management System (SIMS) project typically includes various modules to handle different aspects of student information management, such as student registration, admissions, enrollment, course management, scheduling, grading, and reporting. The system may also include features for communication with parents, teachers, and other stakeholders, as well as tools for generating reports and analytics.

The goal of a SIMS project is to provide an efficient and user-friendly platform for managing student information, which can improve administrative processes, reduce paperwork, and enhance communication between stakeholders. The system can also help institutions make data-driven decisions to improve academic outcomes and student success.

References

- <https://www.powerschool.com/>
- <https://www.infinitecampus.com/>
- <https://www.skyward.com/>
- <https://www.blackbaud.com/>
- <https://edsby.com/>
- <https://www.getalma.com/>
- <https://schooltool.org/>

APPENDIX

WORKING ENVIRONMENT

The technology selected for implementing Student Information Management System is PHP/MYSQL. Apache is used as the HTTP server. The development was done in a 'windows' environment using adobe dreamweaver CS5.

PHP

PHP is a general-purpose scripting language that is especially suited to server-side web development where PHP generally runs on a web server. PHP code is embedded into the HTML source document. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content. It can also be used for command-line scripting and client-side GUI applications. PHP can be deployed on many web servers and operating systems, and can be used with many relational database management systems (RDBMS). It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

MySQL

MySQL is a relational database management system (RDBMS)^[1] that runs as a server providing multi-user access to a number of databases. MySQL is a popular choice of database for use in web applications and is an open source product. The process of setting up a MySQL database varies from host to host, however we will end up with a database name, a user name and a password. Before using our database, we must create a table. A table is a section of the database for storing related information.

In a table we will set up the different fields which will be used in that table. Creating a table in phpMyAdmin is simple, we just type the name, select the number of fields and click the 'go' button. we will then be taken to a setup screen where you must create the fields for the database. Another way of creating databases and tables in phpMyAdmin is by executing simple SQL statements. We have used this method in order to create our database and table.

Apache

The Apache HTTP Server is a web server software notable for playing a key role in the initial growth of the World Wide Web. In 2009 it became the first web server software to surpass the 100 million web site milestone. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Since April 1996 Apache has been the most popular HTTP server software in use. As of November 2010 Apache served over 59.36% of all websites and over 66.56% of the first one million busiest websites.

XAMPP

XAMPP is a small and light Apache distribution containing the most common web development technologies in a single package. Its contents, small size, and portability make it the ideal tool for students developing and testing applications in PHP and MySQL. XAMPP is available as a free download in two specific packages: full and lite. While the full package download provides a wide array of development tools, XAMPP Lite contains the necessary technologies that meet the Ontario Skills Competition standards. The light version is a small package containing Apache HTTP Server, PHP, MySQL, phpMyAdmin, OpenSSL, and SQLite.

Obtaining and Installing XAMPP

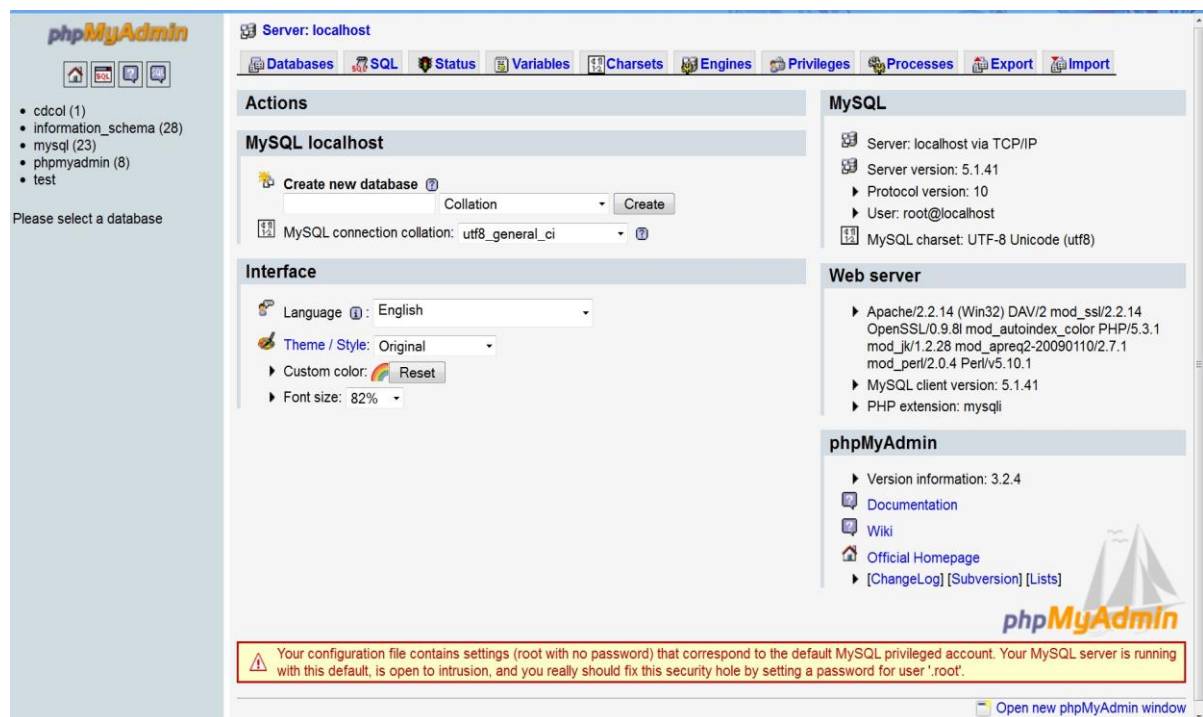
As previously mentioned, XAMPP is a free package available for download and use for various web development tasks. All XAMPP packages and add-ons are distributed through the Apache Friends website at the address: <http://www.apachefriends.org/>. Once on the website, navigate and find the Windows version of XAMPP and download the self-extracting ZIP archive. After downloading the archive, run and extract its contents into the root path of a hard disk or USB drive. For example, the extract path for a local Windows installation would simply be C:\. If extracted properly we will notice a new xampp directory in the root of your installation disk. In order to test that everything has been installed correctly, first start the Apache HTTP Server by navigating to the xampp directory and clicking on the apache_start.bat batch file.



In order to stop all Apache processes we do not close the running terminal application, but instead run another batch file in the xampplite directory called apache_stop.bat.

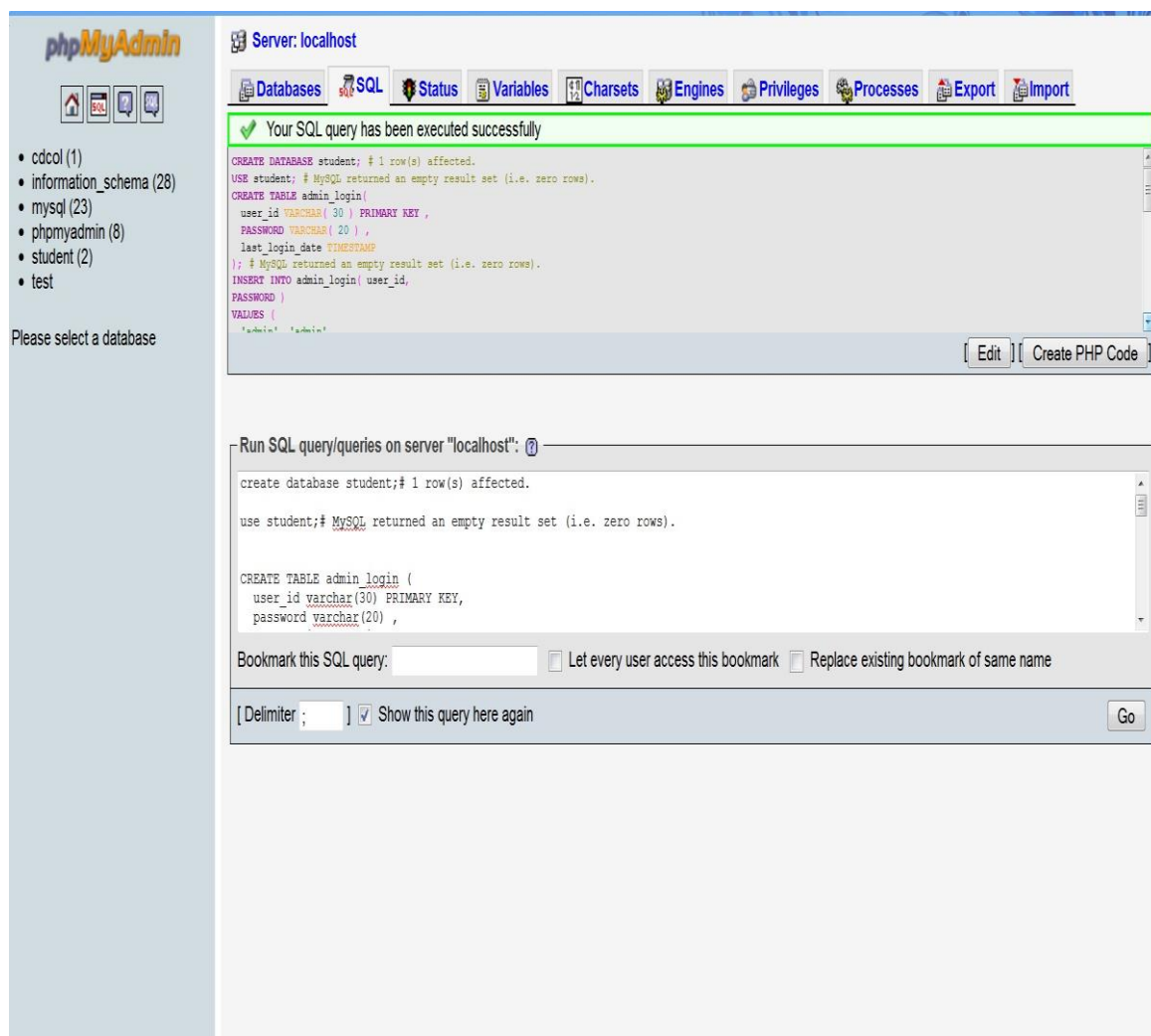
Creating a Database and Inserting Data

Now that we have run and tested Apache and PHP, the next step is running MySQL and creating a database and table which will hold information to be used by our website. In order to start MySQL, navigate to the xampp directory and run the mysql_start.bat batch file. The XAMPP package contains an application called phpMyAdmin which allows developers to administer and maintain MySQL databases. We will be using phpMyAdmin to create a database and table, and enter test data. Before testing phpMyAdmin, make sure that both Apache and MySQL are running by opening their respective batch files: apache_start.bat and mysql_start.bat. Along with Apache and MySQL running in the background, we type http://localhost/phpMyAdmin/ into our web browser. If successful we will be presented with a phpMyAdmin start page similar to the one shown below.



phpMyAdmin start page

The first step with phpMyAdmin running is creating a new database. We create a new database by directly executing SQL statements as shown below. The successful execution of the sql query creates a database 'student' with two tables in it. The tables are admin_login and student_information. We also inserted values in the admin table. The screenshot below shows the successful execution of the query thus creation of a database named student.



Creation of database in mysql using phpMyAdmin

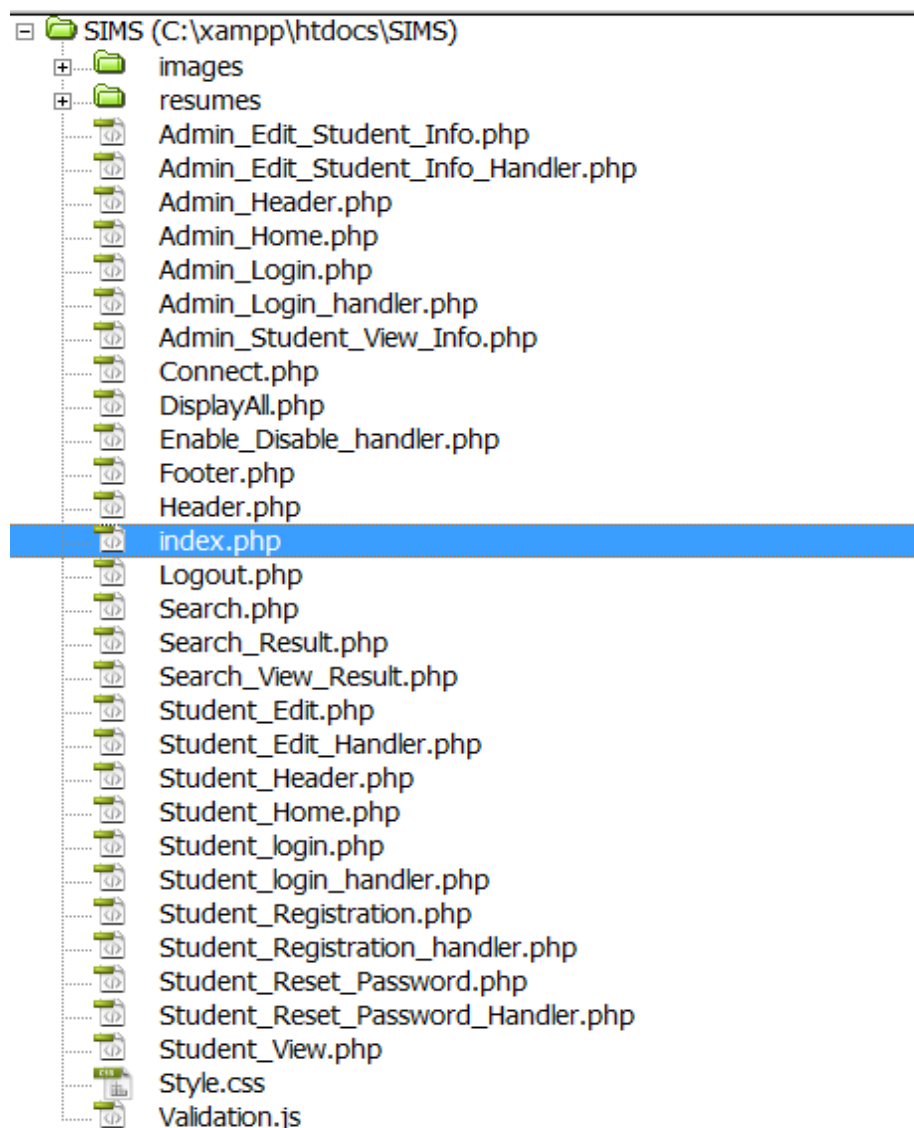
Thus we have learned to create a database in MYSQL by executing sql statements. After creating the database and tables we are now ready to use them in our website “Student Information Management System” .

PROJECT DESCRIPTION

Introduction

Student Information Management System can be used by education institutes 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 this project

The directory structure of the project is as follows:



Description of root directory contents

- **Images Directory** : This directory contains the images uploaded by the students during registration process. Supported formats are the .jpg and .gif files.
- **Resume Directory** : This Directory Contains resumes of students uploaded during registration process of students. Files in this folder can be of .doc, .txt or .pdf format.
- **Admin_Edit_Student_Info.php** : Admin page for editing information of a student. The administrator can change details of a student in this page. Though facility of changing the image and resume are not yet provided but will be provided in future versions of the project.
- **Admin_Edit_Student_Info_Handler.php** : Page handler for handling the Admin_Edit_Student_Info.php file. It writes the edited values in the database on the server.
- **Admin_Header.php** : Header file for pages accessible to administrator only.
- **Admin_Home.p** : Home page for administrator after logging in process.
- **Admin_Login.php** : Login page for administrator access. Shows appropriate message for wrong username and/or password.
- **Admin_Login_handler.php** : Page handler for Admin_Login.php page. It checks the values provided with the values in the database.
- **Admin_Student_View_Info.php** : Page to display student information to the administrator.
- **Connect.php** : Page for database connectivity. It is used whenever database values are required on the page.
- **DisplayAll.php** : Page to display all registered students to the administrator. This facility is only available to the administrator.
- **Enable_Disable_handler.php** : Handler page for enabling/disabling of students account facility. This functionality is only available to the administrator.
- **Footer.php** : Footer file for all pages.
- **Header.php** : Header file for login page and homepage of the site.
- **Index.php** : Homepage of the website.

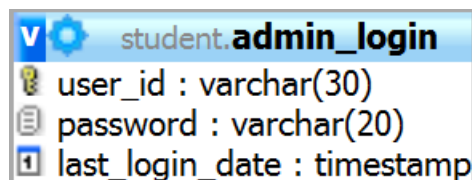
- **Logout.php** : Logout handler page.It Destroys all session variables thusending user session.
- **Search.php** : Search page to search students.It can only be used by administrator.Students can be searched using different fields such as user id,account status etc.
- **Search_Result.php** : Page to display search results to the administrator.
- **Search_View_Result.php** : Page to display student information for anyselected search result.
- **Student_Edit.php** : Page to edit student information.It can be accessed by students.
- **Student_Edit_Handler.php** : Page handler for Student_Edit.php.
- **Student_Header.php** : Header file for student pages.
- **Student_Home.php** : Home page for students after they log into theirrespective accounts.
- **Student_login.php** : Login Page for student login. Appropriate message is displayed if the login is unsuccessful.
- **Student_login_handler.php** : Page handler for Student_Login.php.It checksthe values provided with that in th the database.
- **Student_Registration.php** : Student Registration page.The students enters various details here for registration.
- **Student_Registration_handler.php** : Page Handler for handling the file Student_registration.php.It adds value to the student_information table thus creating a new user.

- **Search.php** : Search page to search students.It can only be used by administrator.Students can be searched using different fields such as user id,account status etc.
- **Search_Result.php** : Page to display search results to the administrator.
- **Search_View_Result.php** : Page to display student information for anyselected search result.
- **Student_Edit.php** : Page to edit student information.It can be accessed by students.
- **Student_Edit_Handler.php** : Page handler for Student_Edit.php.
- **Student_Header.php** : Header file for student pages.
- **Student_Home.php** : Home page for students after they log into theirrespective accounts.
- **Student_login.php** : Login Page for student login. Appropriate message is displayed if the login is unsuccessful.
- **Student_login_handler.php** : Page handler for Student_Login.php.It checksthe values provided with that in th the database.
- **Student_Registration.php** : Student Registration page.The students enters various details here for registration.
- **Student_Registration_handler.php** : Page Handler for handling the file Student_registration.php.It adds value to the student_information table thus creating a new user.

- **Student_View.php** : Page to display student profile with all the details of the student.
- **Style.css** : Stylesheet for the whole site design.
- **Validation.js** : Javascript validations used for validation of form values. Various form entries are validated at the client side using this file only.

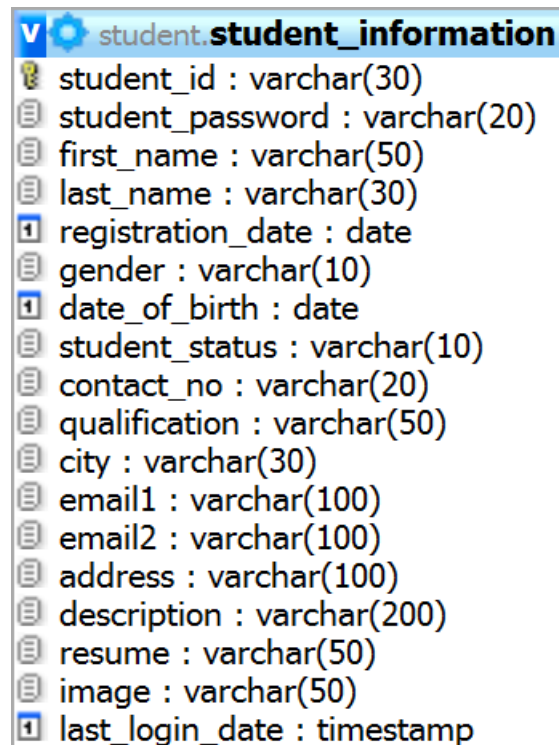
Description of database tables

- admin_login :



- **user_id** : Stores user id of administrator(s).
- **password** : Stores password of the administrator(s).
- **last_login_date** : Stores the last login date of the administrator(s).
- **student_id** : Stores user id of the student(s)
- **student_password** : Stores password of the student(s)
- **first_name** : Stores first name of the student(s)
- **last_name** : Stores last name of the student(s)

- **Student_information :**



student.student_information	
🔑	student_id : varchar(30)
📄	student_password : varchar(20)
📄	first_name : varchar(50)
📄	last_name : varchar(30)
1	registration_date : date
📄	gender : varchar(10)
1	date_of_birth : date
📄	student_status : varchar(10)
📄	contact_no : varchar(20)
📄	qualification : varchar(50)
📄	city : varchar(30)
📄	email1 : varchar(100)
📄	email2 : varchar(100)
📄	address : varchar(100)
📄	description : varchar(200)
📄	resume : varchar(50)
📄	image : varchar(50)
1	last_login_date : timestamp

- student_id : Stores user id of the student(s)
- student_password : Stores password of the student(s)
- first_name : Stores first name of the student(s)
- last_name : Stores last name of the student(s)
- registration_date : Stores the registration date of the student(s).
- gender : Stores the gender of the student(s).
- date_of_birth : Stores the date of birth of the student(s).
- student_status : Stores the current status of the student account(s).

- contact_no : Stores the contact number of the student(s).
- qualification : Stores student(s) qualification.
- city : Stores the city in which the student(s) lives.
- email1 : Stores primary email of the student(s).
- email2 : Stores secondary email of the student(s).
- address : Stores the address of the student(s).
- description : Stores description of the student(s).
- resume : Stores resume of students(s).
- image : Stores image of the student(s).
- last_login_date : Stores last login date of the student(s).

Features

The Website provides following functionalities to the users :

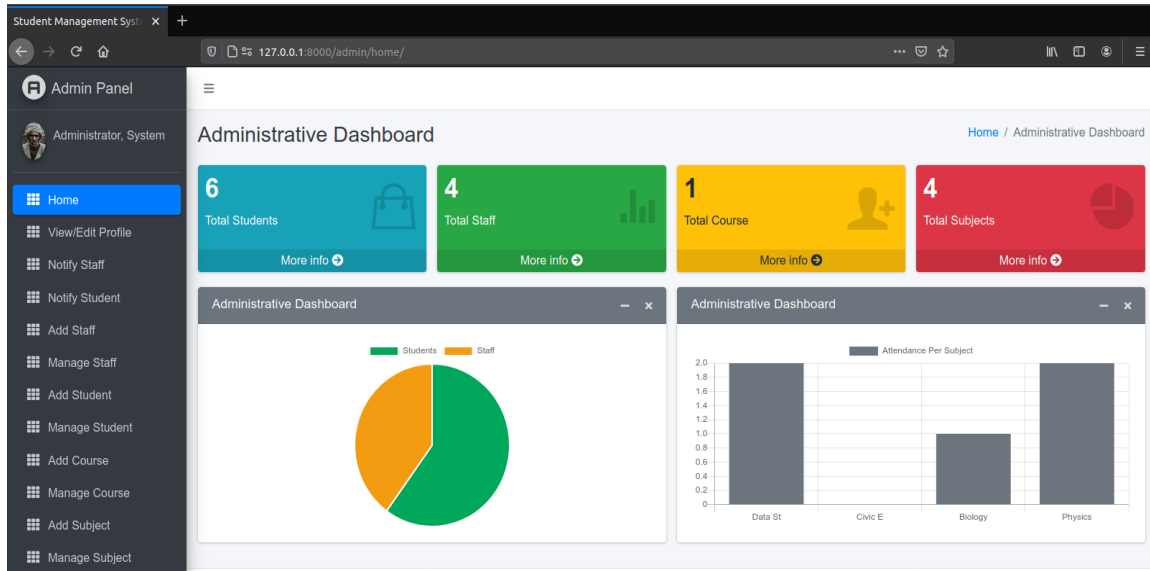
- **Administrator :**
 - Login/Logout
 - View student information
 - Edit Student Information
 - Enable/disable student accounts
 - Search students

- Student :
 - Login/Logout
 - View profile
 - Edit profile
 - Change password
 - Register new profile

SNAPSHOTS

Admin Snapshots

This is the homepage/indexpage of the admin site.



This is the page where staff details can be viewed.

The screenshot shows the 'Staff Feedback Messages' page. The left sidebar is the same as the previous screenshot. The main content area is titled 'Staff Feedback Messages' and contains a table with the following data:

SN	Staff	Course	Message	Sent On	Replied On	Reply
1	Ibrahim Farouq	Computer Science	Hello Admin, how are you today?	Oct. 22, 2020, 8:54 p.m.	Oct. 22, 2020, 8:54 p.m.	I'm good, thanks.

Adding of Students can be done.

The screenshot shows a web browser window with the URL `127.0.0.1:8000/student/add/`. The page is titled "Add Student" and features a sidebar menu on the left with options like "Home", "View/Edit Profile", "Notify Staff", "Notify Student", "Add Staff", "Manage Staff", "Add Student" (highlighted), "Manage Student", "Add Course", "Manage Course", "Add Subject", "Manage Subject", and "Add Session". The main content area contains a form with the following fields: "First name:" (text input), "Last name:" (text input), "Email:" (text input), "Gender:" (dropdown menu with "Male" selected), "Password:" (text input), "Profile pic:" (file upload button labeled "Browse..." and "No file selected."), and "Address:" (text input). A blue header bar at the top of the form area says "Add Student".

The screenshot shows a web browser window with the URL `127.0.0.1:8000/add_session/`. The page is titled "Add Session" and features the same sidebar menu as the previous screenshot. The main content area contains a form with the following fields: "Start year:" (text input with placeholder "dd / mm / yyyy"), "End year:" (text input with placeholder "dd / mm / yyyy"), and a blue button labeled "Add Session". A blue header bar at the top of the form area says "Add Session".

After successful registration the user account needs to be first enabled by the administrator. The students can then login into their accounts with the id and password they choose during registration. The above page shows student login page.

Student Management Syst

127.0.0.1:8000/staff/view/feedback/

Admin Panel

Administrator, System

Home

View/Edit Profile

Notify Staff

Notify Student

Add Staff

Manage Staff

Add Student

Manage Student

Add Course

Manage Course

Add Subject

Manage Subject

Staff Feedback Messages

Home / Staff Feedback Messages

Staff Feedback Messages

SN	Staff	Course	Message	Sent On	Replied On	Reply
1	Ibrahim Farouq	Computer Science	Hello Admin, how are you today?	Oct. 22, 2020, 8:54 p.m.	Pending Response	Reply

Student Management Syst

127.0.0.1:8000/subject/edit/3

Admin Panel

Administrator, System

Home

View/Edit Profile

Notify Staff

Notify Student

Add Staff

Manage Staff

Add Student

Manage Student

Add Course

Manage Course

Add Subject

Manage Subject

Edit Subject

Home / Edit Subject

Edit Subject

Name:

Biology

Staff:

Ibrahim Farouq

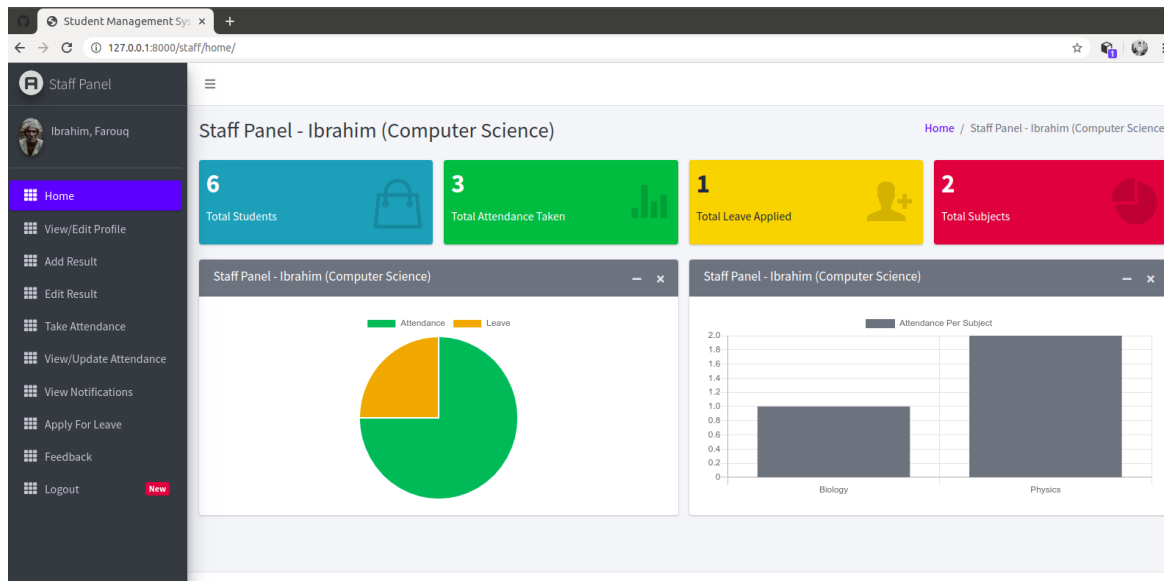
Course:

Computer Science

Edit Subject

The staff can view their profile. The above page shows the user profile. The link to the resume is shown if a resume was uploaded during registration otherwise its not shown.

Staff Snapshots



Staff can view and update their profile by using this.

The screenshot shows the View/Update Profile form. The form includes fields for First name, Last name, Email, Gender, Password, Profile pic, and Address. The form is titled "View/Update Profile" and has a sidebar with navigation options: Home, View/Edit Profile, Add Result, Edit Result, Take Attendance, View/Update Attendance, View Notifications, Apply For Leave, Feedback, and Logout.

Form Fields:

- First name: Farouq
- Last name: Ibrahim
- Email: solo@gmail.com
- Gender: Male
- Password: Fill this only if you wish to update password
- Profile pic: Choose File No file chosen
- Address: Ilorin

Student Management System

127.0.0.1:8000/staff/result/add/

Staff Panel

Ibrahim, Farouq

- Home
- View/Edit Profile
- Add Result
- Edit Result
- Take Attendance
- View/Update Attendance
- View Notifications
- Apply For Leave
- Feedback
- Logout

Result Upload

Home / Result Upload

Result Upload

Subject

Biology

Session Year

From 2019-11-30 to 2020-12-30

Fetch Students

Student List

Owonubi Job

Test Score

Test Score

Exam Score

Exam Score

Save Result

Student Management System

127.0.0.1:8000/staff/result/edit/

Staff Panel

Ibrahim, Farouq

- Home
- View/Edit Profile
- Add Result
- Edit Result
- Take Attendance
- View/Update Attendance
- View Notifications
- Apply For Leave
- Feedback
- Logout

Edit Student's Result

Home / Edit Student's Result

Edit Student's Result

Session Year:

From 2019-11-30 to 2020-12-30

Subject:

Biology

Student:

Akinola Roseline

Test:

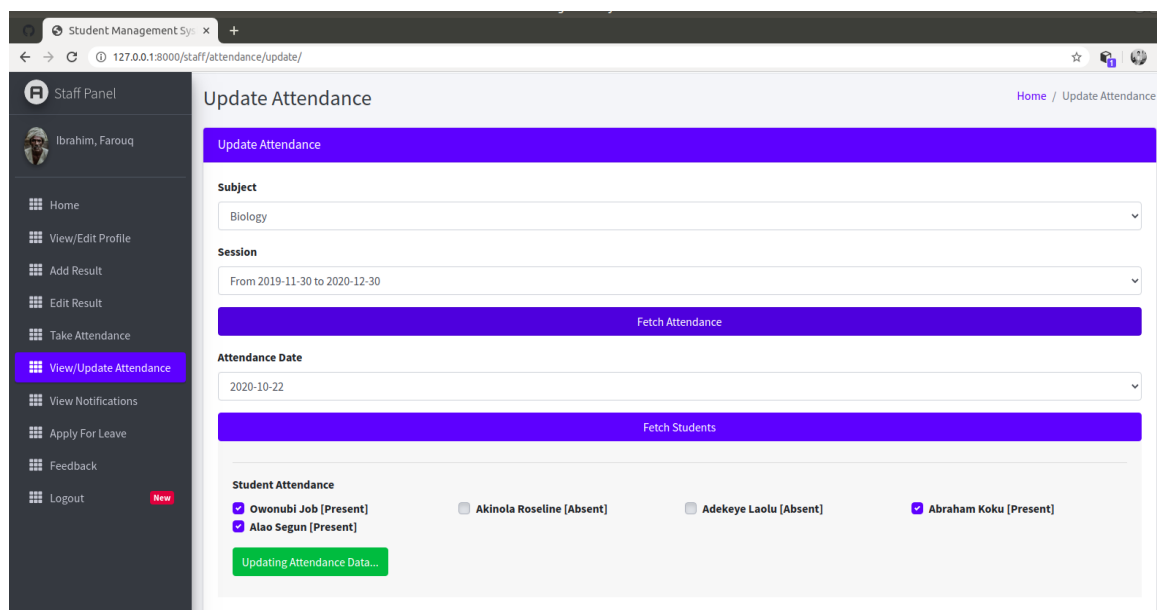
23

Exam:

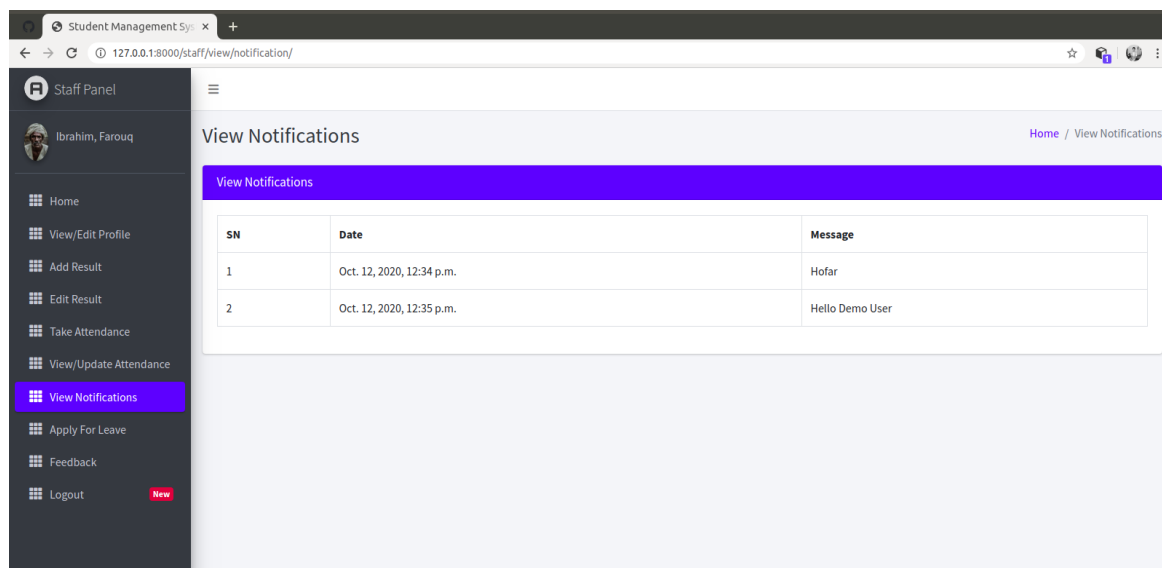
26

Update Result

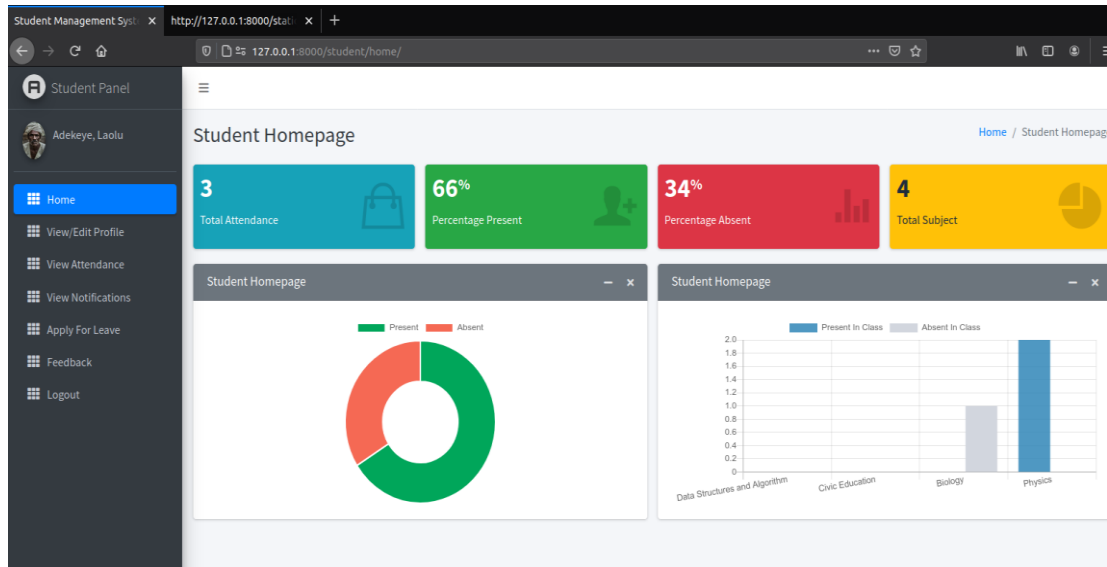
Students can edit their information here.



Attendance and notifications can be seen in this site.



Student Snapshots



The screenshot shows the 'View/Edit Profile' page for a user named Adekeye, Laolu. The page features a sidebar with navigation options: Home, View/Edit Profile, View Attendance, View Notifications, Apply For Leave, Feedback, and Logout. The main content area displays a form for editing the profile. The form includes fields for First name, Last name, Email, Gender, and Password. A red error message 'Invalid Data Provided' is displayed at the top of the form. The form also includes a validation message: 'The given email is already registered'.

Invalid Data Provided

First name:

Last name:

Email:

Gender:

Password:

Student Management System | http://127.0.0.1:8000/stat/

127.0.0.1:8000/student/view/attendance/

Student Panel

Adekeye, Laolu

- Home
- View/Edit Profile
- View Attendance**
- View Notifications
- Apply For Leave
- Feedback
- Logout

View Attendance

[Home](#) / [View Attendance](#)

View Attendance

Select Subject

Physics

Start Date **End Date**

12 / 10 / 2018 30 / 10 / 2020

Fetch Attendance Data

2020-10-22 Present

2020-10-22 Present

Student Management System | http://127.0.0.1:8000/stat/

127.0.0.1:8000/student/apply/leave/

Student Panel

Adekeye, Laolu

- Home
- View/Edit Profile
- View Attendance
- View Notifications
- Apply For Leave**
- Feedback
- Logout

Apply For Leave

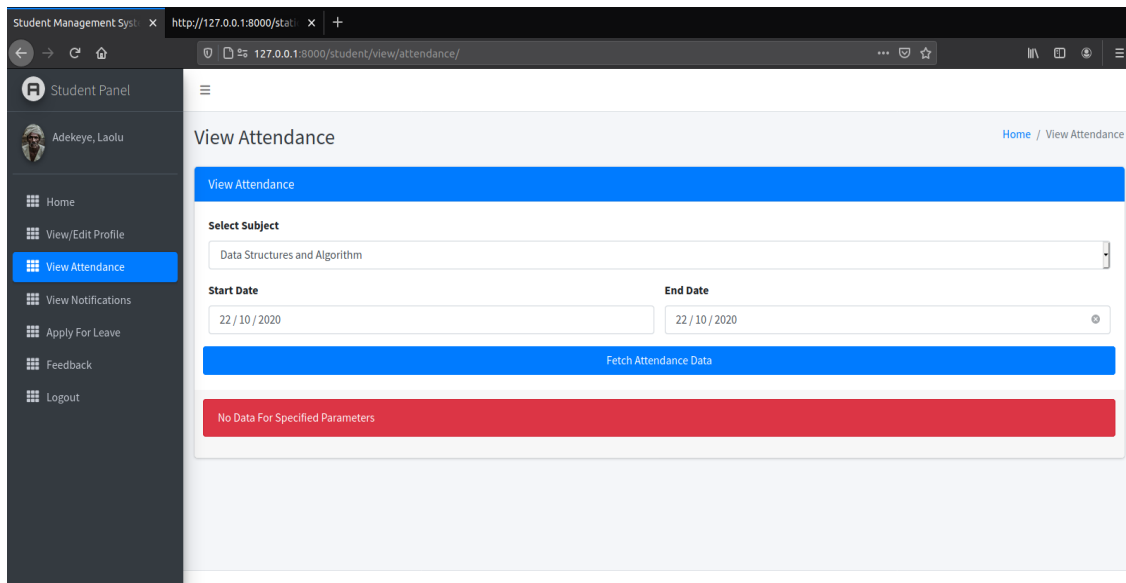
Date: dd / mm / yyyy

Message:

Apply For Leave

Apply for leave

ID	Date	Message	Status
1	2020-10-22	Hello World	Pending



SOURCE CODE

Django

```
"""Django's  
command-  
line utility for  
administrative  
tasks."""
```

```
import os  
import sys
```

```
def main():
```

```
    """Run administrative tasks."""
```

```
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',  
'student_management_system.settings')
```

```
    try:
```

```
        from django.core.management import  
execute_from_command_line
```

```
    except ImportError as exc:
```

```
        raise ImportError(  
            "Couldn't import Django. Are you sure it's installed and "
```

```
            "available on your PYTHONPATH environment variable?  
Did you "
```

```
            "forget to activate a virtual environment?"
```

```
        ) from exc
```

```
        execute_from_command_line(sys.argv)
```

```
if __name__ == '__main__':  
    main()
```

CSS

```
body
```

```
{
```

```
    direction: rtl;
```

```
}
```

```
/* LOGIN */
```

```
.login .form-row {
```

```
    float: right;
```

```
}
```

```
.login .form-row label {
```

```
    float: right;
```

```
    padding-left: 0.5em;
```

```
    padding-right: 0;
```

```
    text-align: left;
```

```
}
```

```
.login .submit-row {
```

```
    clear: both;
```

```
    padding: 1em 9.4em 0 0;
```

```
}
```

```
/* GLOBAL */
```

```
th {  
    text-align: right;  
}
```

```
.module h2, .module caption {  
    text-align: right;  
}
```

```
.module ul, .module ol {  
    margin-left: 0;  
    margin-right: 1.5em;  
}
```

```
.viewlink, .addlink, .changelink {  
    padding-left: 0;  
    padding-right: 16px;  
    background-position: 100% 1px;  
}
```

```
.deletelink {  
    padding-left: 0;  
    padding-right: 16px;  
    background-position: 100% 1px;  
}
```

```
.object-tools {  
    float: left;  
}
```

```
thead th:first-child,  
tfoot td:first-child {  
    border-left: none;  
}
```

```
/* LAYOUT */
```

```
#user-tools {  
    right: auto;  
    left: 0;  
    text-align: left;  
}
```

```
div.breadcrumbs {  
    text-align: right;  
}
```

```
#content-main {  
    float: right;  
}
```

```
#content-related {  
    float: left;  
    margin-left: -300px;  
    margin-right: auto;  
}
```

```
.colMS {  
    margin-left: 300px;  
    margin-right: 0;  
}
```

```
/* SORTABLE TABLES */
```

```
table thead th.sorted .sortoptions {  
    float: left;  
}
```

```
thead th.sorted .text {  
    padding-right: 0;  
    padding-left: 42px;  
}
```

```
/* dashboard styles */
```

```
.dashboard .module table td a {  
    padding-left: .6em;  
    padding-right: 16px;
```

```
}
```

```
/* changelists styles */
```

```
.change-list .filtered table {  
    border-left: none;  
    border-right: 0px none;  
}
```

```
#changelist-filter {  
    right: auto;  
    left: 0;  
    border-left: none;  
    border-right: none;  
}
```

```
.change-list .filtered .results, .change-list .filtered .paginator, .filtered  
#toolbar, .filtered div.xfull {  
    margin-right: 0;  
    margin-left: 280px;  
}
```

```
#changelist-filter li.selected {  
    border-left: none;  
    padding-left: 10px;  
    margin-left: 0;  
    border-right: 5px solid #eaeaea;  
    padding-right: 10px;
```

```
    margin-right: -15px;
}
```

```
.filtered .actions {
    margin-left: 280px;
    margin-right: 0;
}
```

```
#changelist table tbody td:first-child, #changelist table tbody th:first-child {
    border-right: none;
    border-left: none;
}
```

```
/* FORMS */
```

```
.aligned label {
    padding: 0 0 3px 1em;
    float: right;
}
```

```
.submit-row {
    text-align: left
}
```

```
.submit-row p.deletelink-box {
    float: right;
}
```

```
.submit-row input.default {  
    margin-left: 0;  
}
```

```
.vDateField, .vTimeField {  
    margin-left: 2px;  
}
```

```
.aligned .form-row input {  
    margin-left: 5px;  
}
```

```
form .aligned p.help, form .aligned div.help {  
    clear: right;  
}
```

```
form .aligned ul {  
    margin-right: 163px;  
    margin-left: 0;  
}
```

```
form ul.inline li {  
    float: right;  
    padding-right: 0;  
    padding-left: 7px;  
}
```

```
input[type=submit].default, .submit-row input.default {  
    float: left;  
}
```

```
fieldset .fieldBox {  
    float: right;  
    margin-left: 20px;  
    margin-right: 0;  
}
```

```
.errorlist li {  
    background-position: 100% 12px;  
    padding: 0;  
}
```

```
.errornote {  
    background-position: 100% 12px;  
    padding: 10px 12px;  
}
```

```
/* WIDGETS */
```

```
.calendarnav-previous {  
    top: 0;  
    left: auto;  
    right: 10px;
```

```
}
```

```
.calendarnav-next {  
    top: 0;  
    right: auto;  
    left: 10px;  
}
```

```
.calendar caption, .calendarbox h2 {  
    text-align: center;  
}
```

```
.selector {  
    float: right;  
}
```

```
.selector .selector-filter {  
    text-align: right;  
}
```

```
.inline-deletelink {  
    float: left;  
}
```

```
form .form-row p.datetime {  
    overflow: hidden;  
}
```

```
.related-widget-wrapper {  
    float: right;  
}
```

```
/* MISC */
```

```
.inline-related h2, .inline-group h2 {  
    text-align: right  
}
```

```
.inline-related h3 span.delete {  
    padding-right: 20px;  
    padding-left: inherit;  
    left: 10px;  
    right: inherit;  
    float:left;  
}
```

```
.inline-related h3 span.delete label {  
    margin-left: inherit;  
    margin-right: 2px;  
}
```

EditResultview.py

```
from  
django.shortcuts
```

```

import
get_object_or_404,
render, redirect

from django.views import View
from django.contrib import messages
from .models import Subject, Staff, Student, StudentResult
from .forms import EditResultForm
from django.urls import reverse


class EditResultView(View):
    def get(self, request, *args, **kwargs):
        resultForm = EditResultForm()
        staff = get_object_or_404(Staff, admin=request.user)
        resultForm.fields['subject'].queryset =
Subject.objects.filter(staff=staff)
        context = {
            'form': resultForm,
            'page_title': "Edit Student's Result"
        }
        return render(request,
"staff_template/edit_student_result.html", context)


    def post(self, request, *args, **kwargs):
        form = EditResultForm(request.POST)
        context = {'form': form, 'page_title': "Edit Student's
Result"}
        if form.is_valid():
            try:
                student = form.cleaned_data.get('student')

```

```

        subject = form.cleaned_data.get('subject')
        test = form.cleaned_data.get('test')
        exam = form.cleaned_data.get('exam')
        # Validating
        result =
StudentResult.objects.get(student=student,
subject=subject)
        result.exam = exam
        result.test = test
        result.save()
        messages.success(request, "Result Updated")
        return redirect(reverse('edit_student_result'))
    except Exception as e:
        messages.warning(request, "Result Could Not Be
Updated")
    else:
        messages.warning(request, "Result Could Not Be
Updated")
        return render(request,
"staff_template/edit_student_result.html", context)

```

EmailBackend.py

```

from
django.contrib.auth.backends
import ModelBackend

from django.contrib.auth import get_user_model

```

```

class EmailBackend(ModelBackend):
    def authenticate(self, username=None,
password=None, **kwargs):
        UserModel = get_user_model()

```

```
        try:
            user =
        UserModel.objects.get(email=username)
        except UserModel.DoesNotExist:
            return None
        else:
            if user.check_password(password):
                return user
        return None
```

Admin.py

```
from
django.contrib
import admin

from django.contrib.auth.admin import UserAdmin
from .models import *
# Register your models here.

class UserModel(UserAdmin):
    ordering = ('email',)

admin.site.register(CustomUser, UserModel)
admin.site.register(Staff)
admin.site.register(Student)
admin.site.register(Course)
admin.site.register(Subject)
```

Java Script

```
'use
strict';

{
  const SelectBox = {
    cache: {},
    init: function(id) {
      const box = document.getElementById(id);
      SelectBox.cache[id] = [];
      const cache = SelectBox.cache[id];
      for (const node of box.options) {
        cache.push({value: node.value, text: node.text, displayed: 1});
      }
    },
    redisplay: function(id) {
      // Repopulate HTML select box from cache
      const box = document.getElementById(id);
      box.innerHTML = "";
      for (const node of SelectBox.cache[id]) {
        if (node.displayed) {
          const new_option = new Option(node.text, node.value, false,
false);

          // Shows a tooltip when hovering over the option
          new_option.title = node.text;
          box.appendChild(new_option);
        }
      }
    },
    filter: function(id, text) {
      // Redisplay the HTML select box, displaying only the choices
containing ALL
```

```

// the words in text. (It's an AND search.)
const tokens = text.toLowerCase().split(/\s+/);
for (const node of SelectBox.cache[id]) {
  node.displayed = 1;
  const node_text = node.text.toLowerCase();
  for (const token of tokens) {
    if (node_text.indexOf(token) === -1) {
      node.displayed = 0;
      break; // Once the first token isn't found we're done
    }
  }
}
SelectBox.redisplay(id);
},
delete_from_cache: function(id, value) {
  let delete_index = null;
  const cache = SelectBox.cache[id];
  for (const [i, node] of cache.entries()) {
    if (node.value === value) {
      delete_index = i;
      break;
    }
  }
  cache.splice(delete_index, 1);
},
add_to_cache: function(id, option) {
  SelectBox.cache[id].push({value: option.value, text: option.text,
displayed: 1});
},
cache_contains: function(id, value) {
  // Check if an item is contained in the cache
  for (const node of SelectBox.cache[id]) {
    if (node.value === value) {

```

```

        return true;
    }
}
return false;
},
move: function(from, to) {
    const from_box = document.getElementById(from);
    for (const option of from_box.options) {
        const option_value = option.value;
        if (option.selected && SelectBox.cache_contains(from,
option_value)) {
            SelectBox.add_to_cache(to, {value: option_value, text:
option.text, displayed: 1});
            SelectBox.delete_from_cache(from, option_value);
        }
    }
    SelectBox.redisplay(from);
    SelectBox.redisplay(to);
},
move_all: function(from, to) {
    const from_box = document.getElementById(from);
    for (const option of from_box.options) {
        const option_value = option.value;
        if (SelectBox.cache_contains(from, option_value)) {
            SelectBox.add_to_cache(to, {value: option_value, text:
option.text, displayed: 1});
            SelectBox.delete_from_cache(from, option_value);
        }
    }
    SelectBox.redisplay(from);
    SelectBox.redisplay(to);
},
sort: function(id) {

```

```
SelectBox.cache[id].sort(function(a, b) {
    a = a.text.toLowerCase();
    b = b.text.toLowerCase();
    if (a > b) {
        return 1;
    }
    if (a < b) {
        return -1;
    }
    return 0;
});
},
select_all: function(id) {
    const box = document.getElementById(id);
    for (const option of box.options) {
        option.selected = true;
    }
}
};
window.SelectBox = SelectBox;
}
```