A PROJECT REPORT
ON
"EDUMANAGE"

A Dissertation Report
Submitted in fulfillment of the requirements for the award of the degree of

"BACHELOR OF COMPUTER APPLICATION"

UNDER
BENGALURU NORTH UNIVERSITY, KARNATAKA

FOR THE ACADEMIC YEAR 2023-2024

SUBMITTED BY:

NAME: AJMAL BASHEER (U19CU21S0078)
& ASISH JOSE (U19CU21S0063)

UNDER THE GUIDANCE OF
PROF. PRAKASAM M

Krupanidhi Degree College
Department of Computer Science

Krupanidhi Degree College
12/1, ChikkaBellandur, Carmelaram Post
VarthurHobli, Off Sarjapur Road,
Bengaluru,Karnataka 560035

KRUPANIDHI DEGREE COLLEGE

NO: 12/1, CHIKKABELLANDUR, CARMELARAM POST
VARTHUR (H), BENGALURU-35
(RECOGNISED BY THE BENGALURU NORTH UNIVERSITY)

CERTIFICATE

This is to certify that major project work entitled **"EduManage"** is a bonafide work carried out by **AJMAL BASHEER** bearing **Reg.No. U19CU21S0078** and **ASISH JOSE** bearing **Reg.No. U1CU21S0063 in** fulfillment for the award of **Bachelor of Computer Applications of Bengaluru North University**, during the year 2023-2024. The project report has been approved as it satisfies the academic requirements in respect of Project Work prescribed for the said Degree.

Prof. Kavitha H S
(HOD)

Dr. Rajendra Prasad
(PRINCIPAL)

EXAMINERS: **DATE OF EXAMINATION**

1)…… **……**

2)……

`

DECLARATION

I hereby declare that **"EduManage"** is the result of the project work carried out by us under the guidance of **Mr. Prakasam M**, Assistant Professor in fulfilment for the award of Bachelor of Computer Applications of Bengaluru North University.

I also declare this project is the outcome of my own efforts and that it has not been submitted to any other University or Institution for the award of any other degree or diploma or certificate.

Place: Bengaluru

Date: Name: AJMAL BASHEER

& ASISH JOSE

CERTIFICATE

This is to certify that the Project report titled "**EduManage**" is an original work of **AJMAL BASHEER** bearing University **Reg no. U19CU21S0078** and **ASISH JOSE** bearing University **Reg no. U19CU21S0063** is being submitted in fulfilment for the award of Bachelor of Computer Applications of Bengaluru North University.

Place: Bengaluru

Date: Guide Signature

ACKNOWLEDGEMENT

I would like to thank all those who have guided and encouraged me to do what has been done thus far. I avail the opportunity to express our deep sense of gratitude and sincere thanks to our department of Bachelor of Computer Applications.

I am deeply grateful to **Prof. Dr. Suresh Nagpal,** Chairman of the KRUPANIDHI GROUP OF INSTITUTIONS. I acknowledge my delightful thanks to **Ms. Geetha Nagpal**, Vice Chair Person, KRUPANIDHI GROUP OF INSTITUTIONS.

I would like to sincerely thank our dean, **Prof. PM. Shyjan, Dr. Rajendra Prasad,** Principal, KRUPANIDHI DEGREE COLLEGE for providing the facilities.

I sincerely thank **Prof. Kavitha H S**, Head of the Department, for her assistance and collaboration during the duration of the project. I express deep gratitude to my guide **Mr. Prakasam M** for his valuable advice, timely suggestions and assistance in the realization of this major-project. My heartfelt thanks to all the staff members and non-teaching staff of the Computer Science department for providing the necessary facilities and support. Without the support of any one of them, this project would not have been a reality.

I am also grateful to my friends for their valuable suggestions for the success of the project. I also thank my parents for their support in all our activities, without which I would not have been able to carry out the project.

Place: Bengaluru

Date: Name: AJMAL BASHEER

& ASISH JOSE

ABSTRACT

This Django-based Student Information Management System “EduManage” is designed to streamline the organization and management of student data within educational institutions. The system centralizes student records, facilitating easy access and management for administrators and educators. Key features include enrollment processing, academic performance tracking, attendance recording, and schedule management. Built with Django's robust security framework, the system ensures data integrity and privacy. This project aims to enhance educational administration efficiency, improve data-driven decision-making, and foster a more connected school community.

The system provides an intuitive user interface for administrators, teachers, and students, facilitating efficient data entry and retrieval processes. Administrators can manage student and teacher enrollments, update records, and generate reports with ease. Teachers can record grades, track attendance, and generate reports. Students can access their academic progress, attendance records, and teachers assessment marks.

The implementation of this “EduManage” System aims to reduce administrative workload, enhance data accuracy, and improve communication within educational institutions. By leveraging Django's capabilities, the system ensures a reliable and efficient platform for managing student information, ultimately contributing to better educational outcomes, and streamlined administrative processes.

CONTENTS

1. INTRODUCTION

In the rapidly evolving landscape of educational technology, managing student information efficiently has become paramount for educational institutions. This project introduces a comprehensive Student Information Management System "EduManage" developed as a web application using the Django framework. The system is designed to streamline the management of student data and enhance communication between administrators, teachers, and students.

"EduManage" is structured into three distinct modules: Admin, Teacher, and Student, each with specific functionalities tailored to the needs of the users. The admin module serves as the backbone of the system, providing comprehensive control over the entire application. Administrators can enroll users, categorizing them as either students or teachers. Additionally, they have the capability to add and manage departments, courses, and assign timetables. The admin module ensures that the institution's data is accurately organized and easily accessible. The Teacher module empowers educators with tools to manage their classes effectively. Teachers can enter attendance records, add marks, and generate reports for their students. This module is designed to reduce the administrative burden on teachers, allowing them to focus more on teaching and less on paperwork. The ability to generate reports provides valuable insights into student performance, aiding in academic planning and intervention. The student module provides students with a user-friendly interface to access their academic information. Students can view their attendance records and marks, enabling them to stay informed about their academic progress. This module promotes transparency and encourages students to take an active role in their education.

By leveraging the power of the Django framework, this web application ensures scalability, security, and ease of maintenance. The system's responsive design makes it accessible from a variety of devices, including desktops, tablets, and smartphones, catering to the needs of a diverse user base.

The implementation of this "EduManage" System aims to reduce administrative workload, enhance data accuracy, and improve communication within educational institutions. By providing a reliable and efficient platform for managing student information, this project contributes to better educational outcomes and streamlined administrative processes.

2. MODULES

- Administrator
- Teacher
- Student

1. Administrator

1.1 Login

Administrator can directly login using their username and password into Administration dashboard.

1.2 Add User

Administrator can enroll users with a unique username and password. Admins can view and edit user information later.

1.3 Add Student

Administrator can assign users as student from add student page. Admin can assign their class also. Student information such as student name, sex, university registration number, date of birth can be added.

1.4 Add Teacher

Administrator can assign users as teacher from add teacher page. Admin can assign their department also. Personal information such as name, sex, id number, date of birth can be added.

1.5 Add Department

Administrator can view and create new departments with unique id and department name. Admin can edit the department details later.

1.6 Add Class

Administrator can create new classes with unique id and assigning them into a department. Admin can also specify section and semester for the class. Admin can add students to departments also. Admins have the access to edit information about class later also.

1.7 Add Courses

Administrator can create new courses by assigning them into departments using unique id's. Admins can specify Subject name and short name

1.8 Attendance

Administrator should have to add start date and end date for attendance. Then only teachers can mark attendance for students.

1.9 Assigns

Administrator should assign timetable to the teachers and students by choosing class id, course, and teacher. Admin can assign time slots (periods) to teacher. It then reflects as timetable for teacher and student.

2. Teacher

2.1 Login

Each teacher in the college is assigned a unique username and password by the administrator. The username is their teacher ID and the same for password. The teacher may change the password later.

2.2 Homepage

After successful login, the student is presented a homepage with their main sections, attendance, marks, timetable, and reports. In the attendance section, the teacher can

enter the attendance of their respective students for the days on which classes were conducted. There is a provision to enter extra classes and view/edit the attendance of each individual student. In the marks section, the teacher may enter the marks for 3 internals, 2 events and 1 SEE for each student. They can also edit each of the entered marks. The timetable provides the classes assigned to the teacher with the day and timings in a tabular form. Lastly, the teacher can generate reports for each of their assigned class.

2.3 Attendance

There is a list of all the class assigned to teacher. So, for each class there are 3 actions available. They are:

2.4 Enter Attendance

On this page, the classes scheduled or conducted is listed in the form of a list. Initially, all the scheduled classes will be listed from the start of the semester to the current date. Thus, if there is class scheduled for today, it will automatically appear on top of the list. If the attendance of any day is not marked it will be red, otherwise green if marked. Classes can also be cancelled which will make that date as yellow. While entering the attendance, the list of students in that class is listed and there are two options next to each. These options are in the form of a radio button for present and absent. All the buttons are initially marked as present and the teacher just needs to change for the absent students.

2.5 Edit Attendance

After entering attendance, the teacher can also edit it. It is like screen for entering attendance, only the entered attendance is saved and display. The teacher can change the appropriate attendance and save it.

2.6 Extra Class

If a teacher has taken a class other than at the scheduled timings, they may enter the attendance for that as well. While entering the extra class, the teacher just needs to specify the date it was conducted and enter the attendance of each of the students. After submitting extra class, it will appear in the list of conducted classes and thus, it can be edited.

2.7 Student Attendance

For each assigned class, the teacher can view the attendance status of the list of students. The number of attended classes, total number of classes conducted and the attendance percentage is displayed. If the attendance percentage of any of the students is below 75, it will be displayed in red. Thus, the teacher may easily find the list of students not eligible to take a test.

2.8 Student Attendance Details

The teacher can view the attendance detail of all their assigned students individually. That is, for all the conducted classes, it will display whether that student was present or absent. The teacher can also edit the attendance of each student individually by changing the attendance status for each conducted class.

2.9 Marks

On this page, the list of classes assigned to the teacher are displayed along with two actions for each class. These actions are:

2.10 Enter Marks

On this page, the teacher can enter the marks for 3 internal assessments, 2 events and one semester end exam. Initially all of them are marked red to denote that the marks have not been entered yet. Once the marks for a test is entered, it turns green. While entering the marks for a particular test, the list of students in that class is listed and marks can be entered for all of them and submitted. Once, the marks are submitted,

the students can view their respective marks. In case if there is a need to change the marks of any student, it is possible to edit the marks.

2.11 Edit Marks

Marks for a test can be edited. While editing, the list of students in that class is dis played along with already entered marks. The marks to be updated can be changed and submitted. The students can view this change immediately.

2.12 Student Marks

For each assigned class, the teacher has access to the list of students and the marks they obtained in all the tests. This is displayed in a tabular form.

2.13 Timetable

This page is a table which lists the day and timings of each of the classes assigned to the teacher. The row headers are the days of the week and the column headers are the time slots. So, for each day, it specifies the classes in the time slots. The timetable is generated automatically from the assign table, which is a table containing the information of all the teachers assigned to a class with a course and the timings the classes.

2.14 Reports

The last page for the teachers is used to generate reports for each class. The report specifies the list of students in that class and their respective CIE and attendance percentage. CIE is the average of the marks obtained from the tests, 3 internals and 2 events. The CIE is out of 50 and the students with CIE below 25 are marked in red and are not eligible to write the semester end exam. Also, the attendance percentage is displayed with students below 75% marked in red.

3. Students

3.1 Login

Each student in the college is assigned a unique username and password by the administrator. The username is the same as their USN and so is the password. They may change it later according to their wish.

3.2 Homepage

After successful login, the student is presented a homepage with their main sections, attendance, marks, and timetable. In the attendance section the student can view their attendance status which includes the total classes, attended classes and the attendance percentage for each of their courses. In the marks section, the student can view the marks for each of their courses out of 20 for 3 internal assessments, 2 events. Also, the semester end examination for 100 marks. Lastly, the timetable provides the classes assigned to that student and day and time of each in a tabular form.

3.3 Attendance

On the attendance page, there is a list of courses that is dependent on each student. For each course, the course id and name are display along with the attended classes, total classes, and the attendance percentage for that course. If the attendance per centage is below 75 for any course, it is displayed in red denoting shortage of attendance, otherwise it is green. If there is any shortage, it specifies the number of classes to attend to make up for it. If you click on each course, it takes you to the attendance detail page.

3.4 Attendance Detail

This page displays more details for the attendance in each course. For each the course, there is a list of classes conducted and each is marked with the date, day and

whether the student was present or absent on that particular date.

3.5 Marks

The Marks page is a table with an entry for each of their courses. The course id and names are specified along the marks obtained in each of the tests and exams. The tests include 3 internal assessments with marks obtained out of a total of 20, 2 events such as project, assignment, quiz etc., with marks out of 20. Lastly, one semester end exam with marks out of 100.

3.6 Timetable

This page is a table which lists the day and timings of each of the classes assigned to the student. The row headers are the days of the week and the column headers are the time slots. So, for each day, it specifies the classes in the time slots. The timetable is generated automatically from the assign table, which is a table containing the information of all the teachers assigned to a class with a course and the timings the classes.

3.SYSTEM REQUIREMENT SPECIFICATION

Software Requirements Specifications (SRS) is a document that describes what the software will do and how it will be expected to perform. A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide.

Software Requirement

Name	Details
Operating System	Ubuntu 22.04
Database Server	PostgreSQL database
Front End	HTML, CSS, Bootstrap, JS, J Query
Back End	Django
Framework	Django REST Framework
Application Server	WSGI
Browser	Google Chrome, Opera, Firefox, Brave

HARDWARE REQUIREMENT

Name	Details
Processor	Intel Core i3 – 8145U
Random-access memory	16.00 GB
SSD	512 GB
Network	Local Area Network

PROCESSOR:

A processor (CPU) is the logic circuitry that responds to and processes the basic instructions that drive a computer. The CPU is seen as the main and most crucial IC chip in a computer, as it is responsible for interpreting most of computer's commands. Intel Core i3 – 8145U CPU @ 2.10GHz 2-core processor is used in the development of this project.

RANDOM ACCESS MEMORY:

Random-access memory is a form of computer memory that can be read and changed in any order, typically used to store working data and machine code. Project was built with PC of memory space of 16 GB and total usage of RAM was about 6.3 GB.

HARD DRIVE:

A hard disk drive, hard disk, hard drive, or fixed disk is an electro-mechanical data storage device that stores and retrieves digital data using magnetic storage with one or more rigid rapidly rotating platters coated with magnetic material.

OPERATING SYSTEM:

An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs. It is the program that, after being initially loaded into the computer by a boot program, manages all the other application programs in a computer. The application programs make use of the operating system by making requests for services through a defined application program interface (API).

DJANGO:

Django is a high-level Python web framework that promotes rapid development and clean, pragmatic design. It's designed to help developers take their applications from concept to completion as quickly as possible. Django takes security seriously, helping developers avoid many common security mistakes. It's also exceedingly scalable, enabling some of the busiest sites on the web to leverage its ability to quickly and flexibly scale. For those new to Django, there are numerous resources available, including detailed documentation and tutorials which guide you through creating basic applications like a polling app. With Django, developers can define data models in Python, enjoy a rich database-access API, and create clean, elegant URL schemes without the need to write excessive code.

MySQL Database:

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) for accessing and managing data. Known for its reliability, scalability, and ease of use, MySQL is widely used for web applications, data warehousing, and logging applications. It supports various storage engines, transaction processing, and comprehensive security features, making it a popular choice for developers and organizations requiring robust database solutions. MySQL's flexibility allows it to be integrated with a wide range of platforms and programming languages, supporting both small-scale projects and large, complex enterprise environments.

PostgreSQL:

PostgreSQL is a highly respected open-source object-relational database system, renowned for its robustness, feature richness, and performance. With over 35 years of development, PostgreSQL has a strong reputation for reliability and is used by a variety of enterprises and organizations worldwide. The latest version, PostgreSQL 17 Beta 2, offers a preview of upcoming features and improvements, and while it's not recommended for production environments yet, it's available for testing and feedback. For those still using PostgreSQL 12, it's important to note that it will stop receiving fixes after November 14, 2024, so planning an upgrade to a newer version is advisable. For new users, there's a wealth of resources and community support available to help get started with PostgreSQL.

HTML:

HTML (Hypertext Markup Language) is a foundational language for creating and structuring content on the web. It uses a system of tags to define the elements and layout of web pages, encompassing everything from text and images to links and

multimedia. HTML plays a crucial role in establishing the structure and semantic meaning of web documents, making them accessible and understandable to both browsers and developers.

CSS:

Cascading Style Sheets (CSS) are a collection of rules that define the appearance and layout of web pages, allowing designers to efficiently control their look. CSS separates content structure (handled by HTML) from presentation, enabling global style changes by updating a single file. For example, altering the font style in the CSS will apply the change across all pages using that style sheet, unlike HTML, where each instance must be updated individually. CSS files, which have a .css extension, can be created using various tools and should be planned out in advance to ensure consistent and organized styling across a website.

JavaScript:

JavaScript is a programming language commonly used in web development. It was originally developed by Netscape as a means to add dynamic and interactive elements to websites. While JavaScript is influenced by Java, the syntax is more similar to C and is based on ECMAScript, a scripting language developed by Sun Microsystems. JavaScript is a client-side scripting language, which means the source code is processed by the client's web browser rather than on the web server. This means JavaScript functions can run after a webpage has loaded without COMMUNICATING with the server. For example, a JavaScript function may check a web form before it is submitted to make sure all the required fields have been filled out. The JavaScript code can produce an error message before any information is actually transmitted to the server. Like server-side scripting languages, such as PHP and ASP, JavaScript code can be inserted anywhere within the HTML of a webpage. However, only the output of server-side code is displayed in the HTML, while

JavaScript code remains fully visible in the source of the webpage. It can also be referenced in a separate .JS file, which may also be viewed in a browser.

REST framework:

The Django REST Framework (DRF) is a powerful and flexible toolkit for building Web APIs in Django. It simplifies the creation of RESTful APIs by providing a comprehensive set of tools and abstractions, including serializers for data validation and transformation, viewsets for handling different HTTP methods, and a robust authentication and permissions system. DRF supports a wide range of features like pagination, filtering, and content negotiation, making it easier to develop APIs that are both scalable and maintainable. Its modular architecture and extensive documentation make it an ideal choice for developers looking to expose their Django models and business logic through a clean, well-structured API.

uWSGI:

uWSGI is a versatile application server for deploying web applications, particularly those written in Python. It is often used in conjunction with web frameworks like Django and Flask. uWSGI is designed to run applications in a production environment, offering high performance and scalability. It supports multiple protocols, including HTTP, HTTPS, and its own uWSGI protocol, and can handle various concurrency models such as multithreading and multiprocessing. One of its key features is the ability to manage application processes, which can be configured to automatically restart in case of failure, ensuring high availability. Additionally, uWSGI includes robust management tools, such as a monitoring system and support for dynamic reloading of applications. It is frequently used behind a web server like Nginx, which handles client connections and forwards requests to uWSGI, providing an efficient and secure deployment setup.

4. SYSTEM ANALYSIS

PROJECT SCOPE:

College management is becoming increasingly essential in modern education. A College Automation System enables the efficient gathering of all necessary information for management with just a few clicks. The College Management system automates the details that were previously maintained manually, streamlining the entire process. Once data is entered into the system, there is no need for multiple individuals to manage separate sections; a single person can handle all reports and records. This centralization of data management significantly reduces labor and the likelihood of errors. Additionally, the system allows for customizable security settings, ensuring that sensitive information is accessible only to authorized users. This not only enhances the efficiency and accuracy of managing college operations but also ensures that data is secure. By implementing a College Automation System, educational institutions can improve their administrative processes, making them more efficient, secure, and responsive to the needs of students and staff.

PROPOSED SYSTEM:

The proposed college management system aims to develop an advanced platform with enhanced features to streamline administrative processes and improve operational efficiency within educational institutions. Building upon existing management tools, the system will introduce new functionalities such as centralized student and faculty record management, automated attendance tracking, and integration with learning management systems (LMS). It will also incorporate analytics capabilities to provide insights into student performance and administrative trends. The user interface will be redesigned for intuitive navigation, and accessibility features will be enhanced to cater to diverse user needs. Cloud-based

deployment will ensure scalability and easy access across multiple campuses and devices, facilitating seamless collaboration and information sharing among stakeholders. This evolution of the college management system seeks to offer a more integrated, efficient, and user-friendly platform, meeting the evolving needs of modern educational environments.

PURPOSE:

The purpose of this project is to design a comprehensive software system for managing a college database that maintains up-to-date and accurate information about the institution. This system aims to improve the efficiency and flexibility of college record management and to provide a unified, user-friendly platform for accessing student information. The College Automation System will consist of several interconnected modules, including student, faculty, and admin modules. Each module will be designed to handle specific tasks and responsibilities, streamlining the workflow, and reducing redundancy.

The student module will manage student records, including personal details, academic performance, and attendance. The faculty module will handle faculty information, such as personal details, schedules, and performance evaluations. The admin module will oversee the overall administration, including course management, scheduling, and resource allocation. By interconnecting these modules, the software will enable seamless communication and data sharing among different departments, significantly reducing the time and effort required to perform various operational tasks.

This integrated approach will lead to improved accuracy and efficiency in managing college records. It will also provide a single, accessible platform for students, faculty, and administrators to access and update information, thereby enhancing the overall productivity and effectiveness of the institution's management system.

5. SYSTEM DESIGN

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements.

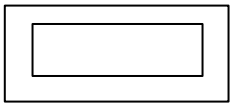
ER DIAGRAM

An Entity Relationship diagram (ERD) show the Relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagram illustrate the logical structure of Database. At first glance an entity Relationship diagram looks very much like a flowchart. It has the specialized symbols, and the meaning of those symbols, that make it unique.

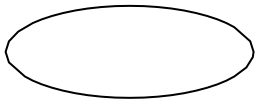
NOTATIONS FOR ER DIAGRAM

Entity

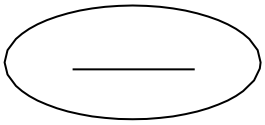
Weak Entity



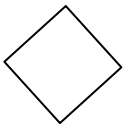
Attribute



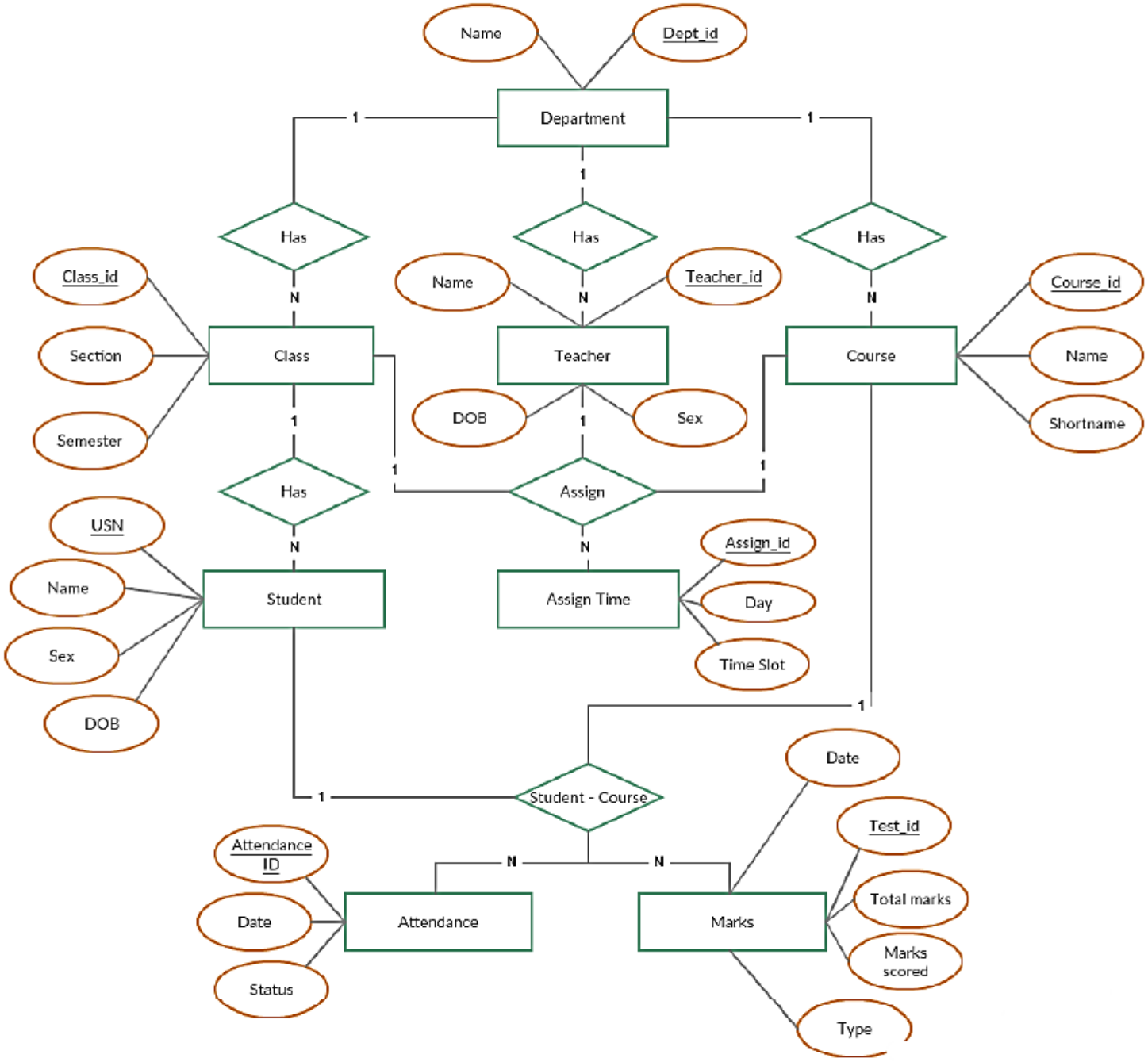
Key Attribute



Relationship



ER-DIAGRAM


Name
Dept_id
Department
1
1
1
Has
Has
Has
N
N
N
Class_id
Section
Semester
Class
Name
Teacher_id
Teacher
DOB
Sex
Course_id
Name
Shortname
Course
1
1
1
1
Has
Assign
N
N
USN
Name
Sex
DOB
Student
Assign_id
Day
Time Slot
Assign Time
1
1
Student - Course
Date
Test_id
Total marks
Marks scored
Type
Marks
N
N
Attendance ID
Date
Status
Attendance

6. DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationship

FEATURES OF DATAFLOW DIAGRAM

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

TYPES OF DATA FLOW DIAGRAM:

- Current physical
- Current logical
- New logical
- New physical

CURRENT PHYSICAL

In this DFD process level includes the name of the people or their position or the name or the computer system that might provide some of the overall-processing. Level includes an identification of the technology used of process the data. Similarly, data flows and data stores are often levels with the name of actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

CURRENT LOGICAL

The physical aspects of the system are removed as much as possible so that the currents system is reduced to its essence to the data and the processors that transforms them regardless of actual physical form.

NEW LOGICAL

This is actually like a current logical model if the users happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional function, absolute function removal and inefficient recognized.

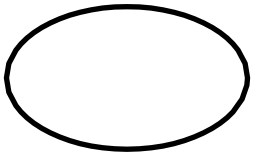
NEW PHYSICAL

In the context of a Data Flow Diagram (DFD), a "new physical" represents a new instance of a physical data store where information is stored and retrieved within a system. It signifies the introduction of a distinct storage repository to manage data associated with the system's processes and interactions. This new physical data store is an essential element in the DFD that allows the system to manage and maintain data persistently, enabling efficient data storage, retrieval, and manipulation. Its inclusion reflects the expansion of the system's capabilities, facilitating the organization and management of data generated or used by the system's various components and processes.

DFD Symbols

Entity -

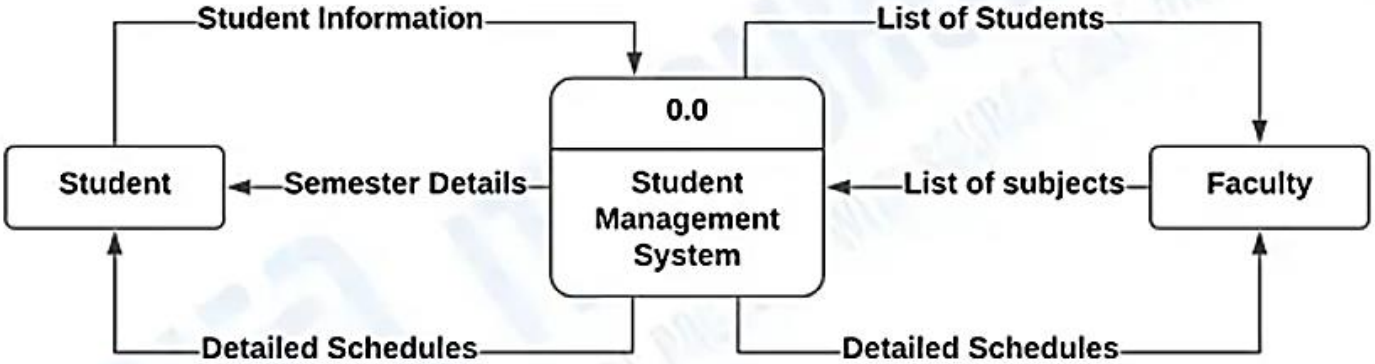
Process -



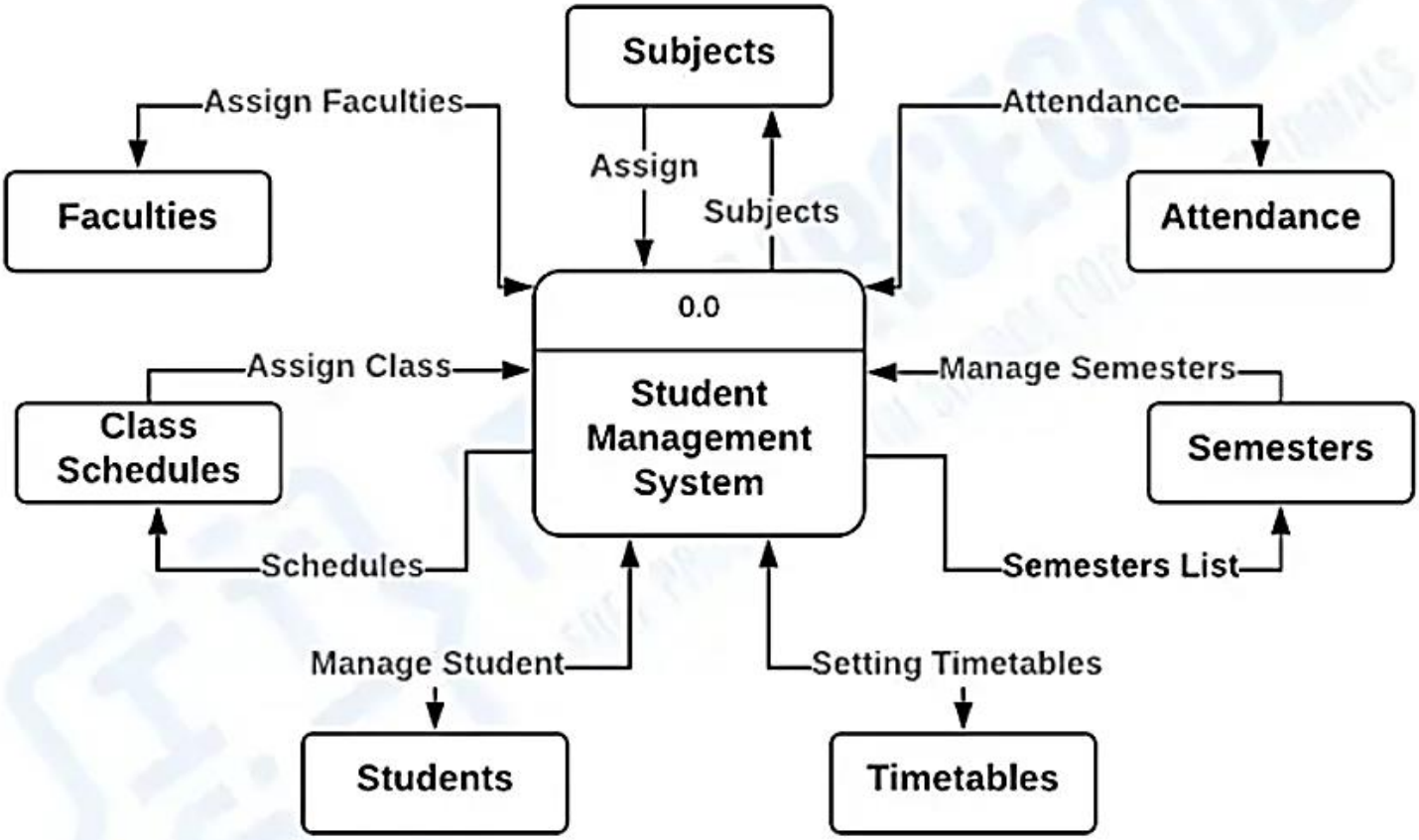
Connector -

DATA FLOW DIAGRAMS:

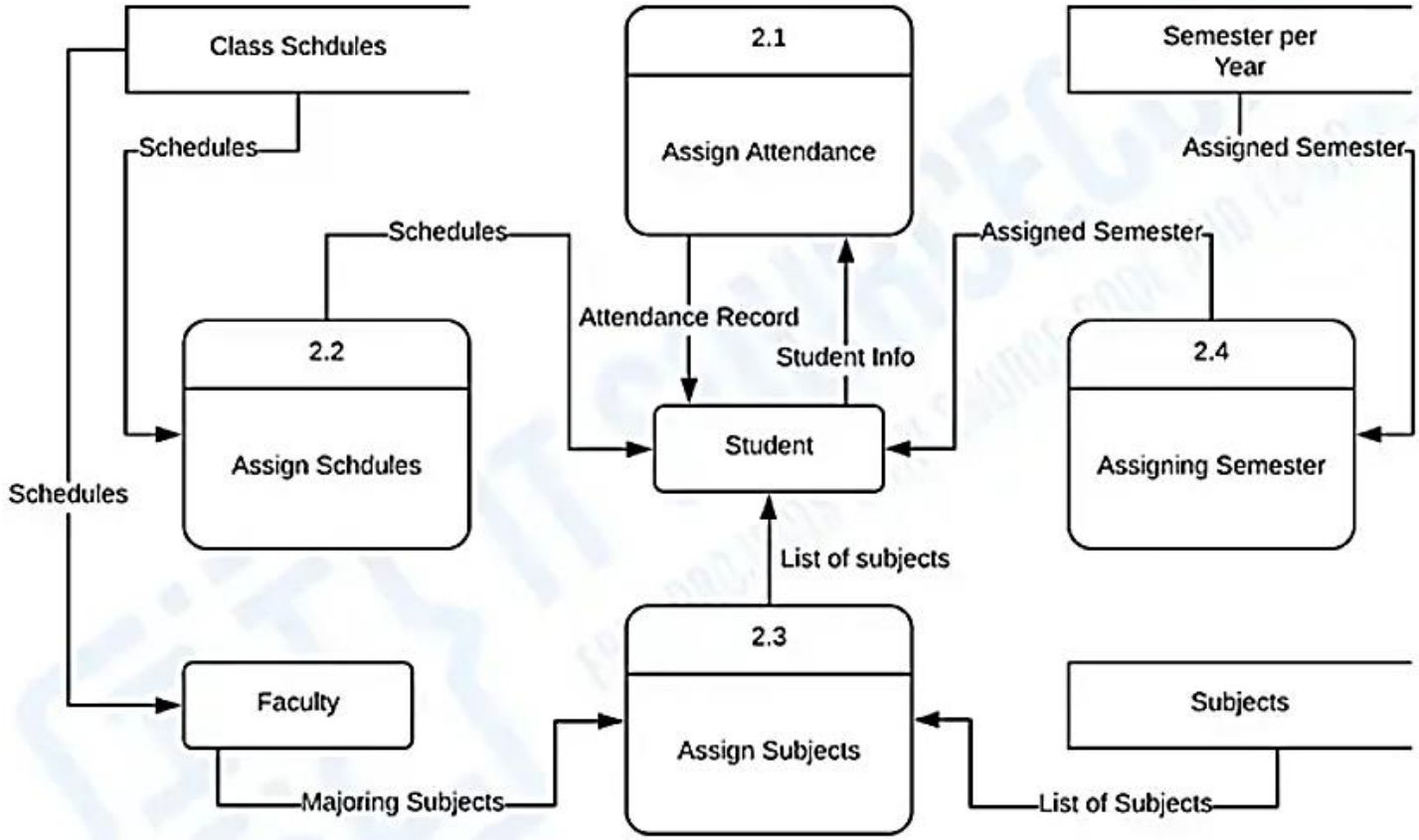
Level 0:



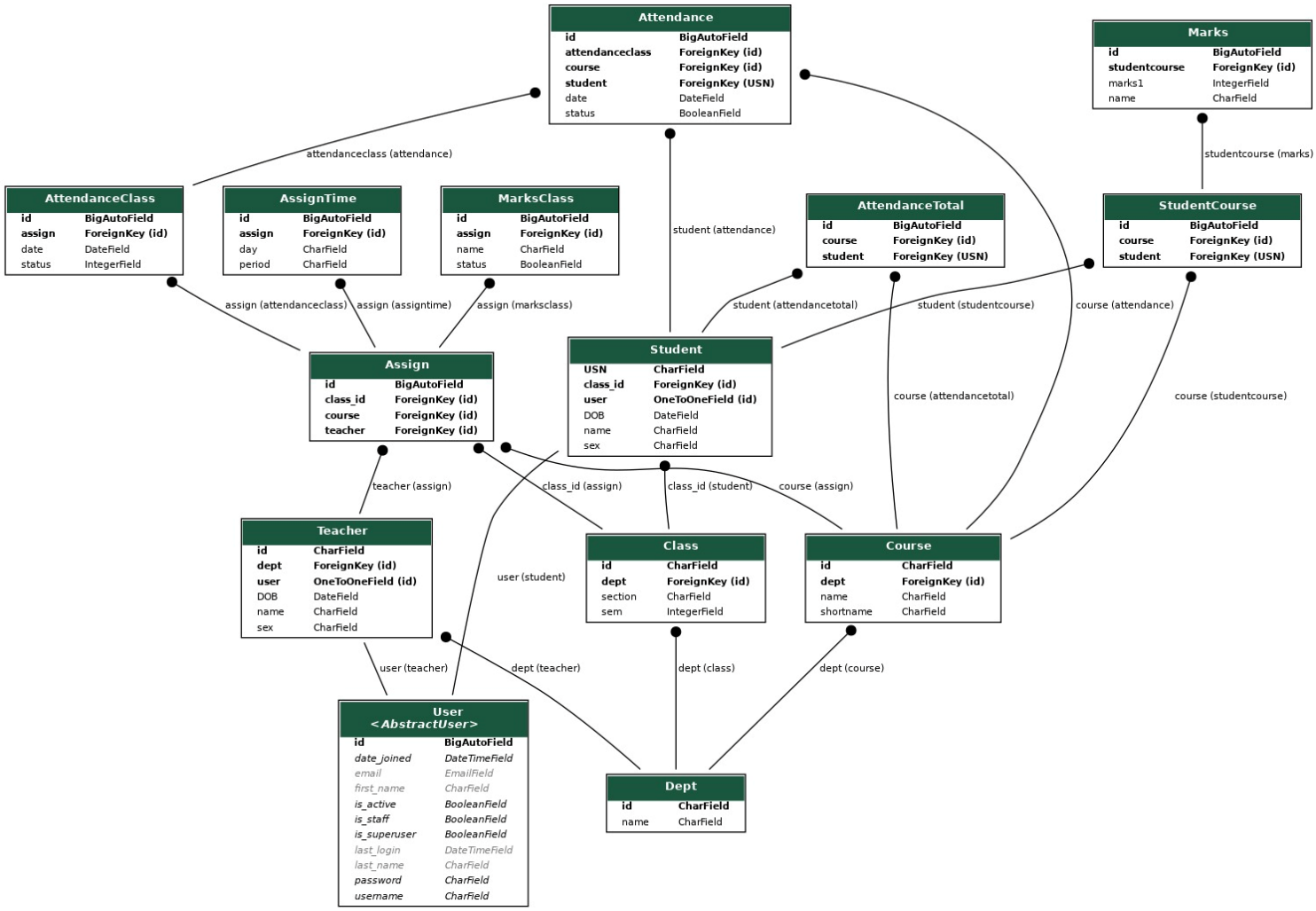
Level 1:


Subjects
Assign Faculties
Attendance
Assign
Subjects
Faculties
Attendance
0.0
Assign Class
Manage Semesters
Student Management System
Class Schedules
Semesters
Schedules
Semesters List
Manage Student
Setting Timetables
Students
Timetables

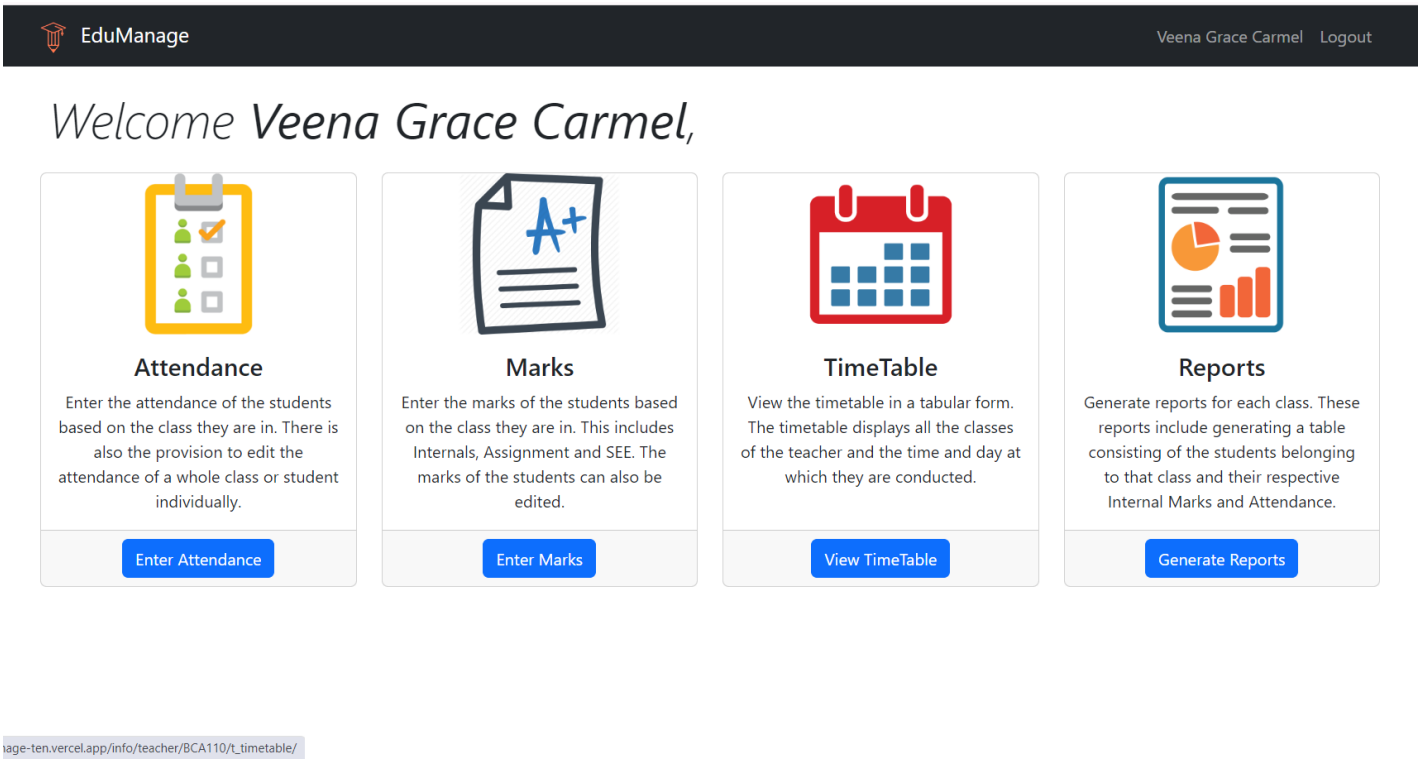
Level 2:


Class Schdules
2.1
Assign Attendance
Semester per Year
Schedules
Assigned Semester
Schedules
Assigned Semester
Attendance Record
Student Info
2.2
Assign Schdules
Student
2.4
Assigning Semester
Schedules
List of subjects
2.3
Assign Subjects
Faculty
Subjects
Majoring Subjects
List of Subjects

7. DATABASE


Attendance
id BigAutoField
attendanceclass ForeignKey (id)
course ForeignKey (id)
student ForeignKey (USN)
date DateField
status BooleanField
Marks
id BigAutoField
studentcourse ForeignKey (id)
marks1 IntegerField
name CharField
attendanceclass (attendance)
studentcourse (marks)
AttendanceClass
id BigAutoField
assign ForeignKey (id)
date DateField
status IntegerField
AssignTime
id BigAutoField
assign ForeignKey (id)
day CharField
period CharField
MarksClass
id BigAutoField
assign ForeignKey (id)
name CharField
status BooleanField
student (attendance)
AttendanceTotal
id BigAutoField
course ForeignKey (id)
student ForeignKey (USN)
StudentCourse
id BigAutoField
course ForeignKey (id)
student ForeignKey (USN)
assign (attendanceclass)
assign (assigntime)
assign (marksclass)
student (attendancetotal)
student (studentcourse)
course (attendance)
Assign
id BigAutoField
class_id ForeignKey (id)
course ForeignKey (id)
teacher ForeignKey (id)
Student
USN CharField
class_id ForeignKey (id)
user OneToOneField (id)
DOB DateField
name CharField
sex CharField
course (attendancetotal)
course (studentcourse)
teacher (assign)
class_id (assign)
class_id (student)
course (assign)
Teacher
id CharField
dept ForeignKey (id)
user OneToOneField (id)
DOB DateField
name CharField
sex CharField
user (student)
Class
id CharField
dept ForeignKey (id)
section CharField
sem IntegerField
Course
id CharField
dept ForeignKey (id)
name CharField
shortname CharField
user (teacher)
dept (teacher)
dept (class)
dept (course)
User
<AbstractUser>
id BigAutoField
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField
last_login DateTimeField
last_name CharField
password CharField
username CharField
Dept
id CharField
name CharField

8. SAMPLE TESTING



After running in VS code, Project running successfully

9. CODING

BACKEND

Backend/manage.py

```
import os
import sys
if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "EduManage.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)
```

Backend/Edumanage/settings.py

```
import os
import dj_database_url
from dotenv import load_dotenv
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
load_dotenv()

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = os.environ.get("SECRET_KEY")

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = os.environ.get("DEBUG")

ALLOWED_HOSTS = ["localhost", "127.0.0.1", ".vercel.app"]

AUTH_USER_MODEL = "info.User"


# Application definition

INSTALLED_APPS = [
    "jazzmin",
    "info.apps.InfoConfig",
    "django.contrib.admin",
```

```
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "rest_framework",
    "djoser",
    "rest_framework.authtoken",
    "apis",
    "django_extensions",
]

MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]

ROOT_URLCONF = "EduManage.urls"

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [os.path.join(BASE_DIR, "templates")],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]

WSGI_APPLICATION = "EduManage.wsgi.application"


# DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.sqlite3',
```

```
#         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
#     }
# }
DATABASES = {
    "default": dj_database_url.config(
        default=os.environ.get("POSTGRES_URL_NO_SSL"),
        conn_max_age=600,
        conn_health_checks=True,
        ssl_require=False,
    )
}


# Password validation

AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME": "django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.CommonPasswordValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.NumericPasswordValidator",
    },
]


# Internationalization
# https://docs.djangoproject.com/en/2.1/topics/i18n/

LANGUAGE_CODE = "en-us"

TIME_ZONE = "Asia/Kolkata"

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/2.1/howto/static-files/

STATIC_URL = "static/"
STATICFILES_DIRS = (os.path.join(BASE_DIR, "static/"),)
STATIC_ROOT = os.path.join(BASE_DIR, "staticfiles_build", "static")

LOGIN_REDIRECT_URL = "/"
LOGOUT_REDIRECT_URL = "/"

REST_FRAMEWORK = {
    "DEFAULT_PERMISSION_CLASSES": ("rest_framework.permissions.IsAuthenticated",),
    "DEFAULT_AUTHENTICATION_CLASSES": (
        "rest_framework.authentication.TokenAuthentication",
        "rest_framework.authentication.SessionAuthentication",
    ),
}
DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"


# JAZZMIN

JAZZMIN_SETTINGS = {
    "site_icon": "asishidea.ico",
    "custom_css": "admin/css/jazzmincustom.css",
    "site_logo": "admin/img/logo.png",
    "copyright": "Ajmal & Asish",
    "icons": {
        "info.assign": "fas fa-users-cog",
        "info.attendanceclass": "fas fa-check-square",
        "info.class": "fas fa-chalkboard",
        "info.course": "fas fa-book-open",
        "info.dept": "fas fa-school",
        "info.studentcourse": "fas fa-clipboard-list",
        "info.student": "fas fa-user-graduate",
        "info.teacher": "fas fa-chalkboard-teacher",
        "info.user": "fas fa-user",
    },
    "changeform_format": "single"
}


JAZZMIN_UI_TWEAKS = {
    "navbar_small_text": False,
    "footer_small_text": False,
    "body_small_text": False,
    "brand_small_text": False,
```

```
    "brand_colour": "navbar-dark",
    "accent": "accent-primary",
    "navbar": "navbar-dark",
    "no_navbar_border": False,
    "navbar_fixed": True,
    "layout_boxed": False,
    "footer_fixed": False,
    "sidebar_fixed": False,
    "sidebar": "sidebar-dark-primary",
    "sidebar_nav_small_text": False,
    "sidebar_disable_expand": True,
    "sidebar_nav_child_indent": False,
    "sidebar_nav_compact_style": False,
    "sidebar_nav_legacy_style": False,
    "sidebar_nav_flat_style": False,
    "theme": "default",
    "dark_mode_theme": None,
    "button_classes": {
        "primary": "btn-primary",
        "secondary": "btn-secondary",
        "info": "btn-info",
        "warning": "btn-warning",
        "danger": "btn-danger",
        "success": "btn-success",
    },
}
```

Backend/Edumanage/urls.py

```
from django.contrib import admin
from django.contrib.auth import views as auth_views
from django.urls import include, path
urlpatterns = [
    path("admin/", admin.site.urls),
    path("", include("info.urls")),
    path("info/", include("info.urls")),
    path("api/", include("apis.urls")),
    path(
        "accounts/login/",
        auth_views.LoginView.as_view(template_name="info/login.html"),
        name="login",
    ),
    path(
        "accounts/logout/", auth_views.LogoutView.as_view(next_page="/"), name="logout"
    ),
]
```

Backend/Edumanage/wsgi.py

```
import os
from django.core.wsgi import get_wsgi_application
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "EduManage.settings")
application = get_wsgi_application()
app = application
```

Backend/apis/serializers.py

```
from rest_framework import serializers
from info.models import *

class DetailSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = "__all__"

class AttendanceSerializer(serializers.ModelSerializer):
    class Meta:
        model = AttendanceTotal
        fields = "__all__"

class MarksSerializer(serializers.ModelSerializer):
    class Meta:
        model = Marks
        fields = "__all__"

class TimeTableSerializer(serializers.ModelSerializer):
    class Meta:
        model = AssignTime
        fields = "__all__"
```

Backend/apis/url.py

```
from django.urls import path
import apis.views as api_view
urlpatterns = [
    path("details/", api_view.DetailView.as_view()),
    path("attendance/", api_view.AttendanceView.as_view()),
    path("marks/", api_view.MarksView.as_view()),
    path("timetable/", api_view.TimetableView.as_view()),
```

Backend/apis/views.py

```
from rest_framework import status

from rest_framework.authtoken.models import Token
```

```
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from rest_framework.views import APIView

import apis.serializers as api_ser
from info.models import *
class DetailView(APIView):
    """
    Returns user's info.
    """
    permission_classes = [
        IsAuthenticated,
    ]
    def get(self, request):
        try:
            # fetching token sent in request header by the user.
            us = Token.objects.filter(user=request.user)
            if us:  # checking for authentication using token authentication.
                        # getting user from in-built user model class.
                user = User.objects.filter(auth_token=us[0]).first()
                # getting student from student model by filtering based on user that we got.
                details = Student.objects.get(user=user)
                serializer = api_ser.DetailSerializer(
                    details, context={"request": request}
                )  # Serializing the data into Json format.
                return Response(
                    {
                        "data": serializer.data,
                    },
                    status=status.HTTP_200_OK,
                )
            else:
                return Response(
```

```
                {"message": "User not authenticated"},
                status=status.HTTP_400_BAD_REQUEST,
            )
        except Exception as e:
            return Response(str(e), status=status.HTTP_400_BAD_REQUEST)

class AttendanceView(APIView):
    """
    This view is used to return user's attendance
    that is to check user's attendance.
    """
    permission_classes = [
        IsAuthenticated,
    ]
    def get(self, request):
        try:
            token = Token.objects.filter(user=request.user).first()
            if token:  # checking for authentication using token authentication.
                # getting user from in-built user model class.
                user = User.objects.get(auth_token=token)
                # getting student from student model by filtering based on user that we got.
                stud = Student.objects.get(user=user)
                # using ass_list and att_list we get the classes assigned to that user
                ass_list = Assign.objects.filter(class_id_id=stud.class_id)
                # and respectively their attendance
                att_list = []
                for ass in ass_list:
                    try:
                        a = AttendanceTotal.objects.get(student=stud, course=ass.course)
                    except AttendanceTotal.DoesNotExist:
                        a = AttendanceTotal(student=stud, course=ass.course)
                        a.save()
                    att_list.append(a)
```

```
            serializer = api_ser.AttendanceSerializer(
                att_list, many=True, context={"request": request}
            )  # Serializing the data into Json format.
            return Response(
                {
                    "user_attendance": serializer.data,
                },
                status=status.HTTP_200_OK,
            )
        else:
         # returning not authenticated message when user isn't authenticated with status code 400.
            return Response(
                {"message": "User not authenticated"},
                status=status.HTTP_400_BAD_REQUEST,
            )
    except Exception as e:
        return Response(str(e), status=status.HTTP_400_BAD_REQUEST)
class MarksView(APIView):
    """
    This view is used to return user's marks
    that is to check user's marks in different subjects as given by the teacher.
    """
    permission_classes = [
        IsAuthenticated,
    ]
    def get(self, request):
        try:
            token = Token.objects.filter(user=request.user).first()
            if token:  # checking for authentication using token authentication.
                user = User.objects.get(auth_token=token)
                stud = Student.objects.get(user=user)

                # using ass_list and sc_list we retrieve all the subjects assigned
```

```
            ass_list = Assign.objects.filter(class_id_id=stud.class_id)
            # and then their respective marks. Store them in a dictionary and return it to the user.
            sc_list = []
            for ass in ass_list:
                sc = StudentCourse.objects.get(student=stud, course=ass.course)
                sc_list.append(sc)
            sc_total = {}
            for sc in sc_list:
                for m in sc.marks_set.all():
                    sc_total[m.studentcourse.course.name] = m.marks1
            return Response(
                {
                    "user_marks": sc_total,
                },
                status=status.HTTP_200_OK,
            )
        else:
            return Response(
                {"message": "User not authenticated"},
                status=status.HTTP_400_BAD_REQUEST,
            )
    except Exception as e:
        return Response(str(e), status=status.HTTP_400_BAD_REQUEST)

class TimetableView(APIView):
    """
    This view is used to check user's class timetable
    It returns the respective class' timetable to which the user is assigned.
    """

    permission_classes = [
        IsAuthenticated,
    ]
```

```
    def get(self, request):
        try:
            token = Token.objects.filter(user=request.user).first()
            if token:  # checking for authentication using token authentication.
                user = User.objects.get(auth_token=token)
                stud = Student.objects.get(user=user)
                asst = AssignTime.objects.filter(assign__class_id=stud.class_id)
                serializer = api_ser.TimeTableSerializer(
                    asst, many=True, context={"request": request}
                )  # Serializing the data into Json format.
                return Response(
                    {
                        "user_marks": serializer.data,
                    },
                    status=status.HTTP_200_OK,
                )
            else:
                return Response(
                    {"message": "User not authenticated"},
                    status=status.HTTP_400_BAD_REQUEST,
                )
        except Exception as e:
            return Response(str(e), status=status.HTTP_400_BAD_REQUEST)
```

Backend/info/admin.py

```
from datetime import datetime, timedelta
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from django.http import HttpResponseRedirect
from django.urls import path
from .models import (Assign, AssignTime, Attendance, AttendanceClass,
```

```
                AttendanceRange, Class, Course, Dept, Marks, Student,
                StudentCourse, Teacher, User)
# Register your models here.
days = {
    "Monday": 1,
    "Tuesday": 2,
    "Wednesday": 3,
    "Thursday": 4,
    "Friday": 5,
    "Saturday": 6,
}
def daterange(start_date, end_date):
    for n in range(int((end_date - start_date).days)):
        yield start_date + timedelta(n)
class ClassInline(admin.TabularInline):
    model = Class
    extra = 0
class DeptAdmin(admin.ModelAdmin):
    inlines = [ClassInline]
    list_display = ("name", "id")
    search_fields = ("name", "id")
    ordering = ["name"]

class StudentInline(admin.TabularInline):
    model = Student
    extra = 0
class ClassAdmin(admin.ModelAdmin):
    list_display = ("id", "dept", "sem", "section")
    search_fields = ("id", "dept__name", "sem", "section")
```

```
    ordering = ["dept__name", "sem", "section"]
    inlines = [StudentInline]
class CourseAdmin(admin.ModelAdmin):
    list_display = ("id", "name", "dept")
    search_fields = ("id", "name", "dept__name")
    ordering = ["dept", "id"]
class AssignTimeInline(admin.TabularInline):
    model = AssignTime
    extra = 0
class AssignAdmin(admin.ModelAdmin):
    inlines = [AssignTimeInline]
    list_display = ("class_id", "course", "teacher")
    search_fields = (
        "class_id__dept__name",
        "class_id__id",
        "course__name",
        "teacher__name",
        "course__shortname",
    )
    ordering = ["class_id__dept__name", "class_id__id", "course__id"]
    raw_id_fields = ["class_id", "course", "teacher"]

class MarksInline(admin.TabularInline):
    model = Marks
    extra = 0
class StudentCourseAdmin(admin.ModelAdmin):
    inlines = [MarksInline]
    list_display = (
        "student",
```

```
        "course",
    )
    search_fields = (
        "student__name",
        "course__name",
        "student__class_id__id",
        "student__class_id__dept__name",
    )
    ordering = (
        "student__class_id__dept__name",
        "student__class_id__id",
        "student__USN",
    )
class StudentAdmin(admin.ModelAdmin):
    list_display = ("USN", "name", "class_id")
    search_fields = ("USN", "name", "class_id__id", "class_id__dept__name")
    ordering = ["class_id__dept__name", "class_id__id", "USN"]
class TeacherAdmin(admin.ModelAdmin):
    list_display = ("name", "dept")
    search_fields = ("name", "dept__name")
    ordering = ["dept__name", "name"]

class AttendanceClassAdmin(admin.ModelAdmin):
    list_display = ("assign", "date", "status")
    ordering = ["assign", "date"]
    change_list_template = "admin/attendance/attendance_change_list.html"
    def changelist_view(self, request, extra_context=None):
        extra_context = extra_context or {}
        try:
```

```
        extra_context["current_range"] = AttendanceRange.objects.latest("id")
    except AttendanceRange.DoesNotExist:
        extra_context["current_range"] = None
    return super().changelist_view(request, extra_context=extra_context)
def get_urls(self):
    urls = super().get_urls()
    my_urls = [
        path("reset_attd/", self.reset_attd, name="reset_attd"),
    ]
    return my_urls + urls
def reset_attd(self, request):
    start_date = datetime.strptime(request.POST["startdate"], "%Y-%m-%d").date()
    end_date = datetime.strptime(request.POST["enddate"], "%Y-%m-%d").date()
    try:
        a = AttendanceRange.objects.all()[:1].get()
        a.start_date = start_date
        a.end_date = end_date
        a.save()
    except AttendanceRange.DoesNotExist:
        a = AttendanceRange(start_date=start_date, end_date=end_date)
        a.save()

    Attendance.objects.all().delete()
    AttendanceClass.objects.all().delete()
    for asst in AssignTime.objects.all():
        for single_date in daterange(start_date, end_date):
            if single_date.isoweekday() == days[asst.day]:
                try:
                    AttendanceClass.objects.get(
```

```
                date=single_date.strftime("%Y-%m-%d"), assign=asst.assign
            )
        except AttendanceClass.DoesNotExist:
            a = AttendanceClass(
                date=single_date.strftime("%Y-%m-%d"), assign=asst.assign
            )
            a.save()
    self.message_user(request, "Attendance Dates reset successfully!")
    return HttpResponseRedirect("../")
admin.site.register(User, UserAdmin)
admin.site.register(Dept, DeptAdmin)
admin.site.register(Class, ClassAdmin)
admin.site.register(Student, StudentAdmin)
admin.site.register(Course, CourseAdmin)
admin.site.register(Teacher, TeacherAdmin)
admin.site.register(Assign, AssignAdmin)
admin.site.register(StudentCourse, StudentCourseAdmin)
admin.site.register(AttendanceClass, AttendanceClassAdmin)
```

Backend/info/models.py

```
import math
from datetime import date, timedelta
from django.contrib.auth.models import AbstractUser
from django.core.validators import MaxValueValidator, MinValueValidator
from django.db import models
from django.db.models.signals import post_delete, post_save
# Create your models here.
sex_choice = (("Male", "Male"), ("Female", "Female"))
time_slots = (
    ("8:30 - 9:45", "8:30 - 9:45"),
```

```
    ("9:45 - 10:45", "9:45 - 10:45"),
    ("11:10 - 12:10", "11:10 - 12:10"),
    ("12:10 - 1:10", "12:10 - 1:10"),
    ("1:50 - 2:40", "1:50 - 2:40"),
    ("2:40 - 3:30", "2:40 - 3:30"),
)
DAYS_OF_WEEK = (
    ("Monday", "Monday"),
    ("Tuesday", "Tuesday"),
    ("Wednesday", "Wednesday"),
    ("Thursday", "Thursday"),
    ("Friday", "Friday"),
    ("Saturday", "Saturday"),
)
test_name = (
    ("Internal test 1", "Internal test 1"),
    ("Internal test 2", "Internal test 2"),
    ("Assignment 1", "Assignment 1"),
    ("Assignment 2", "Assignment 2"),
    ("Semester End Exam", "Semester End Exam"),
)
class User(AbstractUser):
    @property
    def is_student(self):
        if hasattr(self, "student"):
            return True
        return False
    @property
    def is_teacher(self):
```

```
        if hasattr(self, "teacher"):
            return True
        return False
class Dept(models.Model):
    id = models.CharField(primary_key="True", max_length=100)
    name = models.CharField(max_length=200)
    def __str__(self):
        return self.name
class Course(models.Model):
    dept = models.ForeignKey(Dept, on_delete=models.CASCADE)
    id = models.CharField(primary_key="True", max_length=50)
    name = models.CharField(max_length=50)
    shortname = models.CharField(max_length=50, default="X")
    def __str__(self):
        return self.name
class Class(models.Model):
    # courses = models.ManyToManyField(Course, default=1)
    id = models.CharField(primary_key="True", max_length=100)
    dept = models.ForeignKey(Dept, on_delete=models.CASCADE)
    section = models.CharField(max_length=100)
    sem = models.IntegerField()
    class Meta:
        verbose_name_plural = "classes"
    def __str__(self):
        d = Dept.objects.get(name=self.dept)
        return "%s : %d %s" % (d.name, self.sem, self.section)
class Student(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, null=True)
    class_id = models.ForeignKey(Class, on_delete=models.CASCADE, default=1)
```

```
    USN = models.CharField(primary_key="True", max_length=100)
    name = models.CharField(max_length=200)
    sex = models.CharField(max_length=50, choices=sex_choice, default="Male")
    DOB = models.DateField(default="1998-01-01")
    def __str__(self):
        return self.name
class Teacher(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, null=True)
    id = models.CharField(primary_key=True, max_length=100)
    dept = models.ForeignKey(Dept, on_delete=models.CASCADE, default=1)
    name = models.CharField(max_length=100)
    sex = models.CharField(max_length=50, choices=sex_choice, default="Male")
    DOB = models.DateField(default="2001-01-01")
    def __str__(self):
        return self.name
class Assign(models.Model):
    class_id = models.ForeignKey(Class, on_delete=models.CASCADE)
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE)
    class Meta:
        unique_together = (("course", "class_id", "teacher"),)
    def __str__(self):
        cl = Class.objects.get(id=self.class_id_id)
        cr = Course.objects.get(id=self.course_id)
        te = Teacher.objects.get(id=self.teacher_id)
        return "%s : %s : %s" % (te.name, cr.shortname, cl)
class AssignTime(models.Model):
    assign = models.ForeignKey(Assign, on_delete=models.CASCADE)
    period = models.CharField(
```

```
        max_length=50, choices=time_slots, default="11:10 - 12:10"
    )
    day = models.CharField(max_length=15, choices=DAYS_OF_WEEK)
class AttendanceClass(models.Model):
    assign = models.ForeignKey(Assign, on_delete=models.CASCADE)
    date = models.DateField()
    status = models.IntegerField(default=0)
    class Meta:
        verbose_name = "Attendance"
        verbose_name_plural = "Attendance"
class Attendance(models.Model):
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    student = models.ForeignKey(Student, on_delete=models.CASCADE)
    attendanceclass = models.ForeignKey(
        AttendanceClass, on_delete=models.CASCADE, default=1
    )
    date = models.DateField(default="2024-05-01")
    status = models.BooleanField(default="True")

    def __str__(self):
        sname = Student.objects.get(name=self.student)
        cname = Course.objects.get(name=self.course)
        return "%s : %s" % (sname.name, cname.shortname)
class AttendanceTotal(models.Model):
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    student = models.ForeignKey(Student, on_delete=models.CASCADE)
    class Meta:
        unique_together = (("student", "course"),)
    @property
```

```
def att_class(self):
    stud = Student.objects.get(name=self.student)
    cr = Course.objects.get(name=self.course)
    att_class = Attendance.objects.filter(
        course=cr, student=stud, status="True"
    ).count()
    return att_class

@property
def total_class(self):
    stud = Student.objects.get(name=self.student)
    cr = Course.objects.get(name=self.course)
    total_class = Attendance.objects.filter(
        course=cr, student=stud).count()
    return total_class

@property
def attendance(self):
    stud = Student.objects.get(name=self.student)
    cr = Course.objects.get(name=self.course)
    total_class = Attendance.objects.filter(
        course=cr, student=stud).count()
    att_class = Attendance.objects.filter(
        course=cr, student=stud, status="True"
    ).count()
    if total_class == 0:
        attendance = 0
    else:
        attendance = round(att_class / total_class * 100, 2)
```

```
        return attendance


    @property
    def classes_to_attend(self):
        stud = Student.objects.get(name=self.student)
        cr = Course.objects.get(name=self.course)
        total_class = Attendance.objects.filter(
            course=cr, student=stud).count()
        att_class = Attendance.objects.filter(
            course=cr, student=stud, status="True"
        ).count()
        cta = math.ceil((0.75 * total_class - att_class) / 0.25)
        if cta < 0:
            return 0
        return cta
class StudentCourse(models.Model):
    student = models.ForeignKey(Student, on_delete=models.CASCADE)
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    class Meta:
        unique_together = (("student", "course"),)
        verbose_name_plural = "Marks"
    def __str__(self):
        sname = Student.objects.get(name=self.student)
        cname = Course.objects.get(name=self.course)
        return "%s : %s" % (sname.name, cname.shortname)
    def get_cie(self):
        # Fetch the marks for assignments and internal tests
        assignment_1_marks = self.marks_set.filter(
            name='Assignment 1').first().marks1
```

```
        assignment_2_marks = self.marks_set.filter(
            name='Assignment 2').first().marks1
        internal_1_marks = self.marks_set.filter(
            name='Internal test 1').first().marks1
        internal_2_marks = self.marks_set.filter(
            name='Internal test 2').first().marks1

    # Convert assignment marks to a value out of 40
        assignment_1_converted = (assignment_1_marks / 20) * 40
        assignment_2_converted = (assignment_2_marks / 20) * 40

    # Calculate the total CIE marks out of 40
        cie_total = math.ceil(
            (assignment_1_converted + assignment_2_converted + internal_1_marks +
internal_2_marks) / 4)
        return cie_total

    def get_attendance(self):
        a = AttendanceTotal.objects.get(
            student=self.student, course=self.course)
        return a.attendance
class Marks(models.Model):
    studentcourse = models.ForeignKey(StudentCourse, on_delete=models.CASCADE)
    name = models.CharField(
        max_length=50, choices=test_name, default="Internal test 1")
    marks1 = models.IntegerField(
        default=0, validators=[MinValueValidator(0), MaxValueValidator(100)]
    )

    class Meta:
```

```
        unique_together = (("studentcourse", "name"),)

    @property
    def total_marks(self):
        if self.name == "Semester End Exam":
            return 60
        elif self.name == "Assignment 1":
            return 20
        elif self.name == "Assignment 2":
            return 20
        return 40
class MarksClass(models.Model):
    assign = models.ForeignKey(Assign, on_delete=models.CASCADE)
    name = models.CharField(
        max_length=50, choices=test_name, default="Internal test 1")
    status = models.BooleanField(default="False")

    class Meta:
        unique_together = (("assign", "name"),)

    @property
    def total_marks(self):
        if self.name == "Semester End Exam":
            return 60
        elif self.name == "Assignment 1":
            return 20
        elif self.name == "Assignment 2":
            return 20
        return 40
```

```
class AttendanceRange(models.Model):
    start_date = models.DateField()
    end_date = models.DateField()
# Triggers
def daterange(start_date, end_date):
    for n in range(int((end_date - start_date).days)):
        yield start_date + timedelta(n)
days = {
    "Monday": 1,
    "Tuesday": 2,
    "Wednesday": 3,
    "Thursday": 4,
    "Friday": 5,
    "Saturday": 6,
}
def create_attendance(sender, instance, **kwargs):
    if kwargs["created"]:
        try:
            attendance_range = AttendanceRange.objects.all()[:1].get()
            start_date = attendance_range.start_date
            end_date = attendance_range.end_date
        except AttendanceRange.DoesNotExist:
            # Create a default range if none exists
            today = date.today()
            start_date = today
            # Default to one year from today
            end_date = today + timedelta(weeks=20)
            AttendanceRange.objects.create(
                start_date=start_date, end_date=end_date)
```

```
        for single_date in daterange(start_date, end_date):
            if single_date.isoweekday() == days[instance.day]:
                AttendanceClass.objects.get_or_create(
                    date=single_date, assign=instance.assign, defaults={
                        "status": 0}
                )
def create_marks(sender, instance, **kwargs):
    if kwargs["created"]:
        if hasattr(instance, "name"):
            ass_list = instance.class_id.assign_set.all()
            for ass in ass_list:
                try:
                    StudentCourse.objects.get(
                        student=instance, course=ass.course)
                except StudentCourse.DoesNotExist:
                    sc = StudentCourse(student=instance, course=ass.course)
                    sc.save()
                    sc.marks_set.create(name="Internal test 1")
                    sc.marks_set.create(name="Internal test 2")
                    sc.marks_set.create(name="Assignment 1")
                    sc.marks_set.create(name="Assignment 2")
                    sc.marks_set.create(name="Semester End Exam")
        elif hasattr(instance, "course"):
            stud_list = instance.class_id.student_set.all()
            cr = instance.course
            for s in stud_list:
                try:
                    StudentCourse.objects.get(student=s, course=cr)
```

```
            except StudentCourse.DoesNotExist:
                sc = StudentCourse(student=s, course=cr)
                sc.save()
                sc.marks_set.create(name="Internal test 1")
                sc.marks_set.create(name="Internal test 2")
                sc.marks_set.create(name="Assignment 1")
                sc.marks_set.create(name="Assignment 2")
                sc.marks_set.create(name="Semester End Exam")


def create_marks_class(sender, instance, **kwargs):
    if kwargs["created"]:
        for name in test_name:
            try:
                MarksClass.objects.get(assign=instance, name=name[0])
            except MarksClass.DoesNotExist:
                m = MarksClass(assign=instance, name=name[0])
                m.save()
def delete_marks(sender, instance, **kwargs):
    stud_list = instance.class_id.student_set.all()
    StudentCourse.objects.filter(
        course=instance.course, student__in=stud_list).delete()
post_save.connect(create_marks, sender=Student)
post_save.connect(create_marks, sender=Assign)
post_save.connect(create_marks_class, sender=Assign)
post_save.connect(create_attendance, sender=AssignTime)
post_delete.connect(delete_marks, sender=Assign)
```

Backend/info/urls.py

```
from django.contrib import admin
```

```
from django.urls import include, path
from . import views
urlpatterns = [
    path("", views.index, name="index"),
    path("student/<slug:stud_id>/attendance/", views.attendance, name="attendance"),
    path(
        "student/<slug:stud_id>/<slug:course_id>/attendance/",
        views.attendance_detail,
        name="attendance_detail",
    ),
    path("student/<slug:class_id>/timetable/", views.timetable, name="timetable"),
    # path('student/<slug:class_id>/search/', views.student_search, name='student_search'),
    path("student/<slug:stud_id>/marks_list/", views.marks_list, name="marks_list"),
    path(
        "teacher/<slug:teacher_id>/<int:choice>/Classes/", views.t_clas, name="t_clas"
    ),
    path(
        "teacher/<int:assign_id>/Students/attendance/",
        views.t_student,
        name="t_student",
    ),
    path(
        "teacher/<int:assign_id>/ClassDates/", views.t_class_date, name="t_class_date"
    ),
    path("teacher/<int:ass_c_id>/Cancel/", views.cancel_class, name="cancel_class"),
    path("teacher/<int:ass_c_id>/attendance/", views.t_attendance, name="t_attendance"),
    path("teacher/<int:ass_c_id>/Edit_att/", views.edit_att, name="edit_att"),
    path("teacher/<int:ass_c_id>/attendance/confirm/", views.confirm, name="confirm"),
    path(
```

```
    "teacher/<slug:stud_id>/<slug:course_id>/attendance/",
    views.t_attendance_detail,
    name="t_attendance_detail",
),
path(
    "teacher/<int:att_id>/change_attendance/", views.change_att, name="change_att"
),
path(
    "teacher/<int:assign_id>/Extra_class/",
    views.t_extra_class,
    name="t_extra_class",
),
path(
    "teacher/<slug:assign_id>/Extra_class/confirm/",
    views.e_confirm,
    name="e_confirm",
),
path("teacher/<int:assign_id>/Report/", views.t_report, name="t_report"),
path(
    "teacher/<slug:teacher_id>/t_timetable/", views.t_timetable, name="t_timetable"
),
path(
    "teacher/<int:assign_id>/marks_list/", views.t_marks_list, name="t_marks_list"
),
path(
    "teacher/<int:assign_id>/Students/Marks/",
    views.student_marks,
    name="t_student_marks",
),
```

```
    path(
        "teacher/<int:marks_c_id>/marks_entry/",
        views.t_marks_entry,
        name="t_marks_entry",
    ),
    path(
        "teacher/<int:marks_c_id>/marks_entry/confirm/",
        views.marks_confirm,
        name="marks_confirm",
    ),
    path("teacher/<int:marks_c_id>/Edit_marks/", views.edit_marks, name="edit_marks"),
    path("api/auth/", include("djoser.urls")),
    path("add-teacher/", views.add_teacher, name="add_teacher"),
    path("add-student/", views.add_student, name="add_student"),
]
admin.site.site_url = None
admin.site.site_header = "EduManage"
admin.site.site_title = "EduManage"
admin.site.index_title = "College Administrator"
```

Backend/info/views.py

```
from django.contrib.auth import get_user_model
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404, redirect, render
from django.urls import reverse
from django.utils import timezone
from .models import (DAYS_OF_WEEK, Assign, AssignTime, Attendance,
                     AttendanceClass, AttendanceTotal, Class, Course, Dept,
                     MarksClass, Student, StudentCourse, Teacher, time_slots)
```

```
User = get_user_model()
# Create your views here.
@login_required
def index(request):
    if request.user.is_teacher:
        return render(request, "info/t_homepage.html")
    if request.user.is_student:
        return render(request, "info/homepage.html")
    if request.user.is_superuser:
        return render(request, "info/admin_page.html")
    return render(request, "info/logout.html")
@login_required()
def attendance(request, stud_id):
    stud = Student.objects.get(USN=stud_id)
    ass_list = Assign.objects.filter(class_id_id=stud.class_id)
    att_list = []
    for ass in ass_list:
        try:
            a = AttendanceTotal.objects.get(student=stud, course=ass.course)
        except AttendanceTotal.DoesNotExist:
            a = AttendanceTotal(student=stud, course=ass.course)
            a.save()
        att_list.append(a)
    return render(request, "info/attendance.html", {"att_list": att_list})
@login_required()
def attendance_detail(request, stud_id, course_id):
    stud = get_object_or_404(Student, USN=stud_id)
    cr = get_object_or_404(Course, id=course_id)
    att_list = Attendance.objects.filter(course=cr, student=stud).order_by("date")
```

```
    return render(request, "info/att_detail.html", {"att_list": att_list, "cr": cr})
# Teacher Views
@login_required
def t_clas(request, teacher_id, choice):
    teacher1 = get_object_or_404(Teacher, id=teacher_id)
    return render(request, "info/t_clas.html", {"teacher1": teacher1, "choice": choice})
@login_required()
def t_student(request, assign_id):
    ass = Assign.objects.get(id=assign_id)
    att_list = []
    for stud in ass.class_id.student_set.all():
        try:
            a = AttendanceTotal.objects.get(student=stud, course=ass.course)
        except AttendanceTotal.DoesNotExist:
            a = AttendanceTotal(student=stud, course=ass.course)
            a.save()
        att_list.append(a)
    return render(request, "info/t_students.html", {"att_list": att_list})
@login_required()
def t_class_date(request, assign_id):
    now = timezone.now()
    ass = get_object_or_404(Assign, id=assign_id)
    att_list = ass.attendanceclass_set.filter(date__lte=now).order_by("-date")
    return render(request, "info/t_class_date.html", {"att_list": att_list})
@login_required()
def cancel_class(request, ass_c_id):
    assc = get_object_or_404(AttendanceClass, id=ass_c_id)
    assc.status = 2
    assc.save()
```

```
    return HttpResponseRedirect(reverse("t_class_date", args=(assc.assign_id,)))
@login_required()
def t_attendance(request, ass_c_id):
    assc = get_object_or_404(AttendanceClass, id=ass_c_id)
    ass = assc.assign
    c = ass.class_id
    context = {
        "ass": ass,
        "c": c,
        "assc": assc,
    }
    return render(request, "info/t_attendance.html", context)
@login_required()
def edit_att(request, ass_c_id):
    assc = get_object_or_404(AttendanceClass, id=ass_c_id)
    cr = assc.assign.course
    att_list = Attendance.objects.filter(attendanceclass=assc, course=cr)
    context = {
        "assc": assc,
        "att_list": att_list,
    }
    return render(request, "info/t_edit_att.html", context)
@login_required()
def confirm(request, ass_c_id):
    assc = get_object_or_404(AttendanceClass, id=ass_c_id)
    ass = assc.assign
    cr = ass.course
    cl = ass.class_id
    for i, s in enumerate(cl.student_set.all()):
```

```
status = request.POST[s.USN]
if status == "present":
    status = "True"
else:
    status = "False"
if assc.status == 1:
    try:
        a = Attendance.objects.get(
            course=cr, student=s, date=assc.date, attendanceclass=assc
        )
        a.status = status
        a.save()
    except Attendance.DoesNotExist:
        a = Attendance(
            course=cr,
            student=s,
            status=status,
            date=assc.date,
            attendanceclass=assc,
        )
        a.save()
else:
    a = Attendance(
        course=cr,
        student=s,
        status=status,
        date=assc.date,
        attendanceclass=assc,
    )
```

```
            a.save()
            assc.status = 1
            assc.save()
    return HttpResponseRedirect(reverse("t_class_date", args=(ass.id,)))
@login_required()
def t_attendance_detail(request, stud_id, course_id):
    stud = get_object_or_404(Student, USN=stud_id)
    cr = get_object_or_404(Course, id=course_id)
    att_list = Attendance.objects.filter(course=cr, student=stud).order_by("date")
    return render(request, "info/t_att_detail.html", {"att_list": att_list, "cr": cr})
@login_required()
def change_att(request, att_id):
    a = get_object_or_404(Attendance, id=att_id)
    a.status = not a.status
    a.save()
    return HttpResponseRedirect(
        reverse("t_attendance_detail", args=(a.student.USN, a.course_id))
    )
@login_required()
def t_extra_class(request, assign_id):
    ass = get_object_or_404(Assign, id=assign_id)
    c = ass.class_id
    context = {
        "ass": ass,
        "c": c,
    }
    return render(request, "info/t_extra_class.html", context)
@login_required()
def e_confirm(request, assign_id):
```

```
    ass = get_object_or_404(Assign, id=assign_id)
    cr = ass.course
    cl = ass.class_id
    assc = ass.attendanceclass_set.create(status=1, date=request.POST["date"])
    assc.save()
    for i, s in enumerate(cl.student_set.all()):
        status = request.POST[s.USN]
        if status == "present":
            status = "True"
        else:
            status = "False"
        date = request.POST["date"]
        a = Attendance(
            course=cr, student=s, status=status, date=date, attendanceclass=assc
        )
        a.save()

    return HttpResponseRedirect(reverse("t_clas", args=(ass.teacher_id, 1)))
@login_required()
def t_report(request, assign_id):
    ass = get_object_or_404(Assign, id=assign_id)
    sc_list = []
    for stud in ass.class_id.student_set.all():
        a = StudentCourse.objects.get(student=stud, course=ass.course)
        sc_list.append(a)
    return render(request, "info/t_report.html", {"sc_list": sc_list})
@login_required()
def timetable(request, class_id):
    asst = AssignTime.objects.filter(assign__class_id=class_id)
```

```
    matrix = [["" for i in range(9)] for j in range(6)]
    for i, d in enumerate(DAYS_OF_WEEK):
        t = 0
        for j in range(9):
            if j == 0:
                matrix[i][0] = d[0]
                continue
            if j == 3 or j == 6:
                continue
            try:
                a = asst.get(period=time_slots[t][0], day=d[0])
                # matrix[i][j] = a.assign.course_id
                matrix[i][j] = a
            except AssignTime.DoesNotExist:
                pass
            t += 1

    context = {"matrix": matrix}
    return render(request, "info/timetable.html", context)
@login_required()
def t_timetable(request, teacher_id):
    asst = AssignTime.objects.filter(assign__teacher_id=teacher_id)
    class_matrix = [[True for i in range(9)] for j in range(6)]
    for i, d in enumerate(DAYS_OF_WEEK):
        t = 0
        for j in range(9):
            if j == 0:
                class_matrix[i][0] = d[0]
                continue
```

```
            if j == 3 or j == 6:
                continue
            try:
                a = asst.get(period=time_slots[t][0], day=d[0])
                class_matrix[i][j] = a
            except AssignTime.DoesNotExist:
                pass
            t += 1
    context = {
        "class_matrix": class_matrix,
    }
    return render(request, "info/t_timetable.html", context)
# student marks
@login_required()
def marks_list(request, stud_id):
    stud = Student.objects.get(
        USN=stud_id,
    )
    ass_list = Assign.objects.filter(class_id_id=stud.class_id)
    sc_list = []
    for ass in ass_list:
        try:
            sc = StudentCourse.objects.get(student=stud, course=ass.course)
        except StudentCourse.DoesNotExist:
            sc = StudentCourse(student=stud, course=ass.course)
            sc.save()
            sc.marks_set.create(type="I", name="Internal test 1")
            sc.marks_set.create(type="I", name="Internal test 2")
            sc.marks_set.create(type="E", name="Assignment 1")
```

```
            sc.marks_set.create(type="E", name="Assignment 2")
            sc.marks_set.create(type="S", name="Semester End Exam")
        sc_list.append(sc)

    return render(request, "info/marks_list.html", {"sc_list": sc_list})
# teacher marks
@login_required()
def t_marks_list(request, assign_id):
    ass = get_object_or_404(Assign, id=assign_id)
    m_list = MarksClass.objects.filter(assign=ass)
    return render(request, "info/t_marks_list.html", {"m_list": m_list})
@login_required()
def t_marks_entry(request, marks_c_id):
    mc = get_object_or_404(MarksClass, id=marks_c_id)
    ass = mc.assign
    c = ass.class_id
    context = {
        "ass": ass,
        "c": c,
        "mc": mc,
    }
    return render(request, "info/t_marks_entry.html", context)
@login_required()
def marks_confirm(request, marks_c_id):
    mc = get_object_or_404(MarksClass, id=marks_c_id)
    ass = mc.assign
    cr = ass.course
    cl = ass.class_id
    for s in cl.student_set.all():
```

```
        mark = request.POST[s.USN]
        sc = StudentCourse.objects.get(course=cr, student=s)
        m = sc.marks_set.get(name=mc.name)
        m.marks1 = mark
        m.save()
    mc.status = True
    mc.save()
    return HttpResponseRedirect(reverse("t_marks_list", args=(ass.id,)))
@login_required()
def edit_marks(request, marks_c_id):
    mc = get_object_or_404(MarksClass, id=marks_c_id)
    cr = mc.assign.course
    stud_list = mc.assign.class_id.student_set.all()
    m_list = []
    for stud in stud_list:
        sc = StudentCourse.objects.get(course=cr, student=stud)
        m = sc.marks_set.get(name=mc.name)
        m_list.append(m)
    context = {
        "mc": mc,
        "m_list": m_list,
    }
    return render(request, "info/edit_marks.html", context)
@login_required()
def student_marks(request, assign_id):
    ass = Assign.objects.get(id=assign_id)
    sc_list = StudentCourse.objects.filter(
        student__in=ass.class_id.student_set.all(), course=ass.course
    )
```

```
    return render(request, "info/t_student_marks.html", {"sc_list": sc_list})
@login_required()
def add_teacher(request):
    if not request.user.is_superuser:
        return redirect("/")
    if request.method == "POST":
        dept = get_object_or_404(Dept, id=request.POST["dept"])
        name = request.POST["full_name"]
        id = request.POST["id"].lower()
        dob = request.POST["dob"]
        sex = request.POST["sex"]
        # Creating a User with teacher username and password format
        # USERNAME: firstname + underscore + unique ID
        # PASSWORD: firstname + underscore + year of birth(YYYY)
        user = User.objects.create_user(
            # username=name.split(" ")[0].lower() + '_' + id,
            # password=name.split(" ")[0].lower() +
            # '_' + dob.replace("-", "")[:4]
            username=id,
            password="project123",
        )
        user.save()
        Teacher(user=user, id=id, dept=dept, name=name, sex=sex, DOB=dob).save()
        return redirect("/")
    all_dept = Dept.objects.order_by("-id")
    context = {"all_dept": all_dept}
    return render(request, "info/add_teacher.html", context)
@login_required()
def add_student(request):
```

```
# If the user is not admin, they will be redirected to home
if not request.user.is_superuser:
    return redirect("/")
if request.method == "POST":
    # Retrieving all the form data that has been inputted
    class_id = get_object_or_404(Class, id=request.POST["class"])
    name = request.POST["full_name"]
    usn = request.POST["usn"]
    dob = request.POST["dob"]
    sex = request.POST["sex"]
    # Creating a User with student username and password format
    # USERNAME: firstname + underscore + last 3 digits of USN
    # PASSWORD: firstname + underscore + year of birth(YYYY)
    user = User.objects.create_user(
        # username=name.split(" ")[0].lower() + '_' +
        # request.POST['usn'][-3:],
        # password=name.split(" ")[0].lower() +
        # '_' + dob.replace("-", "")[:4]
        username=usn,
        password="project123",
    )
    user.save()

    # Creating a new student instance with given data and saving it.
    Student(
        user=user, USN=usn, class_id=class_id, name=name, sex=sex, DOB=dob
    ).save()
    return redirect("/")
```

```
    all_classes = Class.objects.order_by("-id")
    context = {"all_classes": all_classes}
    return render(request, "info/add_student.html", context)
```

FRONTEND

Frontend/info/templates/admin_page.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta name="keywords"
            content="College, Management, Student, Registration, Education, System" />
        <meta charset="utf-8">
        <meta name="viewport"
            content="width=device-width, initial-scale=1, shrink-to-fit=no">
        <meta name="description" content="EduManage">
        <meta name="author" content="Ajmal Basheer & Asish Jose">
        <title>homepage</title>
        {% load static %}
        <!-- Custom LOGO -->
        <link rel="icon" href="{% static 'asishidea.ico' %}" type="image/x-icon">
        <!-- Bootstrap core CSS -->
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
            rel="stylesheet">
        <style>
            body{
                padding:0px
            }
            .jumbotron {
                padding: 0.5rem;
                margin-bottom: 2.5rem;
            }
            .card-img-top {
                height: 160px;
                object-fit: contain;
            }
        </style>
    </head>
    <body>
        <!-- Navigation -->
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
```

```
        <div class="container">
            <a class="navbar-brand" href="{% url 'index' %}">
                <img src="{% static 'info/images/logo.png' %}"
                     alt="Logo"
                     height="30"
                     width=""
                     class="me-2">
            EduManage</a>
            <button class="navbar-toggler"
                    type="button"
                    data-bs-toggle="collapse"
                    data-bs-target="#navbarResponsive"
                    aria-controls="navbarResponsive"
                    aria-expanded="false"
                    aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarResponsive">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'admin:index' %}">College
Administration</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link"
                           href="#"
                           data-bs-toggle="modal"
                           data-bs-target="#logoutModal">Logout</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
    <!-- Page Content -->
    <div class="container mt-5 pt-4">
        <!-- Jumbotron Header -->
        <header class="jumbotron">
            <h1 class="display-5 text-capitalize fst-italic">
                Welcome <strong>{{ request.user }}</strong>
            </h1>
        </header>
        <!-- Page Features -->
        <div class="row text-center justify-content-center">
            <div class="col-lg-3 col-md-6 mb-4">
                <div class="card h-100">
```

```
                <a class="px-2 py-3" href="{% url 'add_teacher' %}">
                  <img class="card-img-top hiwi"
                      src="{% static 'info/images/teacher.png' %}"
                      alt="Add Teacher"
                      width=""
                      height="">
                </a>
                <div class="card-body">
                  <h4 class="card-title">Add Teacher</h4>
                  <p class="card-text">
                    Enter the details of new faculty to add a new teacher to database. Make sure
to correctly input values.
                  </p>
                </div>
              </div>
            </div>
            <div class="col-lg-3 col-md-6 mb-4">
              <div class="card h-100">
                <a class="px-2 py-3" href="{% url 'add_student' %}">
                  <img class="card-img-top hiwi"
                      src="{% static 'info/images/student.png' %}"
                      alt="Add Student"
                      width=""
                      height="">
                </a>
                <div class="card-body">
                  <h4 class="card-title">Add Student</h4>
                  <p class="card-text">
                    Enter the details of a student to enroll a new student.
                    Fill the details carefully as they are important for academics.
                  </p>
                </div>
              </div>
            </div>
          </div>
          <!-- /.row -->
        </div>
        <!-- /.container -->
        <!-- Logout Modal-->
        <div class="modal fade"
            id="logoutModal"
            tabindex="-1"
            aria-labelledby="exampleModalLabel"
            aria-hidden="true">
          <div class="modal-dialog">
```

```
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
                    <button class="btn-close"
                            type="button"
                            data-bs-dismiss="modal"
                            aria-label="Close"></button>
                </div>
                <div class="modal-body">Select "Logout" below if you are ready to end your
current session.</div>
                <div class="modal-footer">
                    <button class="btn btn-secondary" type="button" data-bs-
dismiss="modal">Cancel</button>
                    <form action="{% url 'logout' %}" method="post">
                        {% csrf_token %}
                        <button class="btn btn-primary" type="submit">Logout</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
    <!-- Bootstrap core JavaScript -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
  </body>
</html>
```

Frontend/info/templates/att_detail.html

```
 {% extends "info/base.html" %}
{% block title %}
  Attendance Detail
{% endblock title %}
{% block content %}
  <div class="card mb-3">
    <div class="card-header">
      <i class="fas fa-table"></i>
      <strong>{{ cr.name }}</strong>
    </div>
    <div class="card-body">
      <div class="table-responsive">
        <table class="table table-bordered"
              id="dataTable"
              width="100%"
              cellspacing="0">
            <thead>
```

```
                <tr>
                    <th>#</th>
                    <th>Date</th>
                    <th>Day</th>
                    <th>Status</th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                {% for a in att_list %}
                    <tr class="row100 body">
                        <td>{{ forloop.counter }}</td>
                        <td>{{ a.date }}</td>
                        <td>{{ a.date|date:"l" }}</td>
                        {% if a.status %}
                            <td class="p-3 mb-2 bg-success text-white">
                                Present <span class="glyphicon glyphicon-thumbs-up"></span>
                            </td>
                        {% else %}
                            <td class="p-3 mb-2 bg-danger text-white">
                                Absent <span class="glyphicon glyphicon-thumbs-down"></span>
                            </td>
                        {% endif %}
                    </tr>
                {% empty %}
                    <p>Student has no attendance</p>
                {% endfor %}
            </tbody>
        </table>
    </div>
  </div>
</div>
{% endblock content %}
```

Frontend/info/templates/attendance.html

```
{% extends "info/base.html" %}
{% block title %}
  Attendance
{% endblock title %}
{% load static %}
{% block content %}
  <div class="card mb-3">
    <div class="card-header">
      <i class="fas fa-table"></i>
      <b>Attendance</b>
```

```
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-bordered text-center"
                        id="dataTable"
                        width="100%"
                        cellspacing="0">
                    <thead class="thead-light ">
                        <tr>
                            <th>Course ID</th>
                            <th>Course name</th>
                            <th>Attended classes</th>
                            <th>Total classes</th>
                            <th>Attendance %</th>
                            <th>Classes to attend</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for a in att_list %}
                            <tr>
                                <td>{{ a.course_id }}</td>
                                <td>
                                    <a href="{% url 'attendance_detail' a.student.USN a.course.id %}">{{
a.course.name }}</a>
                                </td>
                                <td>{{ a.att_class }}</td>
                                <td>{{ a.total_class }}</td>
                                {% if a.attendance < 75 %}
                                    <td class="p-3 mb-2 bg-danger text-white">{{ a.attendance }}</td>
                                {% else %}
                                    <td class="p-3 mb-2 bg-success text-white">{{ a.attendance }}</td>
                                {% endif %}
                                <td>{{ a.classes_to_attend }}</td>
                            </tr>
                        {% empty %}
                            <p>Student has no courses</p>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
{% endblock content %}
```

Frontend/info/templates/base.html

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1" />
      <meta name="keywords"
           content="College, Management, Student, Registration, Education, System" />
      <meta name="description" content="EduManage" />
      <meta name="author" content="Ajmal Basheer & Asish Jose" />
      <title>
         {% block title %}
         {% endblock title %}
      </title>
      {% load static %}
      <!-- Custom LOGO -->
      <link rel="icon" href="{% static 'asishidea.ico' %}" type="image/x-icon">
      <!-- Bootstrap core CSS-->
      <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
           rel="stylesheet" />
      <!-- Custom fonts for this template-->
      <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css"
           rel="stylesheet"
           type="text/css" />
      <!-- Page level plugin CSS-->
      <!-- Custom styles for this template-->
      <link href="https://cdnjs.cloudflare.com/ajax/libs/startbootstrap-sb-admin/5.0.2/css/sb-
admin.css"
           rel="stylesheet" />
      {% block css %}
      {% endblock css %}
      <style>
         .custombgcolor{
            background-color: #212529 !important;
         }
         .sidebar {
            min-height: auto !important;
            height: auto !important;
         }
         .fixed-top {
            position: fixed;
            top: 0;
            right: 0;
            left: 0;
            z-index: 1030;
```

```
            }

            #wrapper {
                display: flex;
                padding-top: 56px;
                min-height:100vh !important;
            }

            #content-wrapper {
                flex: 1;
                overflow-y: auto;
            }
        </style>
    </head>
    <body id="page-top">
        <nav class="navbar navbar-expand navbar-dark bg-dark fixed-top custombgcolor">
            <a class="navbar-brand mr-1" href="{% url 'index' %}">
                <img src="{% static 'info/images/logo.png' %}"
                     alt="Logo"
                     height="30"
                     width=""
                     class="me-2">
            EduManage</a>
            <button class="btn btn-link btn-sm text-white order-1 order-sm-0"
                    id="sidebarToggle"
                    href="#">
                <i class="fas fa-bars"></i>
            </button>
            <!-- Navbar -->
            <div class="collapse navbar-collapse custombgcolor" id="navbarResponsive">
                <ul class="navbar-nav ml-auto custombgcolor">
                    <li class="nav-item">
                        {% if request.user.is_student %}
                            <a class="nav-link text-capitalize" href="{% url 'index' %}">{{
request.user.student.name }}</a>
                        {% elif request.user.is_teacher %}
                            <a class="nav-link text-capitalize" href="{% url 'index' %}">{{
request.user.teacher.name }}</a>
                        {% endif %}
                    </li>
                    <li class="nav-item">
                        <a class="nav-link"
                           href="{% url 'logout' %}"
                           data-toggle="modal"
                           data-target="#logoutModal">Logout</a>
```

```
            </li>
        </ul>
    </div>
</nav>
<div id="wrapper">
    <!-- Sidebar -->
    <ul class="sidebar navbar-nav">
        <li class="nav-item">
            <a class="nav-link" href="{% url 'index' %}">
                <span>Home</span>
            </a>
        </li>
        {% if request.user.is_student %}
            <li class="nav-item">
                <a class="nav-link"
                  href="{% url 'attendance' request.user.student.USN %}">
                   <span>Attendance</span>
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link"
                  href="{% url 'marks_list' request.user.student.USN %}">
                   <span>Marks</span>
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link"
                  href="{% url 'timetable' request.user.student.class_id_id %}">
                   <span>Time Table</span>
                </a>
            </li>
        {% elif request.user.is_teacher %}
            <li class="nav-item">
                <a class="nav-link" href="{% url 't_clas' request.user.teacher.id 1 %}">
                   <span>Attendance</span>
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 't_clas' request.user.teacher.id 2 %}">
                   <span>Marks</span>
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link"
                  href="{% url 't_timetable' request.user.teacher.id %}">
```

```
                <span>Time Table</span>
            </a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{% url 't_clas' request.user.teacher.id 3 %}">
                <span>Reports</span>
            </a>
        </li>
    {% endif %}
</ul>
<div id="content-wrapper">
    <div class="container-fluid">
        <!-- Breadcrumbs-->
        <!-- Page Content -->
        {% block content %}
        {% endblock content %}
    </div>
    <!-- /.container-fluid -->
    <!-- Sticky Footer -->
</div>
<!-- /.content-wrapper -->
</div>
<!-- /#wrapper -->
<!-- Scroll to Top Button-->
<a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
</a>
<!-- Logout Modal-->
<div class="modal fade"
     id="logoutModal"
     tabindex="-1"
     role="dialog"
     aria-labelledby="exampleModalLabel"
     aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
                <button class="close" type="button" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">×</span>
                </button>
            </div>
            <div class="modal-body">Select "Logout" below if you are ready to end your
current session.</div>
            <div class="modal-footer">
```

```
                    <button class="btn btn-secondary" type="button" data-
dismiss="modal">Cancel</button>
                    <form action="{% url 'logout' %}" method="post">
                        {% csrf_token %}
                        <button class="btn btn-primary" type="submit">Logout</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
    <!-- Bootstrap core JavaScript-->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
    <!-- Core plugin JavaScript-->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-
easing/1.4.1/jquery.easing.min.js"></script>
    <!-- Custom scripts for all pages-->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/startbootstrap-sb-admin/5.0.2/js/sb-
admin.min.js"></script>
    {% block scripts %}
    {% endblock scripts %}
  </body>
</html>
```

Frontend/info/templates/edit_marks.html

```
{% extends "info/base.html" %}
{% block content %}
  <form action="{% url 'marks_confirm' mc.id %}" method="post">
    {% csrf_token %}
    <div class="card mb-3">
      <div class="card-header">
        <i class="fas fa-table"></i>
      </div>
      <div class="card-body">
        <div class="table-responsive">
          <table class="table table-bordered"
              id="dataTable"
              width="100%"
              cellspacing="0">
            <thead>
              <tr>
                <th>Student Name</th>
                <th>Total Marks</th>
                <th>Enter Marks</th>
```

```
                    </tr>
                </thead>
                <tbody>
                    {% for m in m_list %}
                        <tr>
                            <td>{{ m.studentcourse.student.name }}</td>
                            <td>{{ m.total_marks }}</td>
                            <td>
                                <input type="number"
                                        name="{{ m.studentcourse.student.USN }}"
                                        min="0"
                                        max="{{ m.total_marks }}"
                                        value="{{ m.marks1 }}">
                            </td>
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
</div>
<input class="btn btn-success" type="submit" value="Submit">
</form>
{% endblock content %}
```

Frontend/info/templates/homepage.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <meta name="description" content="EduManage" />
        <meta name="author" content="Ajmal Basheer & Asish Jose" />
        <meta name="keywords"
            content="College, Management, Student, Registration, Education, System" />
        <title>homepage</title>
        {% load static %}
        <!-- Custom LOGO -->
        <link rel="icon" href="{% static 'asishidea.ico' %}" type="image/x-icon">
        <!-- Bootstrap CSS -->
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
            rel="stylesheet">
        <!-- Custom styles -->
        <style>
            body{
```

```
            padding:0px
        }
        .jumbotron {
            padding: 0.5rem;
            margin-bottom: 0.5rem;
        }
        .card-img-top {
            height: 160px;
            object-fit: contain;
        }
    </style>
</head>
<body>
    <!-- Navigation -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
        <div class="container">
            <a class="navbar-brand" href="{% url 'index' %}">
                <img src="{% static 'info/images/logo.png' %}"
                     alt="Logo"
                     height="30"
                     width=""
                     class="me-2">
            EduManage</a>
            <button class="navbar-toggler"
                    type="button"
                    data-bs-toggle="collapse"
                    data-bs-target="#navbarResponsive"
                    aria-controls="navbarResponsive"
                    aria-expanded="false"
                    aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarResponsive">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link text-capitalize">{{ request.user.student.name }}</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link"
                           href="#"
                           data-bs-toggle="modal"
                           data-bs-target="#logoutModal">Logout</a>
                    </li>
                </ul>
            </div>
```

```
        </div>
    </nav>
    <!-- Page Content -->
    <div class="container mt-5 pt-4">
        <!-- Jumbotron Header -->
        <header class="jumbotron">
            <h1 class="display-3 text-capitalize fst-italic">
                Welcome <strong>{{ request.user.student.name }}</strong>,
            </h1>
        </header>
        <!-- Page Features -->
        <div class="row text-center">
            <div class="col-lg-4 col-md-6 mb-4">
                <div class="card h-100">
                    <a href="{% url 'attendance' request.user.student.USN %}">
                        <img class="card-img-top"
                             src="{% static 'info/images/attendance2.jpg' %}"
                             alt="Attendance"
                             height=""
                             width="">
                    </a>
                    <div class="card-body">
                        <h4 class="card-title">Attendance</h4>
                        <p class="card-text">
                            View the attendance status for each of your courses. The
                            attendance of each course is also displayed as list of classes
                            that were conducted.
                        </p>
                    </div>
                    <div class="card-footer">
                        <a class="btn btn-primary"
                           role="button"
                           href="{% url 'attendance' request.user.student.USN %}">View
Attendance</a>
                    </div>
                </div>
            </div>
            <div class="col-lg-4 col-md-6 mb-4">
                <div class="card h-100">
                    <a href="{% url 'marks_list' request.user.student.USN %}">
                        <img class="card-img-top"
                             src="{% static 'info/images/marks.jpg' %}"
                             alt="Marks"
                             height=""
                             width="">
```

```
                </a>
                <div class="card-body">
                  <h4 class="card-title">Marks</h4>
                  <p class="card-text">
                    View the marks obtained for each of your courses. These include
                    the marks of 2 internal assessment, 2 Assignment and the Semester
                    End Exam
                  </p>
                </div>
                <div class="card-footer">
                  <a class="btn btn-primary"
                    role="button"
                    href="{% url 'marks_list' request.user.student.USN %}">View Marks</a>
                </div>
              </div>
            </div>
            <div class="col-lg-4 col-md-6 mb-4">
              <div class="card h-100">
                <a href="{% url 'timetable' request.user.student.class_id_id %}">
                  <img class="card-img-top"
                     src="{% static 'info/images/timetable.png' %}"
                     alt="Timetable"
                     height=""
                     width="">
                </a>
                <div class="card-body">
                  <h4 class="card-title">TimeTable</h4>
                  <p class="card-text">
                    View the timetable in a tabular form. The timetable displays all
                    the courses of the student and the time and day at which they
                    are conducted.
                  </p>
                </div>
                <div class="card-footer">
                  <a class="btn btn-primary"
                    role="button"
                    href="{% url 'timetable' request.user.student.class_id_id %}">View
TimeTable</a>
                </div>
              </div>
            </div>
          </div>
        </div>
        <!-- Logout Modal -->
        <div class="modal fade"
```

```
            id="logoutModal"
            tabindex="-1"
            aria-labelledby="exampleModalLabel"
            aria-hidden="true">
           <div class="modal-dialog">
              <div class="modal-content">
                 <div class="modal-header">
                    <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
                    <button type="button"
                          class="btn-close"
                          data-bs-dismiss="modal"
                          aria-label="Close"></button>
                 </div>
                 <div class="modal-body">Select "Logout" below if you are ready to end your
current session.</div>
                 <div class="modal-footer">
                    <button class="btn btn-secondary" type="button" data-bs-
dismiss="modal">Cancel</button>
                    <form action="{% url 'logout' %}" method="post">
                       {% csrf_token %}
                       <button class="btn btn-primary" type="submit">Logout</button>
                    </form>
                 </div>
              </div>
           </div>
        </div>
        <!-- Bootstrap JS -->
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
    </body>
</html>
```

Frontend/info/templates/login.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <meta name="description" content="EduManage">
        <meta name="keywords"
             content="College, Management, Student, Registration, Education, System" />
        <meta name="author" content="Ajmal Basheer & Asish Jose">
        <title>Login - EduManage</title>
```

```
    {% load static %}
    <link rel="icon" href="{% static 'asishidea.ico' %}" type="image/x-icon">
    <!-- Bootstrap 5 CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
        rel="stylesheet">
    <!-- FontAwesome for icons -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.6.0/css/all.min.css"
        rel="stylesheet">
    <style>
    .py-4 {
        padding-top: 1rem !important;
        padding-bottom: 1rem !important;
    }
    body {
        background-color: #f8f9fa;
    }
    .btn-custom {
        background-color: #343a40;
        border: none;
    }
    .btn-custom:hover {
        background-color: #23272b;
    }
    </style>
</head>
<body>
    <header class="bg-dark text-white py-4 mb-4">
        <h1 class="display-4 text-center">
            <img src="{% static 'info/images/logo.png' %}"
                alt="Logo"
                height="75px"
                width=""
                class="me-2">
            EduManage
        </h1>
    </header>
    <br />
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-6 col-lg-4">
                <div class="card border-0 shadow">
                    <div class="card-header bg-light border-0 py-3">
                        <h4 class="text-center mb-0">Login</h4>
                    </div>
                    <div class="card-body p-4">
```

```
                    <form method="post">
                        {% csrf_token %}
                        {% if form.non_field_errors %}
                            <div class="alert alert-danger" role="alert">
                                {% for error in form.non_field_errors %}{{ error }}{% endfor %}
                            </div>
                        {% endif %}
                        {% for field in form %}
                            <div class="mb-3">
                                {% if field.errors %}<div class="alert alert-danger py-2"
role="alert">{{ field.errors }}</div>{% endif %}
                                {% if field.name == 'password' %}
                                    <div class="input-group">
                                        <input type="{{ field.field.widget.input_type }}"
                                               name="{{ field.name }}"
                                               id="{{ field.name }}"
                                               class="form-control {% if field.errors %}is-invalid{% endif
%}"
                                               placeholder="{{ field.label }}"
                                               autocomplete="current-password"
                                               {% if field.field.required %}required{% endif %}>
                                        <span class="input-group-text" id="togglePassword">
                                            <i class="fas fa-eye"></i>
                                        </span>
                                    </div>
                                {% else %}
                                    <input type="{{ field.field.widget.input_type }}"
                                           name="{{ field.name }}"
                                           id="{{ field.name }}"
                                           class="form-control {% if field.errors %}is-invalid{% endif %}"
                                           placeholder="{{ field.label }}"
                                           autocomplete="username"
                                           {% if field.field.required %}required{% endif %}>
                                {% endif %}
                            </div>
                        {% endfor %}
                        <div class="d-grid">
                            <button class="btn btn-custom btn-lg text-white"
type="submit">Login</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
```

```
        </div>
        <script>
        document.getElementById('togglePassword').addEventListener('click', function () {
            var passwordField = document.getElementById('password');
            var icon = this.querySelector('i');
            if (passwordField.type === 'password') {
                passwordField.type = 'text';
                icon.classList.remove('fa-eye');
                icon.classList.add('fa-eye-slash');
            } else {
                passwordField.type = 'password';
                icon.classList.remove('fa-eye-slash');
                icon.classList.add('fa-eye');
            }
        });
        </script>
        <!-- Bootstrap 5 JS Bundle with Popper -->
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
    </body>
</html>
```

Frontend/info/templates/marks_list.html

```
{% extends "info/base.html" %}
{% block title %}
    Marks
{% endblock title %}
{% load static %}
{% block content %}
    <div class="card mb-3">
        <div class="card-header">
            <i class="fas fa-table"></i>
            <b>Marks</b>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-bordered"
                      id="dataTable"
                      width="100%"
                      cellspacing="0">
                    <thead>
                      <tr>
                          <th>Course ID</th>
                          <th>Course name</th>
                          <th>Internals 1</th>
```

```
                <th>Internals 2</th>
                <th>Assignment 1</th>
                <th>Assignment 2</th>
                <th>SEE-Final</th>
            </tr>
        </thead>
        <tbody>
            {% for sc in sc_list %}
                <tr>
                    <td>{{ sc.course_id }}</td>
                    <td>{{ sc.course.name }}</td>
                    <td>
                        {% for m in sc.marks_set.all %}
                            {% if m.name == "Internal test 1" %}{{ m.marks1 }}{% endif %}
                        {% endfor %}
                    </td>
                    <td>
                        {% for m in sc.marks_set.all %}
                            {% if m.name == "Internal test 2" %}{{ m.marks1 }}{% endif %}
                        {% endfor %}
                    </td>
                    <td>
                        {% for m in sc.marks_set.all %}
                            {% if m.name == "Assignment 1" %}{{ m.marks1 }}{% endif %}
                        {% endfor %}
                    </td>
                    <td>
                        {% for m in sc.marks_set.all %}
                            {% if m.name == "Assignment 2" %}{{ m.marks1 }}{% endif %}
                        {% endfor %}
                    </td>
                    <td>
                        {% for m in sc.marks_set.all %}
                            {% if m.name == "Semester End Exam" %}{{ m.marks1 }}{% endif
%}
                        {% endfor %}
                    </td>
                </tr>
            {% empty %}
                <p>Student has no courses</p>
            {% endfor %}
        </tbody>
    </table>
  </div>
</div>
```

```
    </div>
{% endblock content %}
```

Frontend/info/templates/t_report.html

```
{% extends "info/base.html" %}
{% load static %}
{% block content %}
    <div class="card mb-3">
        <div class="card-header">
            <i class="fas fa-table"></i>
            <b>Marks</b>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-bordered"
                        id="dataTable"
                        width="100%"
                        cellspacing="0">
                    <thead>
                        <tr>
                            <th class="t_report-t-head">Student USN</th>
                            <th class="t_report-t-head">Student Name</th>
                            <th class="t_report-t-head">Attendance</th>
                            <th class="t_report-t-head">Internal Assessment</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for sc in sc_list %}
                            <tr>
                                <td>{{ sc.student_id }}</td>
                                <td>{{ sc.student.name }}</td>
                                {% if sc.get_attendance < 75 %}
                                    <td class="p-3 mb-2 bg-danger text-white">{{ sc.get_attendance }}</td>
                                {% else %}
                                    <td class="p-3 mb-2 bg-success text-white">{{ sc.get_attendance }}</td>
                                {% endif %}
                                {% if sc.get_cie < 20 %}
                                    <td class="p-3 mb-2 bg-danger text-white">{{ sc.get_cie }}</td>
                                {% else %}
                                    <td class="p-3 mb-2 bg-success text-white">{{ sc.get_cie }}</td>
                                {% endif %}
                            </tr>
                        {% empty %}
                            <p>Student has no courses</p>
                        {% endfor %}
```

```
                </tbody>
            </table>
        </div>
    </div>
  </div>
{% endblock content %}
```

Frontend/info/templates/t_students.html

```
{% extends "info/base.html" %}
{% load static %}
{% block content %}
  <div class="card mb-3">
    <div class="card-header">
      <i class="fas fa-table"></i>
      <b>Attendance</b>
    </div>
    <div class="card-body">
      <div class="table-responsive">
        <table class="table table-bordered"
            id="dataTable"
            width="100%"
            cellspacing="0">
          <thead>
            <tr>
              <th>USN</th>
              <th>Student name</th>
              <th>Attended classes</th>
              <th>Total classes</th>
              <th>Attendance %</th>
              <th>Classes to attend</th>
            </tr>
          </thead>
          <tbody>
            {% for a in att_list %}
              <tr>
                <td>{{ a.student_id }}</td>
                <td>
                  <a href="{% url 't_attendance_detail' a.student.USN a.course_id %}">{{
a.student.name }}</a>
                </td>
                <td>{{ a.att_class }}</td>
                <td>{{ a.total_class }}</td>
                {% if a.attendance < 75 %}
                  <td class="p-3 mb-2 bg-danger text-white">{{ a.attendance }}</td>
                {% else %}
```

```
                        <td class="p-3 mb-2 bg-success text-white">{{ a.attendance }}</td>
                    {% endif %}
                    <td>{{ a.classes_to_attend }}</td>
                </tr>
            {% empty %}
                <p>No students</p>
            {% endfor %}
        </tbody>
    </table>
  </div>
 </div>
</div>
{% endblock content %}
```

Frontend/info/templates/timetable.html

```
{% extends "info/base.html" %}
{% block title %}
  TimeTable
{% endblock title %}
{% load static %}
{% block content %}
  <div class="card mb-3">
    <div class="card-header">
      <h2 class="text-center">
        <i class="fas fa-calendar-days"></i>
        Timetable
      </h2>
    </div>
    <div id="card-body">
      <div class="table-responsive">
        <table class="table table-bordered table-striped table-condenced"
            display="fixed"
            width="100%"
            cellspacing="0"
            id="dataTable">
          <thead>
            <tr>
              <th></th>
              <th>8:30 - 9:45</th>
              <th>9:45 - 10:45</th>
              <th>Break</th>
              <th>11:10 - 12:10</th>
              <th>12:10 - 1:10</th>
              <th>Lunch</th>
              <th>1:50 - 2:40</th>
```

```
                <th>2:40 - 3:30</th>
            </tr>
        </thead>
        <tbody>
            {% for i in matrix %}
                <tr>
                    {% for j in i %}
                        {% if forloop.counter == 1 %}
                            <td>
                                <b>{{ j }}</b>
                            </td>
                        {% else %}
                            <td>
                                {% if j.assign.course_id and j.assign.course.shortname %}
                                    {{ j.assign.course_id }}-{{ j.assign.course.shortname }}
                                {% elif j.assign.course_id %}
                                    {{ j.assign.course_id }}
                                {% elif j.assign.course.shortname %}
                                    {{ j.assign.course.shortname }}
                                {% endif %}
                            </td>
                        {% endif %}
                    {% endfor %}
                </tr>
            {% endfor %}
        </tbody>
    </table>
  </div>
 </div>
</div>
{% endblock content %}
```

10.SYSTEM TESTING

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The completion of a system will be achieved only after it has been thoroughly tested. Though this gives a feel the project is completed, there cannot be any project without going through this stage. Hence in this stage it is decided whether the project can undergo the real time environment execution without any break downs, therefore a package can be rejected even at this stage.

Unit testing:

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.

Integration testing:

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements.

System testing:

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements, integration testing passed components are taken as input.

White box testing:

White box testing is an approach that allows testers to inspect and verify the inner workings of a software system-its code, infrastructure, and integrations with external systems

.

Black box testing:

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test.

Verification and validation:

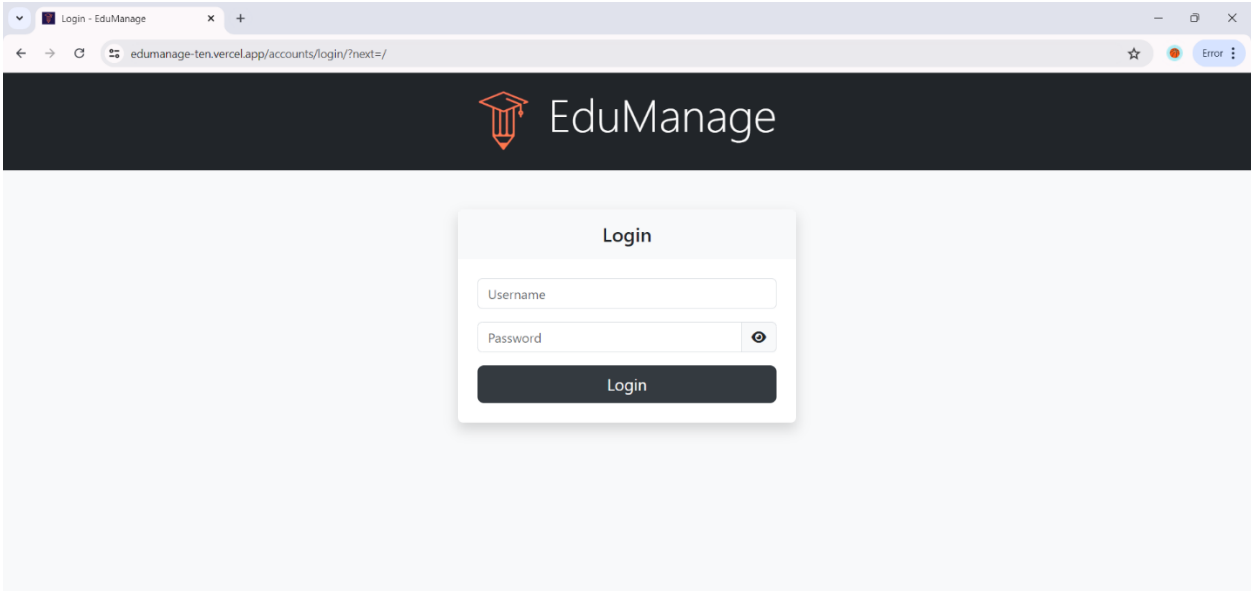
Validation is the process of checking whether the specification captures the customer's requirements, while verification is the process of checking that the software meets specifications. Verification includes all the activities associated with the producing high-quality software.

Acceptance Testing:

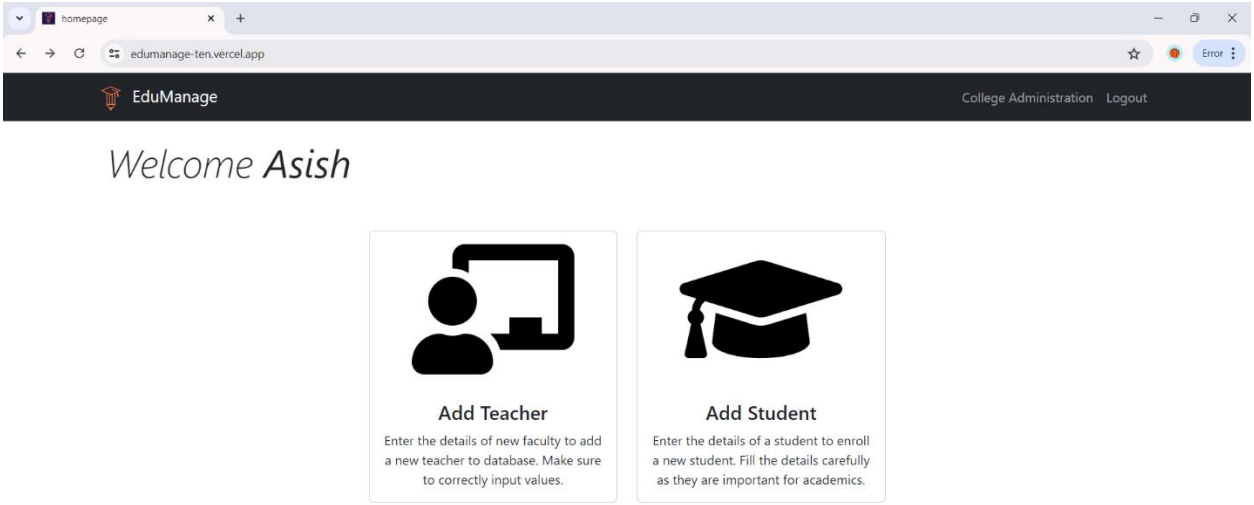
Acceptance testing performed by the customer is known as user acceptance testing (UAT). Since our project is on college management system, the teachers are a key stakeholder. Hence, it was important to allow the teachers to test the software and get their approval as they intend to use the software the most. Therefore, we met and a gave a demonstration of the project to our teacher Prof. Prakasam Sir. We showed him all the features and functionality of the website. He went through all the different web pages and asked several questions on the working of the code. Overall, he was happy with the working and results of the software

11. SCREENSHOTS

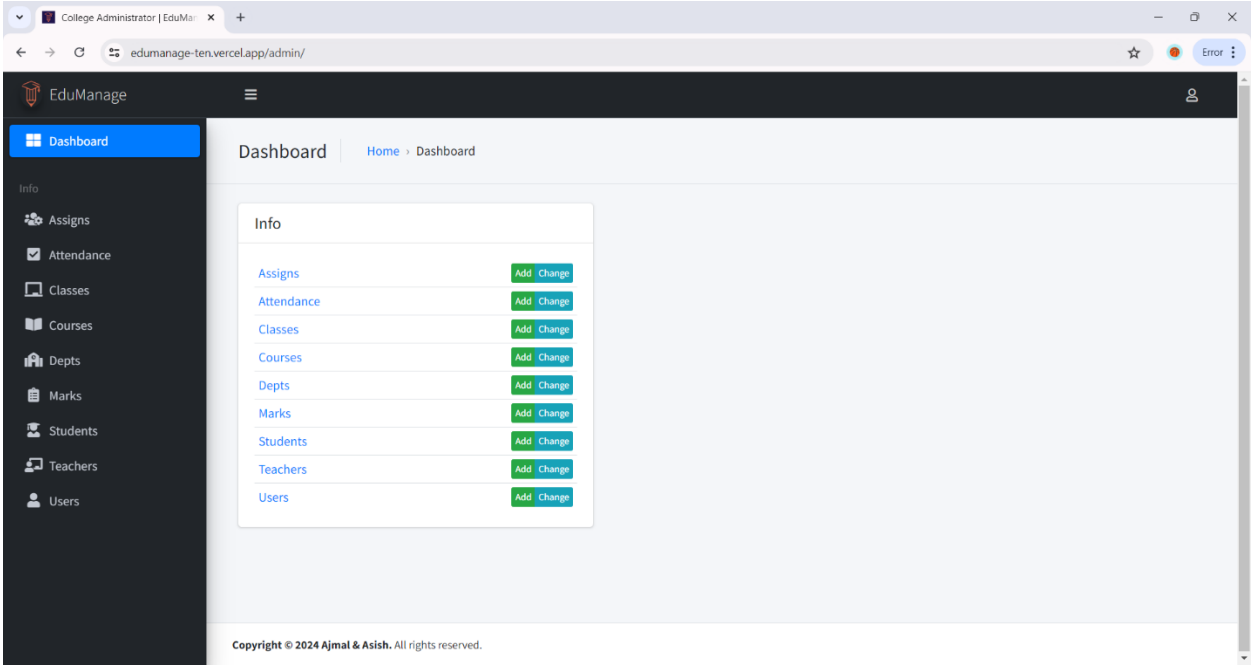
Login



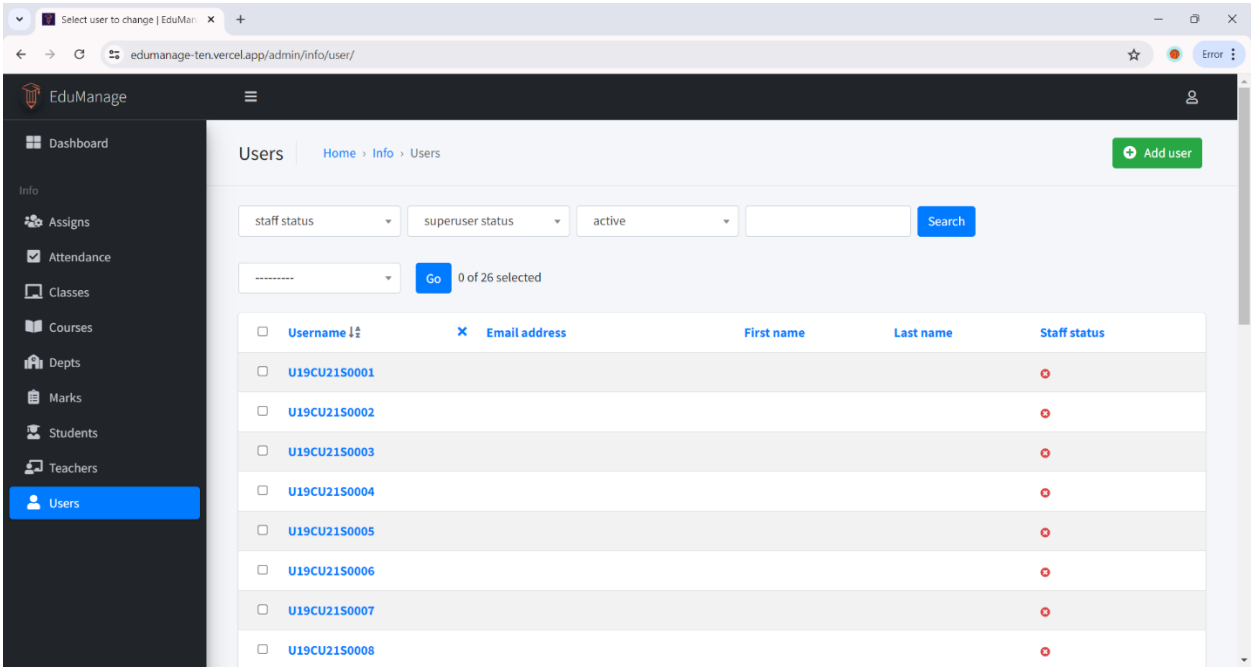
Admin Dashboard



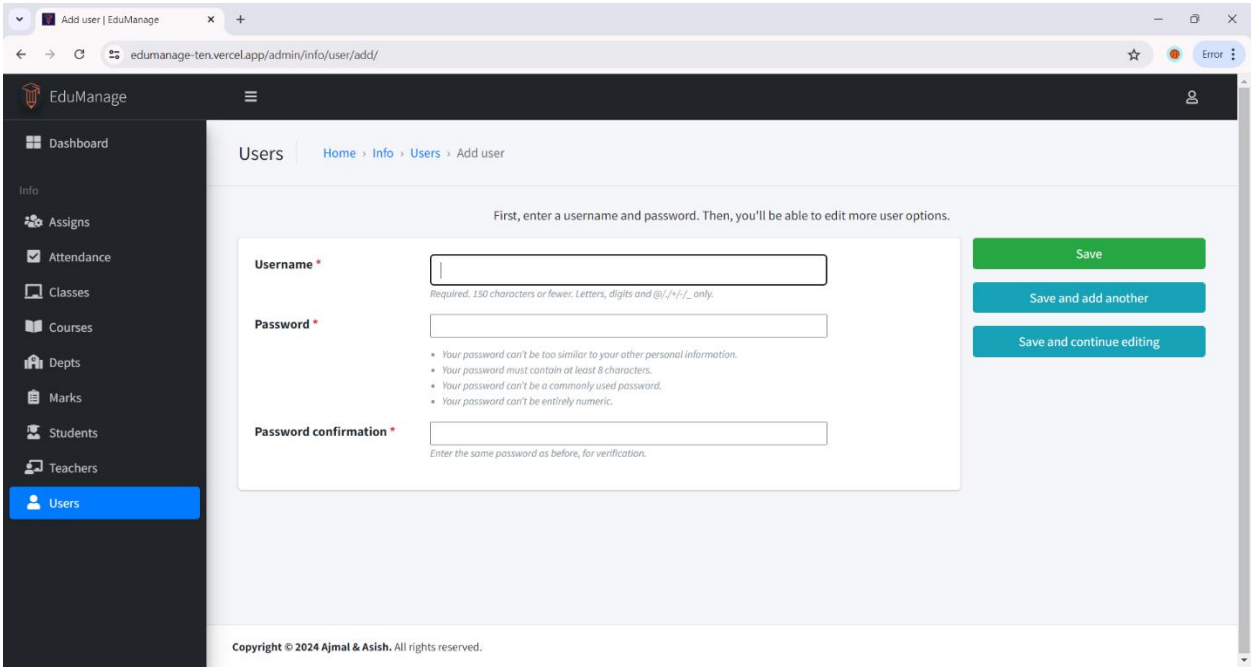
Administration Page



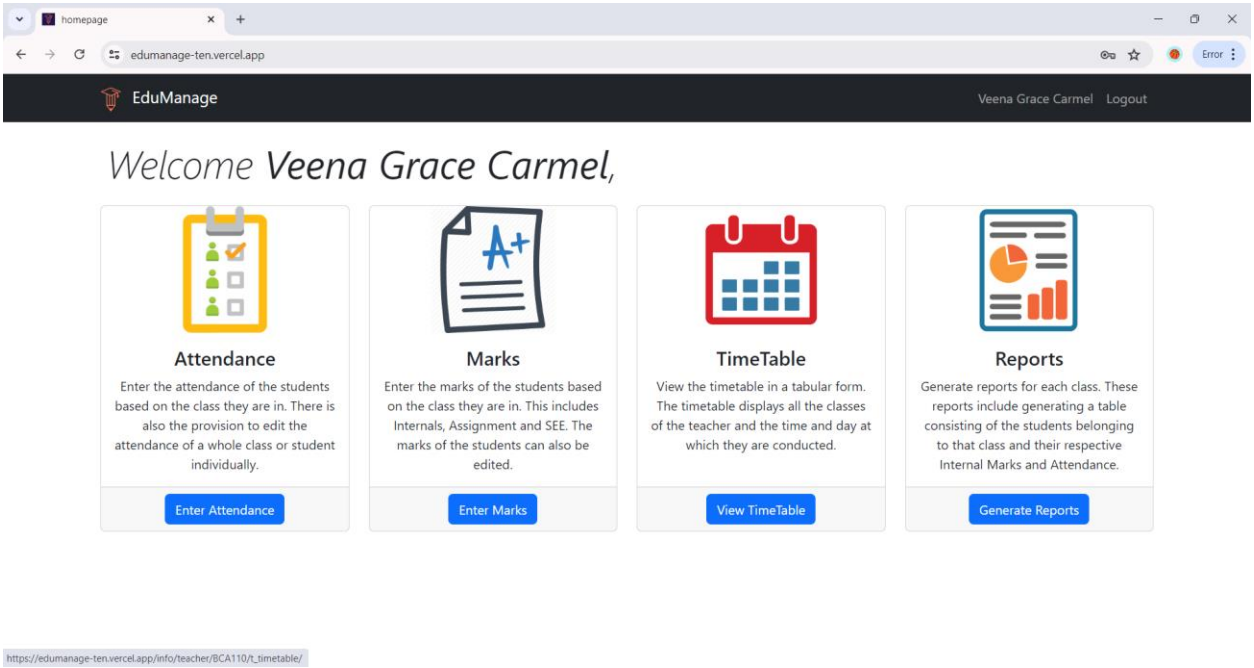
Users Page



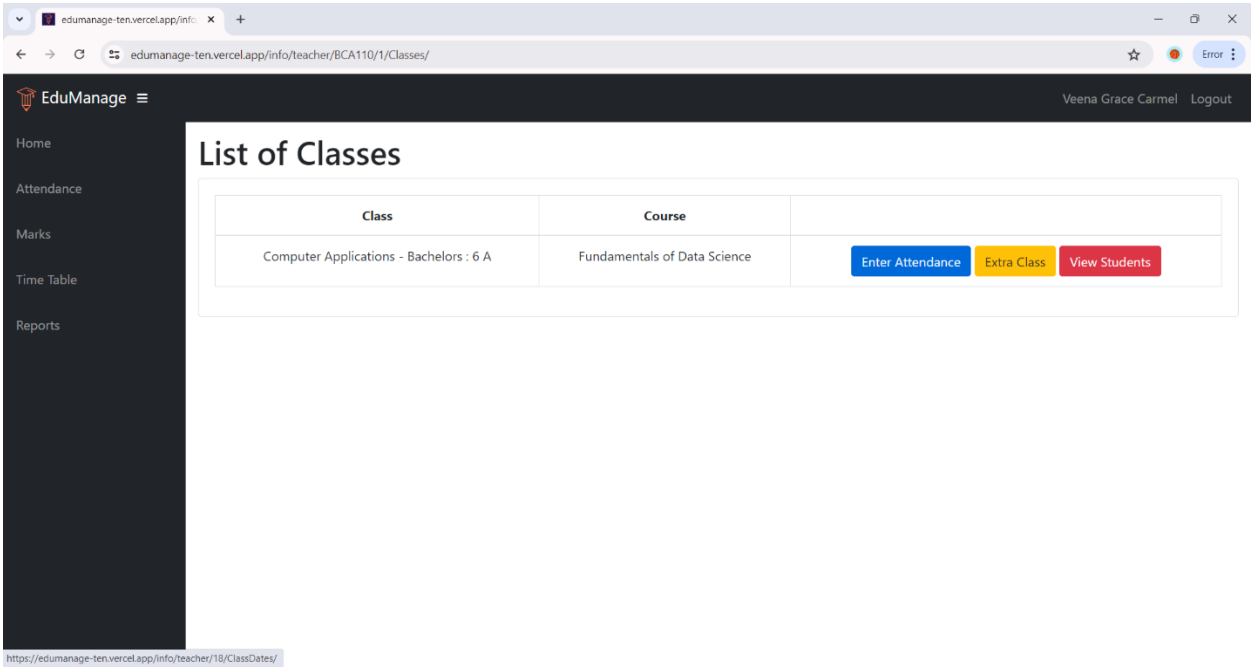
Add User Page



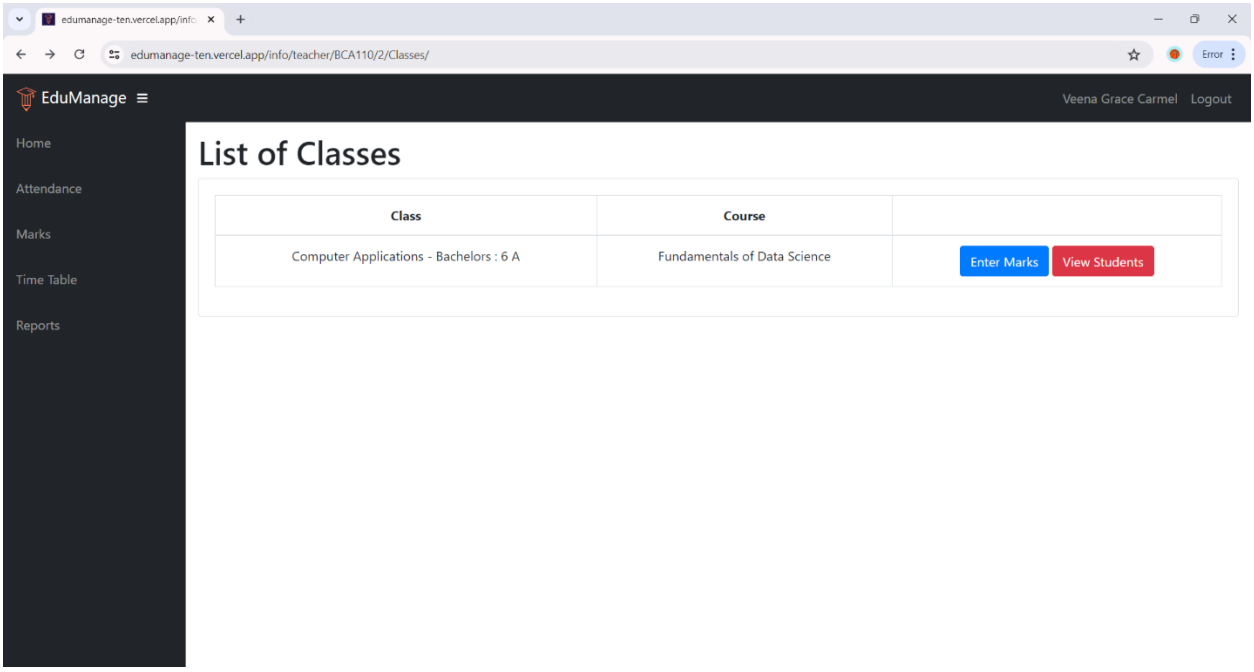
Teachers Dashboard



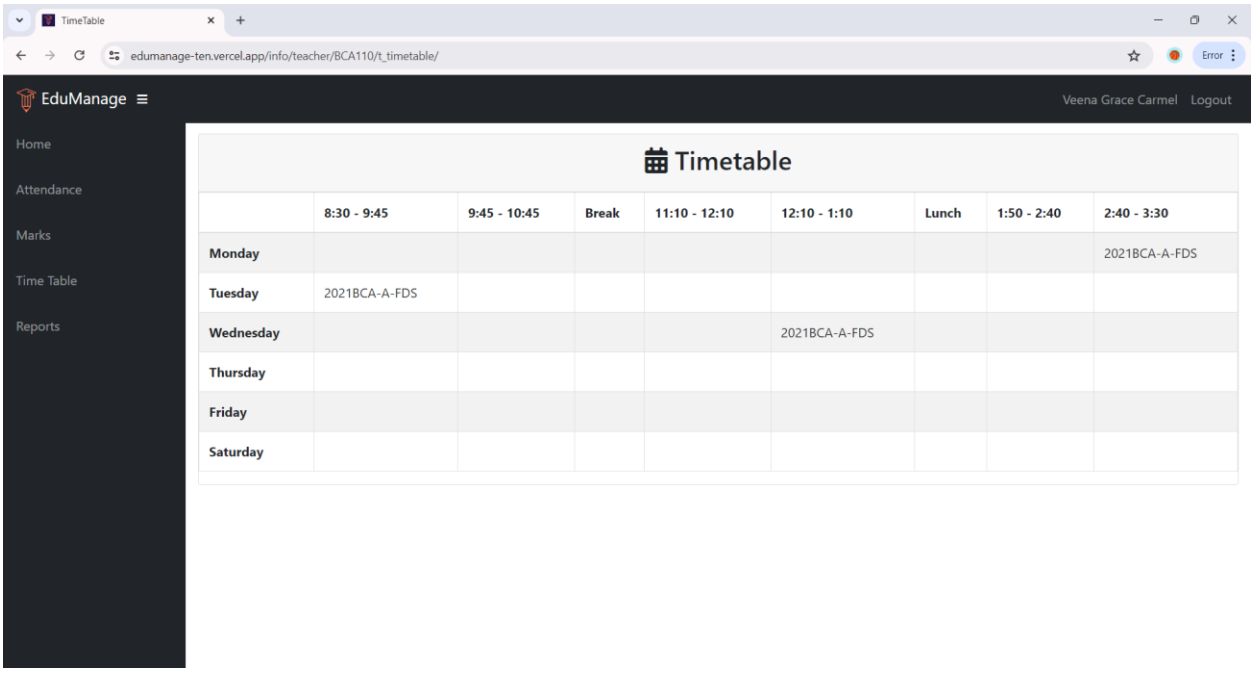
Attendance Page



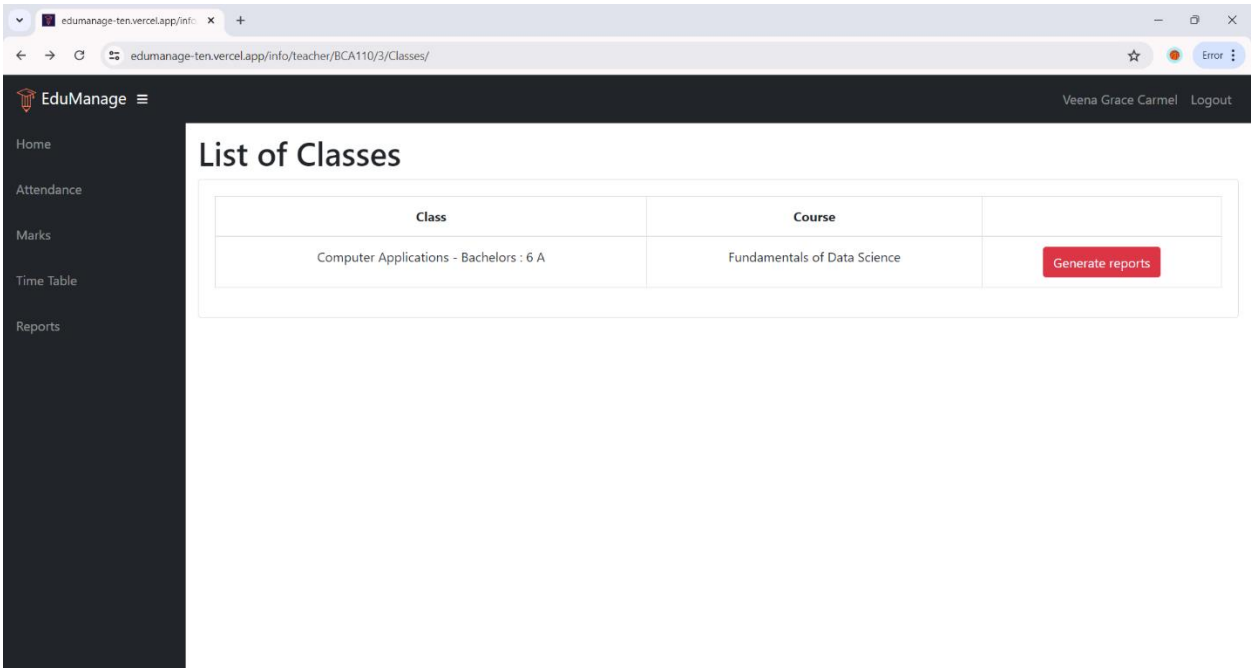
Marks Page



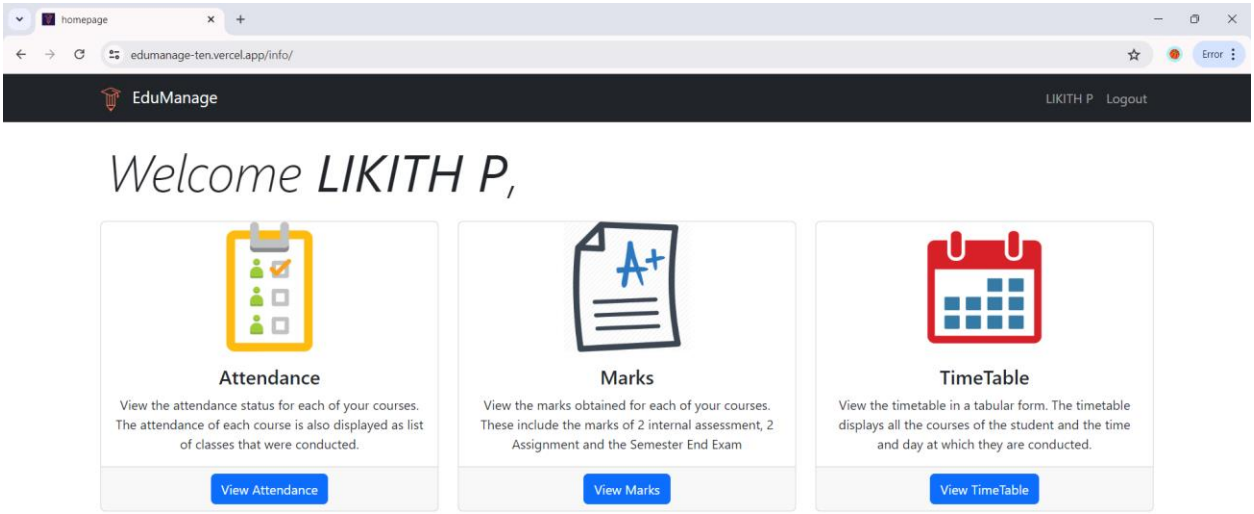
Timetable Page



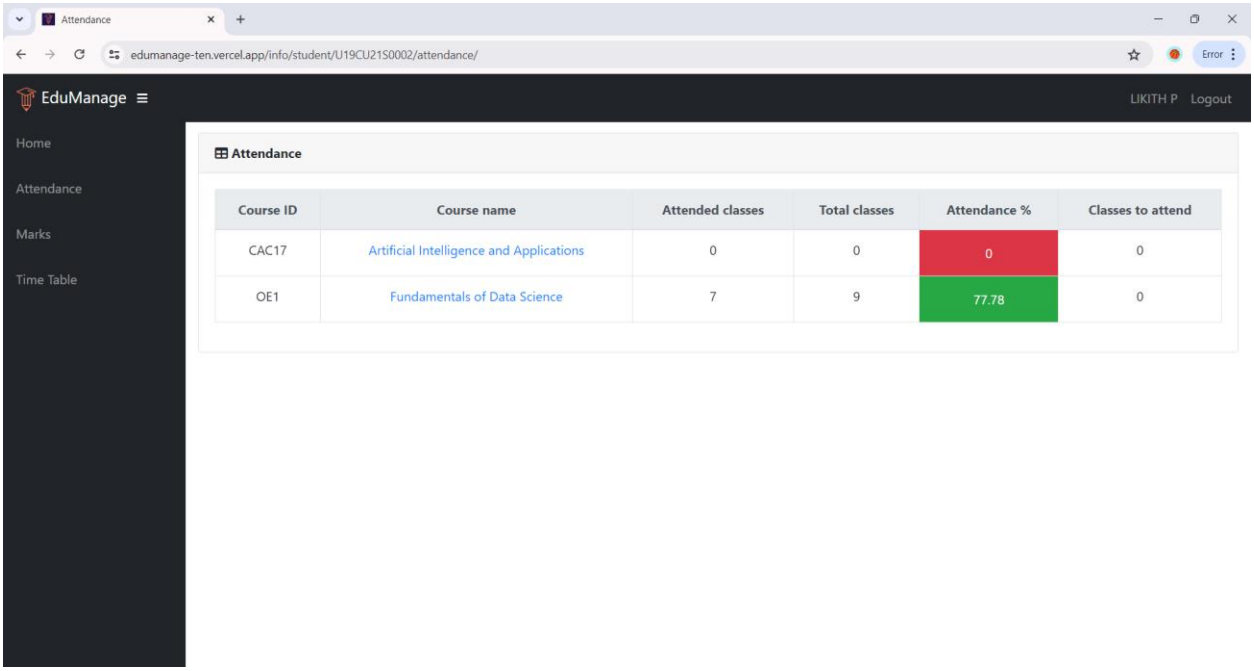
Reports Page



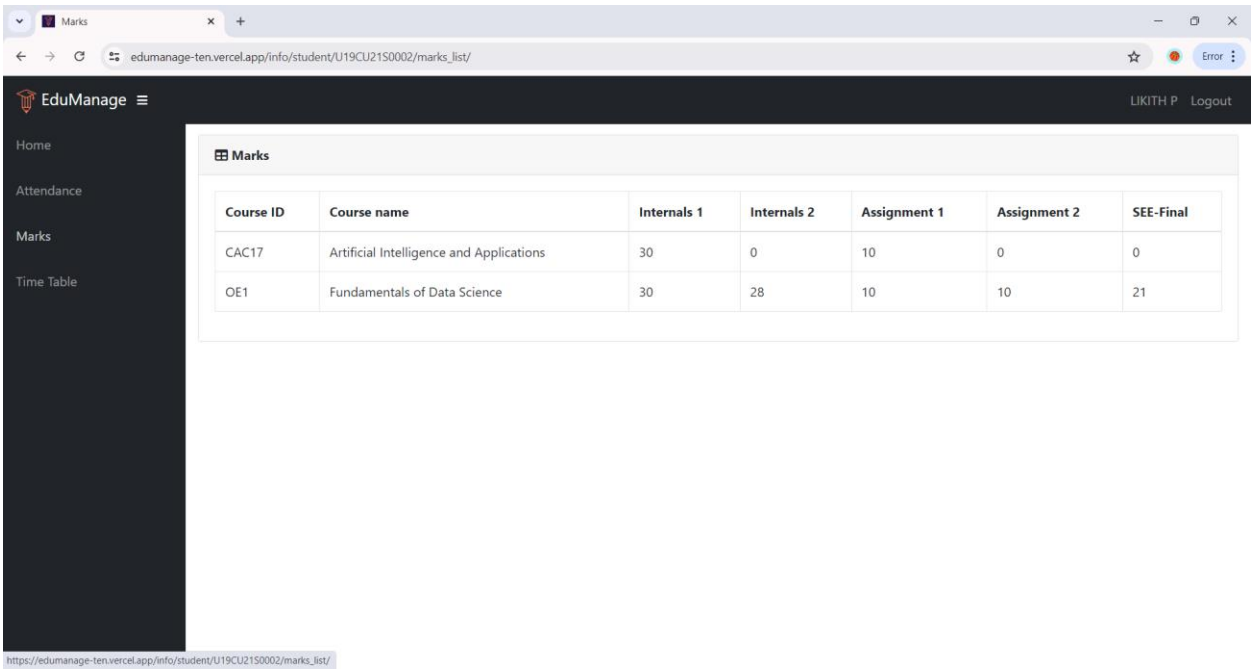
Students Dashboard



Students Attendance Page

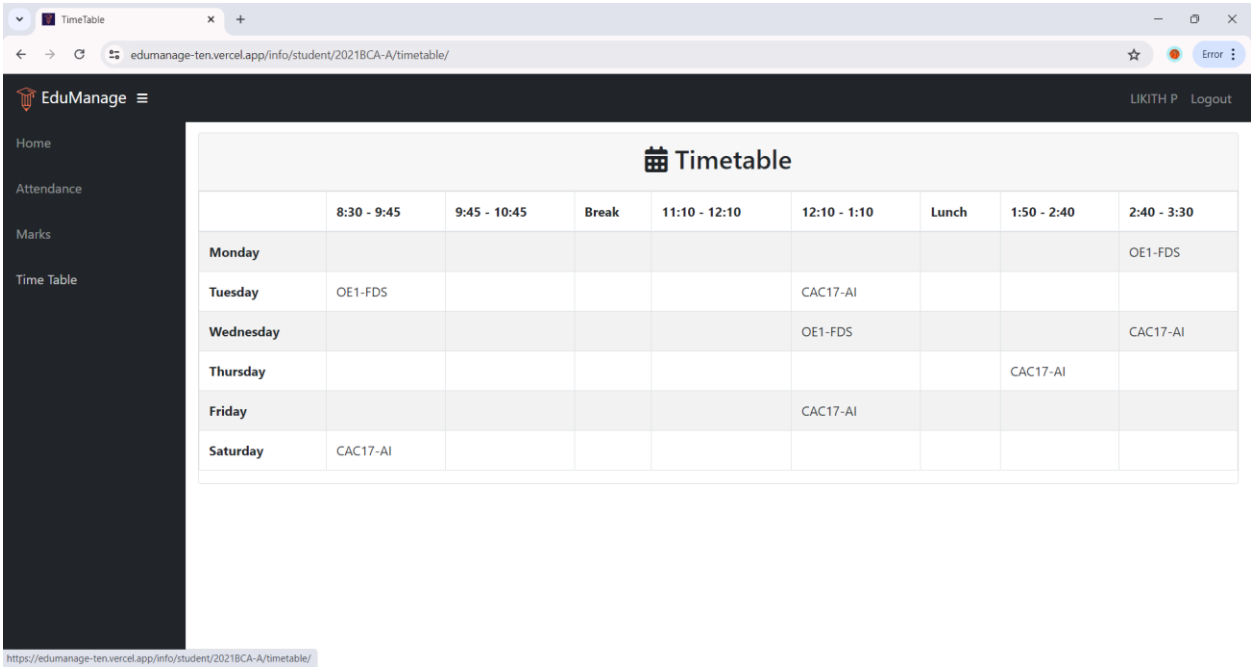


Students Marks Page



Course ID	Course name	Internals 1	Internals 2	Assignment 1	Assignment 2	SEE-Final
CAC17	Artificial Intelligence and Applications	30	0	10	0	0
OE1	Fundamentals of Data Science	30	28	10	10	21

Students Timetable Page



	8:30 - 9:45	9:45 - 10:45	Break	11:10 - 12:10	12:10 - 1:10	Lunch	1:50 - 2:40	2:40 - 3:30
Monday								OE1-FDS
Tuesday	OE1-FDS				CAC17-AI			
Wednesday					OE1-FDS			CAC17-AI
Thursday							CAC17-AI	
Friday					CAC17-AI			
Saturday	CAC17-AI							

12. CONCLUSION

The project “EduManage” system developed using Django framework represents a robust and scalable solution for efficiently managing academic records. Throughout this project, we have successfully implemented features that cater to the diverse needs of students, faculty, and administrative staff.

The system offers a centralized repository for student information, including personal details, academic records, attendance, and schedules. This centralization ensures easy access, retrieval, and management of data. With a focus on usability, the system provides intuitive interfaces for different user roles, ensuring that students, faculty, and administrators can perform their tasks with minimal training. The implementation of role-based access control ensures that sensitive data is protected and that users have access only to the information relevant to their roles.

The system simplifies various administrative processes, such as attendance tracking, timetable generation, and grade management, reducing manual workload and minimizing errors. The system is designed to be scalable and flexible, allowing for future enhancements and integrations as institutional needs evolve.

In conclusion, “EduManage” system stands as a testament to the effective application of modern web technologies in educational administration. It not only improves operational efficiency but also enhances the overall experience for students and staff.

13. FUTURE ENHANCEMENTS

To further improve the functionality and usability of the “EduManage” system, several future enhancements can be considered. These enhancements aim to provide a more comprehensive, user-friendly, and efficient system that meets the evolving needs of educational institutions.

For improving accessibility develop a mobile app for iOS and Android platforms to provide students and faculty with easy access to the system on-the-go, including notifications, schedules, and grades. Enhance the web interface to ensure it is fully responsive and optimized for mobile devices, providing a seamless user experience across all screen sizes. For improving performance analytics implement advanced analytics to provide deeper insights into student performance, enabling early identification of at-risk students and tailored intervention strategies. Develop attendance trend analysis to help identify patterns and provide actionable insights to improve student engagement.

Future enhancements for the “EduManage” system include integrating AI-powered chatbots to provide instant support for common queries, thereby reducing administrative workload. Enhanced communication tools such as an internal messaging system and a robust notification system for email, SMS, or in-app alerts will facilitate better interaction and timely updates. Security measures will be boosted by implementing multi-factor authentication (MFA) and ensuring data encryption. Additionally, a continuous user feedback mechanism will be established to gather regular input on system performance and usability, driving ongoing improvements.

By incorporating these future enhancements, the system can evolve into an even more powerful and user-friendly tool, further supporting the academic and administrative needs of educational institutions.

14. BIBLIOGRAPHY

1. Software Engineering - R.S. Pressman
2. Elmasri and Navathe: Fundamentals of Database Systems, 7th Edition, Pearson Education, 2016.
3. Ian Sommerville: Software Engineering, 10th edition, Person Education Ltd, 2015
4. Roger S Pressman: Software Engineering- A Practitioners approach,8th edition, McGraw-Hill Publication, 2015.

WEB REFERENCE

5. https://en.wikipedia.org/wiki/Requirements-engineering
6. https://web.cs.dal.ca/ hawkey/3130/srs-template-ieee.doc
7. http://www.ntu.edu.sg/home/cfcavallaro/Reports/Report%20writing.htmTop
8. https://en.wikipedia.org/wiki/Class diagram
9. https://www.djangoproject.com/
10. https://getbootstrap.com/
11. https://www.tutorialspoint.com/
12. https://creately.com/
13. https://www.overleaf.com/project