PROJECT REPORT ON

ACADEMIC MONITORING SYSTEM

SUBMITTED TO THE

SAVITRIBAI PHULE PUNE UNIVERSITY

BACHELOR OF ENGINEERING

SUBMITTED BY:-

BABITA KUMARI (72025574M)

SHIVKANYA BYALE (72025911J)

VAIBHAV MORE (72025781G)

Guide

DR. P. P. MANE

Department of Electronics and Telecommunication Engineering

Modern Education Society's College of Engineering, Pune

[2022-2023]

SAVITRIBAI PHULE PUNE UNIVERSITY

Modern Education Society's College of Engineering
(WADIA COLLEGE CAMPUS), Pune-411001

Department of Electronics and Telecommunication Engineering

This is to certify that the final year project seminar entitled **"Academic Monitoring System"** has been

Submitted by:-

1. Babita Kumari (72025574M)
2. Shivkanya Byale (72025911J)
3. Vaibhav More (72025781G)

As a partial fulfilment for the Bachelor of Engineering degree in Electronics & Telecommunication of Savitribai Phule Pune University during the academic year 2022-2023.

Dr. P. P. Mane	**Dr. R. S. Kadam**	**Dr. P. P. Mane**
Project Guide	Project Co-ordinator	Head of Department
Professor		Professor

Place: Pune

Date: 3rd June 2023

ACKNOWLEDGEMENTS

Ability and ambition are not enough for success. Many able people fail to achieve anything worthwhile because he or she has not been properly guided and directed. The project on "ACADEMIC MONITORING SYSTEM" is an outcome of guidance, moral support & devotion bestowed on us throughout our work. First and foremost, I offer our sincere thanks to Dr. R. S. KADAM for their guidance and constant supervision as well as for providing necessary information during seminar preparation as a Project Coordinator. We express our gratitude to Dr. P.P.Mane, head of the E&TC department for their kind cooperation.

Finally, I would like to express my gratitude towards my parents & and all teaching and non-teaching staff members of the E&Tc department for their kind cooperation and encouragement which help us in the completion of this project.

Thanking you,
Babita Kumari
Shivkanya Byale
Vaibhav More

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AMS	Academic Monitoring System
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets

CHAPTER 1

ABSTRACT

Every organisation should be concerned with academic monitoring since it can determine whether or not a particular organisation, such as a school or a company in the public or private sector, will be successful in the future. In order to maximise employees' and students' performance, organisations must maintain track of those who work for them.

It has become challenging to control student attendance during lecture sessions. Because manual computation is time-consuming and prone to error, it becomes difficult to calculate the attendance percentage. To track students' participation in the class, an effective web-based application for attendance management is created. This application uses technology to electronically record attendance, which is then stored in a database system.

Our primary goal is to create a distinctive student management system that will enhance both students' and administrative authorities' experiences with data management in departments. The system was created using Python, MongoDB, the Django framework, and Java script for the front end. Customers will be able to check in from any location with an internet connection. Following that, kids will be able to perform a variety of jobs that are intended for them. Software called the Student Management System is useful for both managers and students. All tasks are completed manually under the current system. It is expensive and takes a lot of time. The many student-related activities are handled via our student management system.

CHAPTER 2

INTRODUCTION

The academic monitoring system is a web-based tool created for institutes to assess and monitor teacher and student attendance. Designing and implementing a system and user interface will offer a complete alternative to the existing demand for paper records. Through a safe, online interface integrated into the college website, the department has direct access to every facet of a student's academic progress. Only the information required for a user's tasks is displayed by the system, which uses user authentication. Additionally, each subsystem has authentication capabilities that let authorised people add or edit data in that subsystem. Before any actual record alteration occurs, all data is carefully examined and approved on the server. The system is expected to boost the effectiveness of the institute record management by reducing the amount of time needed for users to access and distribute student records. It includes a sophisticated logging system to track all users' access and ensure compliance with data access requirements.

In the quick-paced digital age, educational institutions have several difficulties in effectively managing academic data, monitoring student achievement, and fostering effective communication between professors, students, and parents. An effective response to these problems is the Academic Monitoring System, which is built with Django. The goal of this project is to develop a user-friendly web-based platform that will help educational institutions improve student performance monitoring, expedite academic procedures, and promote open communication within the academic community. Django is a high-level Python web framework that is renowned for its scalability, security, and flexibility. The Academic Monitoring System makes use of its power. The goal of this project is to provide a comprehensive and adaptable system suited to the particular requirements of educational institutions by utilising the built-in features and expandable nature of Django.

.

Software called the "Attendance Monitoring System" was created to track students' everyday attendance at the college. Here, the staff members in charge of the topics are

in charge of recording the students' attendance. Each member of the team will receive a unique login and password based on the topic matter they manage. Here, a precise report is provided based on the student's attendance. This approach will also aid in determining a student's eligibility for attendance. Weekly and monthly attendance reports for the students are produced. The Academic Monitoring System is a Django and sqlite3-based web application that offers functionality like student database registration, attendance tracking, results tracking, and associated data monitoring.

Django-based attendance management system with source code in Python Django, HTML, CSS, JavaScript, and Bootstrap were used to construct the Django Attendance System. While the students can check their attendance, grades, and the class schedule, the teacher can manage the students' attendance, grades, and reports.

An application that lets you monitor your students attendance is called an attendance management system. designed to be used for daily student attendance in institutions, clubs, and schools Obtaining attendance data is made simpler by this.details on a certain student in a particular class. The information will be sorted by a teacher for a certain class by the operators when they are given the information. A student's attendance eligibility requirements can also be determined with the help of this approach. A system known as an academic monitoring system collects daily data on student attendance and uses it to track, monitor, and compile it. It helps the faculty quickly and accurately record and preserve student attendance reports, attendance histories, absentee records, and other activities.

An attendance management system's main objective is to automate the conventional way of taking attendance. This program's automatic report generation at the end or in the middle of the session is another consideration in its design.

.

CHAPTER 3

OBJECTIVES

- A web-based tool called "THE ACADEMIC MONITORING SYSTEM" was created to keep track of and maintain institute attendance.
- The technology will also assist in assessing a student's attendance.
- The Student and the Staff can send feedback about subject to admin and the admin(Hod) can reply.
- As per requirements an accurate report containing can be generated.
- Three key modules that can access the main dashboard are the admin/HOD dashboard, the staff dashboard, and the student dashboard.
- By keeping the records secure in the system, this technique does away with the necessity for paper-based records.
- This technique will be useful in assessing pupils' attendance.

CHAPTER 4

LITERATURE SURVEY

In addition to giving teachers and students easy access, we discovered that we could significantly reduce the time and effort required to precisely track and manage student attendance. Our system, which enables schools to grant teachers a set of permissions to handle student attendance data, requires a secure architecture and role-based access. A flexible and easily scalable system that enables teachers to access student attendance data and record absences.

For the purpose of tracking daily professor and student attendance in university departments, a web-based application was created for the system. Accessing a specific student's attendance in a certain class is made easier thanks to this. A student's eligibility for attendance will be assessed using this system, which will also assist in report generation. These days, attendance evaluation is done manually, which requires a lot of time and effort and sometimes results in errors. This method not only increases work efficiency and tracks students' academic progress, but it also helps to conserve both human and material resources.

An academic monitoring system is a piece of software that aids educational institutions in monitoring student progress, spotting possible issues, and delivering prompt interventions to improve outcomes. The system has the ability to gather information from several sources, including behaviour reports, grades, and test results. Additionally, it can offer analytics and visualisations to assist instructors and administrators in making judgements about students' performance.

An academic monitoring system can assist schools in identifying students who are at risk of falling behind or dropping out, which is one of its key advantages. Teachers can intervene early and offer tailored support to help students stay on track by tracking attendance and grades in real-time. Additionally, the system can assist schools in seeing patterns and trends in students' performance over time, which can guide the creation of

curricula and instructional methods.

An additional advantage of an academic monitoring system is that it can assist schools in enhancing communication with parents and other stakeholders. Schools may keep parents informed and involved in the learning process by regularly updating them on student progress. The system can make it easier for teachers, administrators, and support personnel to communicate with one another, which can enhance departmental coordination and collaboration.

Today's market is home to a number of different sorts of academic monitoring systems. Some are made for particular grade levels or subject areas, while others are more general-purpose tools that may be tailored to fit any school's or district's needs. Blackboard Learn, Infinite Campus, and PowerSchool are a few of the well-liked academic monitoring programmes.

Real-time access to student data, such as attendance records, grades, assignments, and test results, is made possible via the popular academic monitoring system PowerSchool. Along with analytics and reporting capabilities for administrators, the system also has options for instructors and parents to communicate.

Another well-known academic tracking application is Infinite Campus, which provides a full range of tools for monitoring student progress. The system has modules for tracking attendance, providing grades, scheduling, managing discipline, and more. Additionally, interaction with third-party programmes like Microsoft Office and Google Classroom is offered.

An academic monitoring capability is also included in the learning management system, Blackboard Learn. The system offers resources for monitoring student progress, controlling assignments, and interacting with both students and parents. Additionally, it has reporting and analytics tools to assist schools in making data-driven choices on curriculum and instruction.[12]

In conclusion, a district or school trying to enhance student outcomes might benefit

greatly from implementing an academic monitoring system. These systems can assist schools in identifying at-risk students early, offering targeted interventions, and improving overall performance by facilitating real-time access to student data, simplifying communication between stakeholders, and promoting data-driven decision-making.

comparisons between respondents' use of the manual approach and the online system for tracking and reporting attendance.

	Factors	Strongly Manual System	Agreed Online System
1	Attendance system needs	8.4%	32.9%
2	Attendance system is efficient & effective	5.8%	33.8%
3	Attendance sheets are easy to allocate	12.5%	27.8%
4	Record keeping is more organized	5.1%	38.9%
5	The potential of losing attendance sheets is low	4.5%	40.2%
6	Easy to track students' absences	10.4%	36.2%
7	The format of the report for students' absences is consistent	6.2%	34.9%
8	Delivery of reports for students' absences are easy to produce	5.8%	34.4%
9	The process to report students' absences is easy	6.7%	36.9%
10	Students are not difficult to contact to attend the discipline trials via emails	8.2%	27.3%
11	The process to inform student's status to parents are effective	12%	26.2%
12	Cost to develop, manage & maintain the attendance system is low	13.8%	34.9%
13	The attendance system process is fast	7.1%	40.2%
	Average	8.2%	34.2%

Figure 4.1: 4.1 Comparison of manual and online system

This chapter will go over how to conduct a literature review to acquire data for the project. In addition, this chapter will go through the Academic Monitoring Sys-

tem project approach that will be used to construct the project. This chapter will also go through the software, hardware, and other technologies that will be used to create the system. Additionally, it includes the schedule, which will serve as a guide for the project. The existing student management system that is currently in use will be the main topic of the literature review. One of the most crucial stages development systems should take from the beginning to the finish of a project is the project methodology.

Digitalizing the Old Approach

It takes a lot of time for students and professors to conduct departmental sessions with traditional student attendance because of all the roll-calling concerns. The process takes a lot of time from both teachers and pupils. Mendonca et al.[1] decreased the duration of the entire attendance verification process by creating an online method. Teachers were required to call each student's name in class in place of the customary practise and note the attendance when the student responded. It provides a simpler and more efficient method of keeping track of attendance. The approach they suggest will eliminate the need for instructors to record student attendance on a paper sheet. They can create attendance records by pulling the essential data from the database, eliminating the need for paper in the process.

Another study created and implemented an attendance management system using mobile devices. Using VB.NET and SQL Server, an Android-based programme for managing attendance on the go was created. This project enables the upkeep of student attendance, the computation of attendance grades, and the production of reports. The system consists of five parts: administration, registration, students, SMS, and an Android component. Students can send messages to the system notifying professors of their absence using the Android component. Parents can receive SMS alerts on their children's behaviour.[13]

Fingerprint Recognition Based

The majority of studies have shown that fingerprint or hand gesture recognition is an extremely effective way for an attendance management system. Automated fingerprint

recognition is a process that compares a set of known and unknown fingerprints in a database to a set of one or more unknown fingerprints. According to Mohamed and Raghu[14], a specific finger assumption tool that is a part of a unique finger impression attendance framework is used. Students can use their fingertips to touch the sensor on the device to check their essence. But this approach doesn't work because fingerprint scanners can't always identify something the first time.

An attendance system using smartphone GPS and fingerprint technology was described by Soewito et al.[15] Due to the usage of fingerprint recognition, the process is time-consuming.

GPS-based Attendance System

A person's location and direction may be found at any time, wherever on Earth, thanks to the Global Positioning System, or GPS. People still require objects in the sky to know where they are and how to go to other places, but satellites now make use of them. The time and attendance monitoring system developed by Kumar and Kumar[3] in their work was implemented on an Android mobile app. The adoption of cellphones lessens the requirement for extra biometric scanning apparatus. A particular place, which can be located via GPS, is one of the organisation's components. Each student's location can be ascertained using the GPS on their mobile devices, and these locations are essential for time and attendance tracking.

Barcode / QR code Based

With each item, there is a barcode that graphically represents the information that machines can read. Like a barcode, a quick response code, or QR code, is used for speedy communication. In spite of this, due to its two-dimensionality, it can store information in both perpendicular directions. Therefore, compared to a barcode, a QR code may be able to store many times more data. Noor et al.[16] introduced their discussion of an automated method for tracking student attendance. Everyone in this system has a distinct ID that is issued with a barcode that the smartphone app may scan. The mobile app may scan the unique ID that each user in this system has, which has

a barcode. This method had the disadvantage that one student might manipulate it by using the IDs of the other participants in the arrangement.

A different strategy hinges on capturing attendance and updating data in one place. The suggested approach, which was developed using QR code technology and is based on research by Sutar et al.[17], is a smart attendance system that would accelerate the attendance process by making and scanning QR codes. The system is based on QR technology and operates as an application for mobile devices.

Additionally, Sunaryono et al.[8] advise implementing "an Android-based course attendance system using face recognition" to guarantee student participation in the course. The course details are delivered from the front of the class using a QR code that contains the information. Just a selfie of their face and the display of a QR code are needed from the student's phone. After that, the image will be sent to the server for attendance management.

Face Recognition Based

Face detection is the concept of identifying human faces in movies or images that are used as references. A face recognition system is a form of technology that can examine facial pictures from a video or photograph and compare them to a database of both known and unidentified faces. The face recognition-based attendance management system was created by Smitha to create a structured classroom attendance system utilising face recognition techniques[7]. The system can track participation by using facial ID. It locates faces using a camera, then identifies them. Facial detection and recognition are the two components of the system.The system will use the Local Binary Pattern Histogram (LBPH) to recognise students' faces in the live-streamed video from the class, and if the recognised face is located in the database, the system will mark the students' attendance.

Varadharajan et al.[18] also covered face recognition technologies in their work. In order to take images, they positioned a camera inside the classroom. Following the discovery and identification of faces in the database, attendance is recorded as a present. If a student's absence is recorded, parents are notified of their child's whereabouts.

The study by Chandramouli et al.[4], in which they used NVIDIA's Jetson Nano, is one of several that aims to modernise time management guidelines as well as how attendance is controlled in a particular way. The device is placed in the classroom, where it displays the pupils' names and pictures. The photographs were obtained through OpenCV. The developer kit for the NVIDIA Jetson Nano would be the CPU board. Once the extraction has been processed, faces are identified using a Haar classifier. The LBPH algorithm was then used to help them identify data from the appropriate class teacher, which is generated and updated hourly in an Excel spreadsheet.

Prangchumpol notes in his study, "Face Recognition for Attendance Management System Using Multiple Sensors," that he is still unable to confirm or correct the data when an error occurs in the data or accurately identify students' faces. In order to make the face recognition-based attendant system more effective, he also wants to make the system's concepts easy for pupils to grasp. Using the Android Face Recognition with Deep Learning technique, this type of validation seeks to learn how to recognise faces. Using cloud storage, the database and web server are connected.[5]

Three important problems were addressed by the methods Alburaiki and colleagues created: Using the cameras in mobile phones, we'll start by automatically identifying and analysing faces. The second is a facial recognition API built on machine learning. The Maps API is the last. The result reveals that face recognition has achieved excellent accuracy in recognising students' faces even under unfavourable circumstances. By recording a student's attendance after recognising their face and location, the system provided real-world samples of responses. The lecturer also had the option of viewing a report of recorded attendance.[6]

Android-Based Authorized ID and Password

For touch-based mobile devices, the Android OS was primarily created. Its foundations consist of various open-source software programmes and a significantly modified Linux kernel. Every time you use an Android-based smartphone and access an application or website, you may be asked to sign up or sign in. Normally, you can request to have a login and password created. The fact that this process is now so well-liked may cause some users to register their accounts without paying attention to their passwords

because it has nearly become a routine step. Sadly, there is a lot of risk if a user selects subpar credentials.

A sophisticated Android-based attendance system was created and implemented by Hameed. The system generates attendance data automatically and provides a more convenient, affordable, and quick way to monitor online student attendance. The three features of the attendance system are the admin account, which can log in and modify the database; the instructor account, which can designate students as present; and the reporter account, which can check attendance records and report on all tasks.

According to the study by Islam et al.[21], the course instructor will be able to easily collect attendance using phones running the Android OS, which has been built to save attendance both on the device and servers, as well as verify statistics and print a paper version. Their system uses the recorded data to keep parents informed about their child's enrollment at the institution by recording attendance, noting intruders' admittance, calculating attendance percentages, and sending emails and text messages to parents. The suggested technology would make it possible to access the internet at any time and from any location, which might be quite helpful for instructors trying to keep track of their students' attendance.

Kumbhar et al.[19] created an attendance management system to solve problems with students using Android devices to attend class. For system access, both professors and students must install APKs on their smartphones. A special ID and password were also given to them. The application must be completed by students along with their parents' information. Once the programme has been launched by the lecturer and is ready to be used during attendance checking, the student can easily register their attendance with a single click. There is attendance data available for professors on a weekly and monthly basis. Every month, SMS is used to notify parents of their child's attendance.

Android-Based RFID

Many Android smartphones use NFC (near-field communication) technology for digital payments, and some academics consider RFID to be a simpler variation of that technology. Souza et al.[20] investigate various frameworks for board participation that use various innovations. In light of this discussion, the board is advised to utilise

a different method of participation designed expressly for institutions of a lower tier. RFID and mobile application components are both included in the suggested model. For recording student interaction in the database at the back end, the RFID component is advised. Their families' attendance details will be provided in the application phase. The application component is used as a fallback method to track attendance when there is no electricity or insufficient capacity for the RFID component to deliver data.

Numerous investigations have been conducted by researchers to develop biometric recognition based on smart attendance systems.

They focused on creating technical solutions to problems with identifying, recognising, recording, monitoring, placing, and tracking pupils or staff rather than traditional systems that impose duplication of labour and the need for more processes and daily resources. In past studies that recommended using attendance management systems, academic institutions were employed as test subjects. Therefore, we shall find that the facial recognition-based method, whether with portable or mobile devices, has grown much more popular than any other available methodology by highlighting recent research that was conducted on this subject. And the most effective and widely used technology for researchers to work on is now this one.

Author & Year	Technology	Main Findings
Sutar et al. (2022)	Android, and QR code	Create a QR code for each lesson and suggest a system that involves scanning it for evaluation.
Kumar and Kumar (2021)	Android, GPS, and Server	Students' locations are determined using GPS on their phones. Specifically, this is a key for recording
Sunaryono et al. (2021)	Android,Face Recognition, and QR code	Using a smartphone, each student will request to have their face and the class's QR code captured.
Alburaiki et al. (2021)	Android, GPS, and Face Recognition	Lecturers take attendance, and students sign themselves in by scanning their faces and whereabouts.
Smitha (2020)	loT, Face Recognition, and Email Server	Faces will be recognised from a live classroom video and used to track attendance at each lesson.
Souza et al. (2019)	Android, RFID, and Email Server	RFID is used at the back end to track student involvement. The application's goal is to provide families with information about attendance.
Chandramouli et al. (2021)	loT, NVIDIA Jetson Nano and Face Recognition	Faces are recognised by the LBPH algorithm after being classified by the Haar classifier, and attendance is marked by comparing the histogram to the
Susanto et al. (2021)	Android, and Face Recognition	Making an Android application by utilising OpenCV for speed detection and facial detection will include the
Prangchumpol (2019)	Android, IoT, Cloud and Face Recognition	Use Android face recognition with deep learning to identify the face.
Hameed (2019)	Android, Authorized Username/Password and Web Server	The system's three roles are database administrator, teacher, and reporter, each of which has the authority to make changes to the database and label students as present or absent.

Figure 4.2: Summary of recent findings and developments of Academic Monitoring System.

CHAPTER 5

DESIGN AND METHODOLOGY

The Academic Monitoring System is a web-based tool created to simplify and automate the process of monitoring and managing academic information within an educational institution. It seeks to offer a centralised platform for efficiently managing academic data access for administrators, professors, and students.

Requirements Gathering

Gathering requirements is the first step in building an academic monitoring system. This entails identifying the important stakeholders, comprehending their requirements, and outlining the fundamental capabilities and features of the system. User identification, student enrollment, course management, grade tracking, attendance monitoring, and report generation are examples of typical requirements.

Functional requirements documentation Make a list of the system's functional needs based on stakeholder feedback and specified objectives. This might have characteristics like:

1) Student and instructor profiles: Record pertinent data on students and teachers, such as contact information, academic history, and personal particulars.

2) Real-time monitoring of student attendance is enabled, with manual or automatic entry options.

3) Provide a platform for tracking, computing, and managing student grades. This platform should support a variety of grading schemes and weighting schemes.

4) Management of assignments and examinations: Make it easier to create, distribute, submit, and grade assignments and exams.

5) Consider non-functional criteria, such as scalability, security, usability, performance, and accessibility, in addition to functional requirements. These factors are necessary to provide a dependable and user-friendly system.

6) Prioritise the requirements you've gathered based on their significance and viability.

Vs code instalation

Follow these steps to install Visual Studio Code on your computer:

1) Visit https://code.visualstudio.com/download to see the official Visual Studio Code page.

2) Choose the Windows, macOS, or Linux download package that is relevant for your operating system.

3) Run the installer after downloading it.

4) To finish the installation procedure, adhere to the on-screen directions.

5) Following successful installation, you can access Visual Studio Code by clicking on the desktop icon or by looking for it in the Start menu. You may then start changing your code by creating new files or openingexistingones.

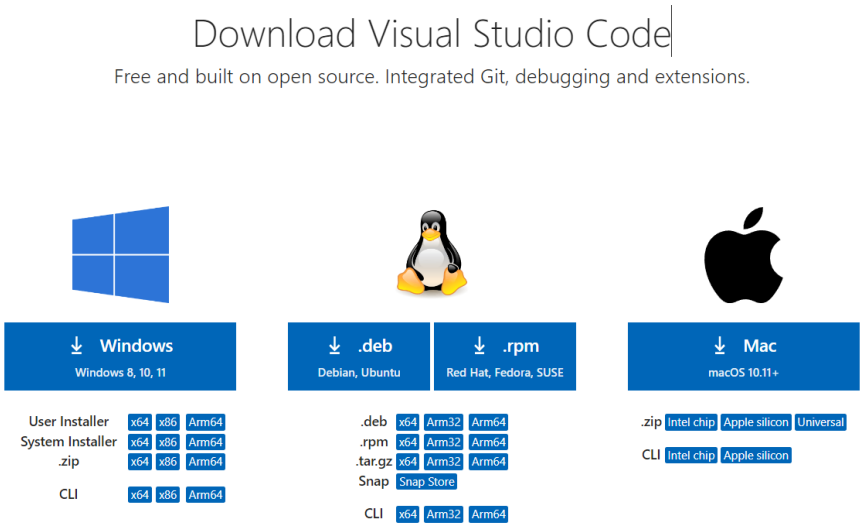


Figure 5.1: VS code installation

Microsoft created the free and open-source Visual Studio Code, sometimes known as VS Code, for Windows, Linux, and macOS. Since its 2015 release, it has grown to be one of the most widely used code editors among developers.

Numerous programming languages are supported by VS Code, including HTML, CSS, JavaScript, TypeScript, Python, and Java. Debugging, Git integration, syntax highlighting, code completion, and extensions are just a few of the capabilities it offers. These can all be obtained via the VS Code Marketplace.

The IntelliSense functionality of VS Code is one of its standout features. As you input your code, IntelliSense suggests variables, functions, and methods. As a result, productivity is increased and errors are decreased.
In addition to its core features, VS Code also offers a number of extensions that can enhance its functionality. For example, the Live Server extension allows you to preview your HTML code in a web browser

VS Code features

VS Code's primary features include things like:

Based on the context of your code, VS Code offers intelligent code completion and recommendations. VS Code has a built-in debugger that you can use to step through your code and find problems.
Git integration: VS Code comes with integrated Git support, enabling you to manage your code repositories and work with others. Extension Market: VS Code offers a sizable extension market where you can find tens of thousands of extensions to give your editor new features.
Customization: VS Code gives you the option to personalise your editor with themes, keybindings, and other options.

Necessary VS code Extentions required

To install Python Django and Code Runner on VS Code, follow these steps:

1) Install Python: If you haven't already, download and install Python from the official website (https://www.python.org/downloads/). Make sure to select the option to add Python to your system PATH during the installation process.

2) Install Django: Open a command prompt or terminal window and run the following command to install Django:

pip install django 3) Code Runner: Open VS Code and go to the Extensions tab on the left-hand side. Search for "Code Runner" and install the extension by Microsoft.

4) Configure Code Runner: Open VS Code settings by clicking on the gear icon in

the bottom left corner and selecting "Settings". Search for "Code Runner" in the search bar and click on "Edit in settings.json". Add the following lines to the file:

"code-runner.executorMap": "*.py": "python" , "code-runner.saveFileBeforeRun": true This will configure Code Runner to use Python as the default executor for.py files and save the file before running it.

5) Create a new Django project: Open a command prompt or terminal window and navigate to the directory where you want to create your new Django project. Then, run the following command: django-admin startproject projectname 6) Replace "projectname" with the name of your project 7) Create a new Django app: Navigate to the newly created project directory and run the following command: python manage.py startapp appname 8) Replace "appname" with the name of your app.

9) Run the development server: Navigate to the project directory and run the following command: python manage.py runserver This will start the development server and you can access your Django app by opening a web browser and navigating to http://localhost:8000/.

That's it! You now have a new Django project and app set up on VS Code with Code Runner. You can now start editing your code and see the changes in real-time in your web browser.

Getting started with Visual Studio

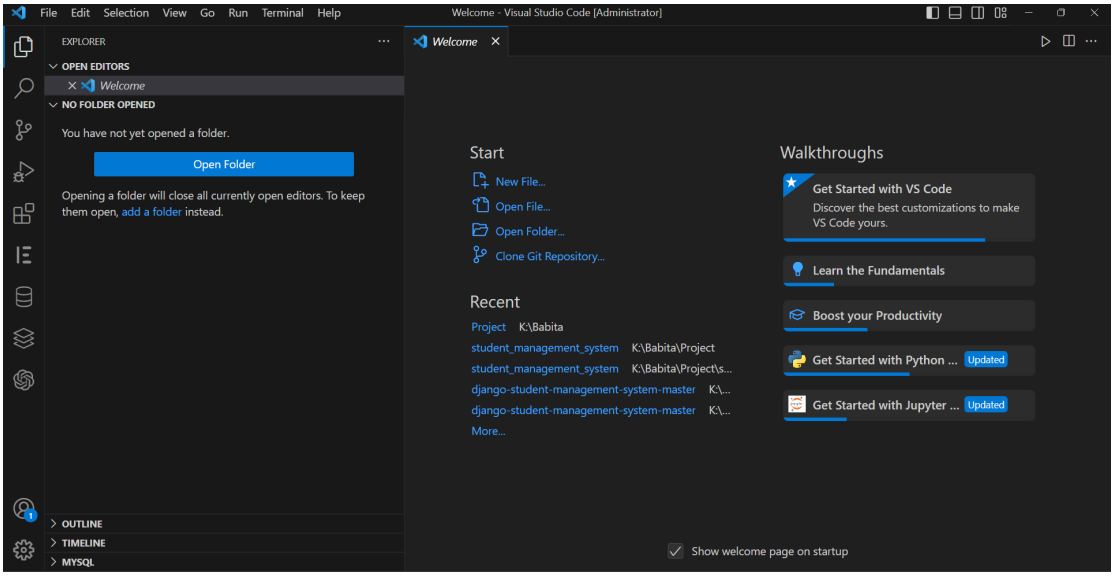


Figure 5.2: VS code installation

Database Design

The design of the database structure comes after the requirements have been obtained. Django uses an object-relational mapping (ORM) mechanism that enables programmers to specify the database structure using Python classes. The relevant entities, such as students, courses, grades, and attendance records, are mapped to the appropriate database tables, and the links between these entities are constructed.There are various important factors to take into account while designing a database for a Django academic monitoring system. First and foremost, it's crucial to choose a Database Management System (DBMS) that fits the system's needs and is compatible with Django. Making a data model that outlines entities, properties, and relationships is the next stage. Tables are made to represent entities and their properties, and this forms the basis for constructing the database structure. Data integrity and consistency are guaranteed when relationships between tables are properly defined using foreign keys. In addition, indexing and normalisation strategies are used to enhance query performance and get rid of redundant data. By following these procedures, a well-structured and effective database design may be created, satisfying the data management requirements of the Academic Monitoring System.

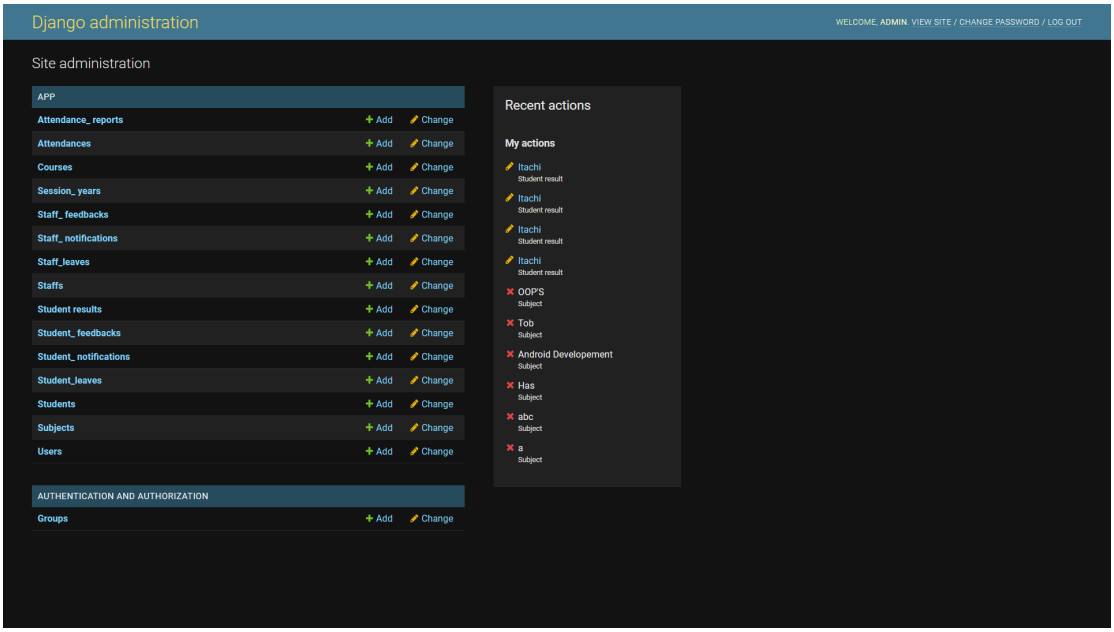


Figure 5.3: django administration

1. **Entity-Relationship (ER) Modeling**

Entity-Relationship (ER) models, which reflect the system's entities (such as students, professors, courses, etc.), are the first step in database design. The key data elements, their characteristics, and the connections between them can be found using the ER model. The groundwork for designing the database schema is laid out in this step.The method for storing data in Django is called a model. Django connects code to data using object-relational mapping (ORM). In order to communicate with data from several relational databases, including SQLite, PostgreSQL, and MySQL, the Django web framework includes a default object-relational mapping layer (ORM). Using this ORM API provided by Django, we are able to create, edit, delete, and query objects.

2. Database Schema Design

Entity-Relationship (ER) models, which reflect the system's entities (such as students, professors, courses, etc.), are the first step in database design. The key data elements, their characteristics, and the connections between them can be found using the ER model. The groundwork for designing the database schema is laid out in this step. One of the most crucial steps in creating a web application is designing the database. Maintaining appropriate relationships between the tables is essential if you're using a relational database.

A database table is represented by one model per Django application. Django models use the Relational Database Management System (RDBMS) by default. Therefore, you must create your Django models while maintaining appropriate interactions between them.

There are three different kinds of relationships in a relational database system:
One-to-One relationship
One-to-Many relationship (or Many-to-One)
Many-to-Many relationship

These three different kinds of relationships are supported by Django models.

A. One-to-One relationship

In this kind of connection, one record from one model is associated with precisely one record from another model. One-To-One field provides this relation's definition. Security motives are served by this kind of partnership.

B. One-to-Many relationship

One record of the first model is associated with several records of the second model in this kind of relationship, while only one record of the second model is related to one record of the first model. Foreign Key is the relationship's definition. Relationships of this kind are common in the workplace.

C. Many-to-Many relationship

Each record of the first model is associated with numerous records of the second model in this type of relationship, and each record of the second model is related to numerous records of the first model. The Many-To-Many field in this relation describes it.

3. Normalization

Redundancy in the database design is removed by the normalisation process, which guarantees data integrity. To prevent data duplication and reduce update anomalies, it entails separating the data into different tables. The database design is made more effective and adaptable by the use of normalisation techniques such as First Normal Form, Second Normal Form, Third Normal Form, etc.

4. Primary Keys and Foreign Keys

Each record in a table has a primary key that uniquely identifies it. If they are not explicitly defined, Django creates primary keys for each model automatically. By referring to the primary key of another table, foreign keys create associations between tables. They aid in preserving data integrity and upholding referential integrity restrictions. To define and maintain primary keys and foreign keys in the models, Django offers simple methods.

5. Indexing

For the purpose of improving query performance, indexing is a crucial component of database design. The database engine can quickly get the necessary data by building indexes on columns that receive a lot of queries. The db index argument in the field specification of the model in Django can be used to define indexes on particular fields. The system's overall performance can be greatly enhanced by proper indexing, especially when working with huge datasets.

6. Data Validation and Constraints

Data validation and constraints ensure that the data in the database complies with estab-

lished guidelines and standards. Django offers built-in field types with validation criteria, such as EmailField for email addresses, IntegerField for integers, and CharField for texts. In order to impose particular business rules and limitations on the data, custom validators can also be used.

7. Database Optimization

The performance and effectiveness of the system can be increased by using database optimisation techniques. This includes utilising caching techniques, enhancing database queries, and utilising capabilities unique to databases (such as stored procedures or database views). Writing effective and optimised database queries is made simpler by Django's ORM (Object-Relational Mapping) layer, which abstracts the underlying database.

8. Data Migration

Changes to the database schema might be necessary as the Academic Monitoring System develops. An integrated migration system provided by Django makes it possible to alter database schemas without losing any data. Changes to the database's data and structure are tracked and implemented using migrations. The database can be kept up to date with changing models and schema designs by producing and using migrations.

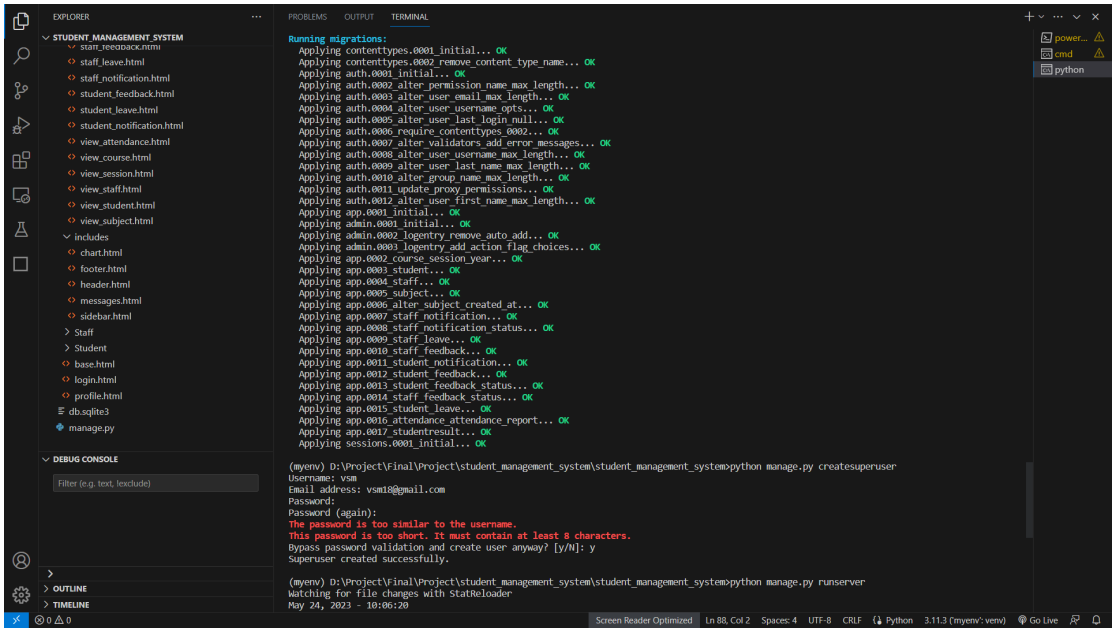


Figure 5.4: Migration

9. Backup and Recovery

The database must be regularly backed up in order to protect against corruption or loss of data. To ensure that the system's data can be restored in the event of any unantic-

ipated situations, it is crucial to adopt backup methods. Regular automated backups, offsite storage, and a recovery strategy can all be used in this way to lessen the effects of data loss.

ER diagram

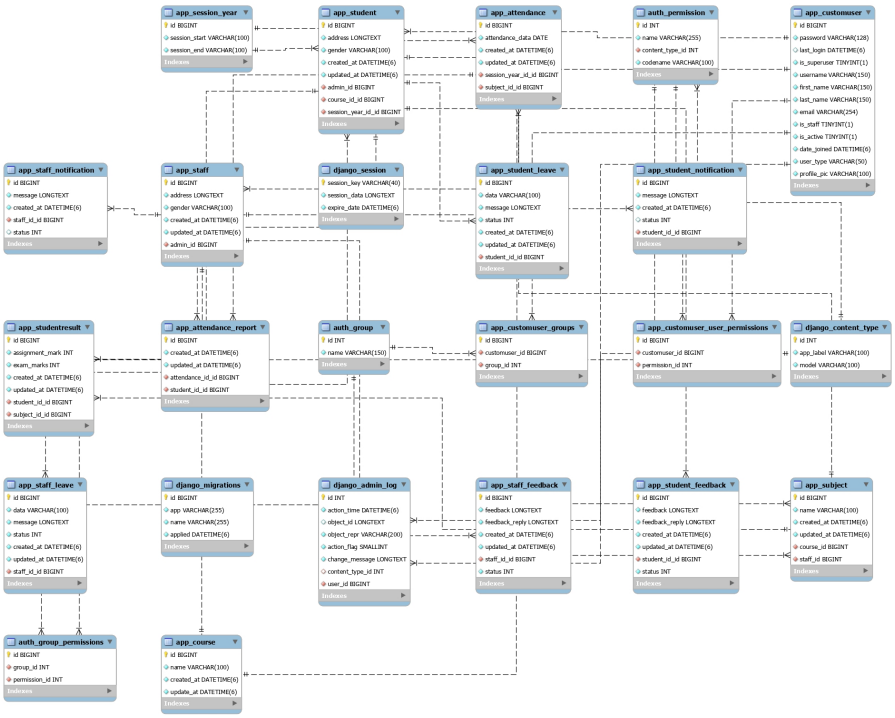


Figure 5.5: ER

based on the above ER diagram the tables included in the database as aur entities are as

1)app_session_year

2)app_staff

3)app_attendance_report

4)django_migrations

5)app_course

6)app_staff_notification

7)app_studentresult

8)app_staff_leave

9)auth_group_permissions

10)app_student

11)django_session

12)auth_group

13)django_admin_log

14)app_attendance

15)app_student_leave

16)app_customuser_groups

17)app_staff_feedback

18)app_customer_user_permissions

19)app_student_feedback

20)app_student_notifications

21)auth_permissions

22)app_customuser

23)django_content_type

24)app_subject

The Entities can be described for their funtionality as

- app_session_year: This entity represents the year of the academic session, such as the first year, second year, or third year.
- app_staff: This entity represents the staff members who teach the courses in the academic session.
- app_attendance_report: This entity represents the attendance reports submitted by the students.
- django_migrations: This entity represents the database migrations that have been applied to the database.
- app_course: This entity represents the courses offered by the institution.
- app_staff_notification: This entity represents the notifications sent to the staff members.
- app_student_result: This entity represents the results of the students' courses.
- app_staff_leave: This entity represents the leaves taken by the staff members.
- auth_group_permissions: This entity represents the permissions assigned to groups.
- app_student: This entity represents the students enrolled in the courses.
- django_session: This entity represents the user sessions in the Django project.
- auth_group: This entity represents the groups of users in the Django project.
- django_admin_log: This entity represents the admin logs in the Django project.
- app_attendance: This entity represents the attendance records of the students.
- app_student_leave: This entity represents the leaves taken by the students.
- app_customuser_groups: This entity represents the groups assigned to users.

Fields and relationships between the sevral entities

The academic monitoring system is a web application built using Django framework. It has several entities, each with its own set of fields and relationships. Here's a brief explanation of each entity:

app_session_year: This entity represents the year of the academic session, such as the first year, second year, or third year. It has fields for the session ID, session name, start date, end date, and whether the session is active or not. It also has fields for the creation and update timestamps. The entity has a many-to-many relationship with the app_course entity through a through table.

app_staff: This entity represents the staff members who teach the courses in the academic session. It has fields for the staff ID, first name, last name, email, phone number, address, and creation and update timestamps. The entity has a many-to-many relationship with the app_course entity through a through table, and also has a many-to-many relationship with the app_session_year entity through a through table.

app_attendance_report: This entity represents the attendance reports submitted by the students. It has fields for the attendance report ID, student ID, course ID, session year ID, date, status, and creation and update timestamps.

django_migrations: This entity represents the database migrations that have been applied to the database. It has fields for the migration ID, app name, migration name, and the timestamp when the migration was applied.

app_course: This entity represents the courses offered by the institution. It has fields for the course ID, course name, course code, and creation and update timestamps. The entity has a many-to-many relationship with the app_session_year entity through a through table, and also has a many-to-many relationship with the app_staff entity through a through table, and a many-to-many relationship with the app_student entity through a through table.

app_staff_notification: This entity represents the notifications sent to the staff members. It has fields for the notification ID, staff ID, notification title, notification message, and creation and update timestamps.

app_student_result: This entity represents the results of the students' courses. It has fields for the result ID, student ID, course ID, session year ID, term, academic year, grade, and creation and update timestamps.

app_staff_leave: This entity represents the leaves taken by the staff members. It has fields for the leave ID, staff ID, leave type, start date, end date, reason, status, and creation and update timestamps.

auth_group_permissions: This entity represents the permissions assigned to groups. It has fields for the permission ID, group ID, and permission object ID.

app_student: This entity represents the students enrolled in the courses. It has fields for the student ID, first name, last name, email, password, and creation and update timestamps. The entity has a many-to-many relationship with the app_course entity through a through table.

django_session: This entity represents the user sessions in the Django project. It has fields for the session key, session data, and the expiration date of the session.

auth_group: This entity represents the groups of users in the Django project. It has fields for the group ID and group name.

django_admin_log: This entity represents the admin logs in the Django project. It has fields for the log ID, user ID, content type ID, object ID, action flag, change message, and creation and update timestamps.

app_attendance: This entity represents the attendance records of the students. It has fields for the attendance ID, student ID, course ID, session year ID, date, status, and creation and update timestamps.

app_student_leave: This entity represents the leaves taken by the students. It has fields for the leave ID, student ID, leave type, start date, end date, reason, status, and creation and update timestamps.

app_customuser_groups: This entity represents the groups assigned to users. It has fields for the ID, user ID, and group ID.

app_staff_feedback: This entity represents the feedback submitted by the staff members. It has fields for the feedback ID, staff ID, feedback title, feedback message, and creation and update timestamps.

app_customer_user_permissions: This entity represents the permissions assigned to users. It has fields for the permission ID, user ID, and permission object ID.

Explaination of how the entities are connected to each other and the fields associated with them.

app_session_year:

Field: id (primary key)
Field: name (string)
Field: start_date (date)
Field: end_date (date)
Field: is_active (boolean)
Field: created_at (datetime)
Field: updated_at (datetime)
Relationship: Many-to-many relationship with app_course (through table)

app_staff:

Field: id (primary key)
Field: first_name (string)
Field: last_name (string)
Field: email (string)
Field: phone_number (string)
Field: address (string)
Field: created_at (datetime)
Field: updated_at (datetime)
Relationship: Many-to-many relationship with app_course (through table)
Relationship: Many-to-many relationship with app_session_year (through table)

app_attendance_report:

Field: id (primary key)
Field: student_id (integer, foreign key to app_student)
Field: course_id (integer, foreign key to app_course)

Field: session_year_id (integer, foreign key to app_session_year)

Field: date (date)

Field: status (string)

Field: created_at (datetime)

Field: updated_at (datetime)

django_migrations: Field: id (primary key)

Field: app (string)

Field: name (string)

Field: applied (datetime)

app_course:

Field: id (primary key)

Field: name (string)

Field: code (string)

Field: created_at (datetime)

Field: updated_at (datetime)

Relationship: Many-to-many relationship with app_session_year (through table)

Relationship: Many-to-many relationship with app_staff (through table)

Relationship: Many-to-many relationship with app_student (through table)

app_staff_notification:

Field: id (primary key)

Field: staff_id (integer, foreign key to app_staff)

Field: title (string)

Field: message (string)

Field: created_at (datetime)

Field: updated_at (datetime)

app_student_result:

Field: id (primary key)

Field: student_id (integer, foreign key to app_student)

Field: course_id (integer, foreign key to app_course)

Field: session_year_id (integer, foreign key to app_session_year) Field: term (string)

Field: academic_year (string)

Field: grade (string)

Field: created_at (datetime)

Field: updated_at (datetime)

app_staff_leave:

Field: id (primary key)

Field: staff_id (integer, foreign key to app_staff)

Field: leave_type (string)

Field: start_date (date)

Field: end_date (date)

Field: reason (string)

Field: status (string)

Field: created_at (datetime)

Field: updated_at (datetime)

auth_group_permissions:

Field: id (primary key)

Field: group_id (integer, foreign key to auth_group)

Field: permission_id (integer, foreign key to auth_permission)

app_student:

Field: id (primary key)

Field: first_name (string)

Field: last_name (string)

Field: email (string)

Field: password (string)

Field: created_at (datetime)

Field: updated_at (datetime)

Relationship: Many-to-many relationship with

app_course (through table)

django_session:

Field: session_key (primary key)

Field: session_data (string)

Field: expire_date (datetime)

auth_group:

Field: id (primary key)

Field: name (string)

django_admin

Django Project Setup

An additional Django project is made in order to start building the academic monitoring system. Specifically, this entails setting up the database connection, the project directory, and the project settings. Static files and templates are among the many configurations that are defined.

A web application's setup in a Django project requires a number of stages. Here is a quick rundown of the Django project setup procedure:

1. Install Django

First, ensure that Python is installed on your system. Then, using pip (the package installer for Python), you can install Django by running the command pip install Django in your command prompt or terminal. This will download and install the latest version of Django.

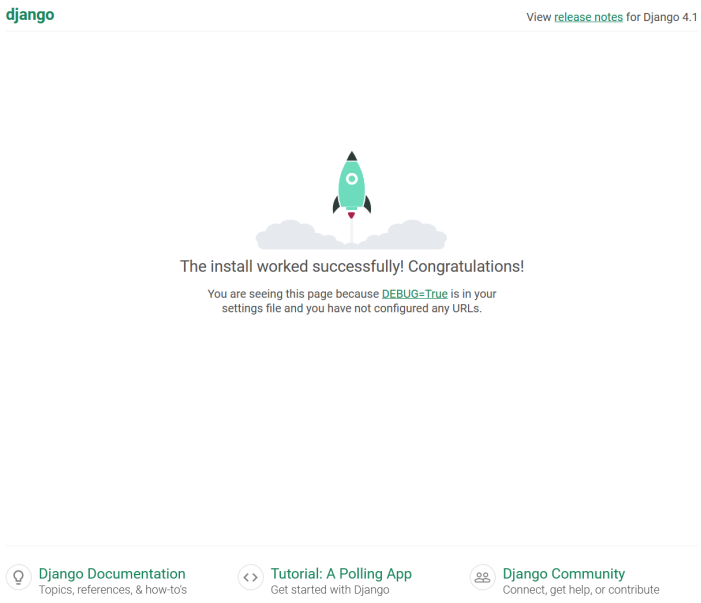


Figure 5.6: Django Set Up

2. Create a Project

Using the command django-admin startproject projectname after installing Django will allow you to start a new project. Change "projectname" to the name you've chosen for your project. The files and directories required to begin creating your Django application will be created in a new directory with the project name supplied by this command.

3. Navigate to Project Directory

Use the cd command in the terminal or command prompt to get to the project directory. The majority of your development work will be done in this area.

4. Run the Development Server

Use the python manage.py runserver command to test your Django project on the development server. Once the server has been started, you can view your application in a web browser using the local server address provided (often http://127.0.0.1:8000/).

5. Create Django Apps

Django employs a modular design with reusable parts known as "apps." Python manage.py startapp appname is the command you can use to build applications within your project. Change "appname" to the name you want for your app. A new directory for the app will be created as a result, containing files for the models, views, templates, and other components.

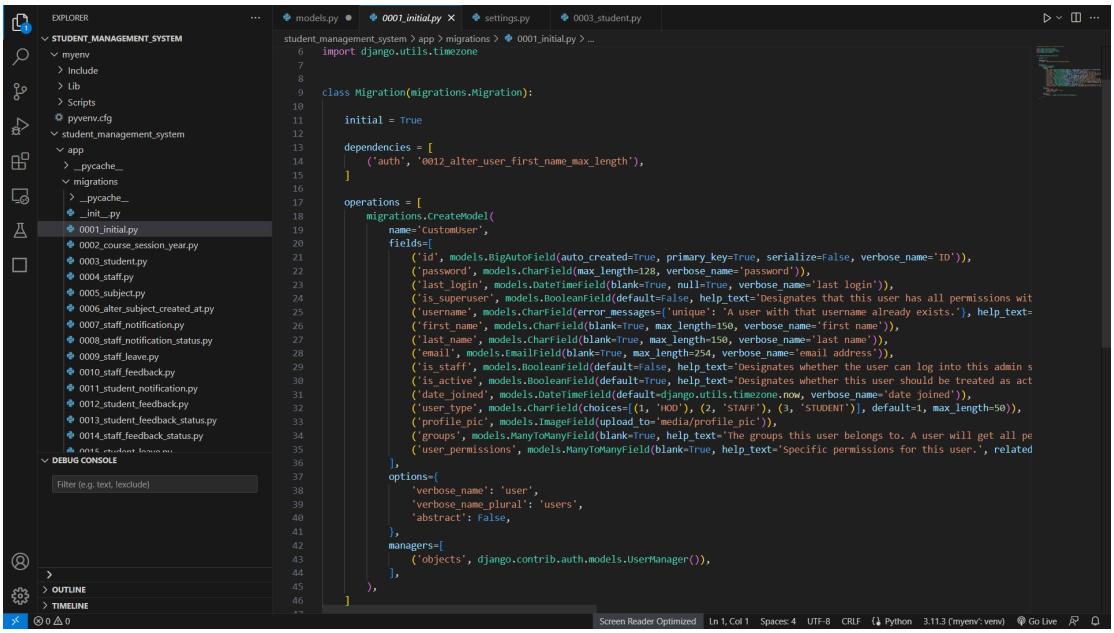


Figure 5.7: Django Set Up

6. Configure Database

Multiple database backends are supported by Django. For development purposes, it automatically uses SQLite. By defining the database engine, name, user, password, and other pertinent information in the project's settings.py file, you can customise the database settings.

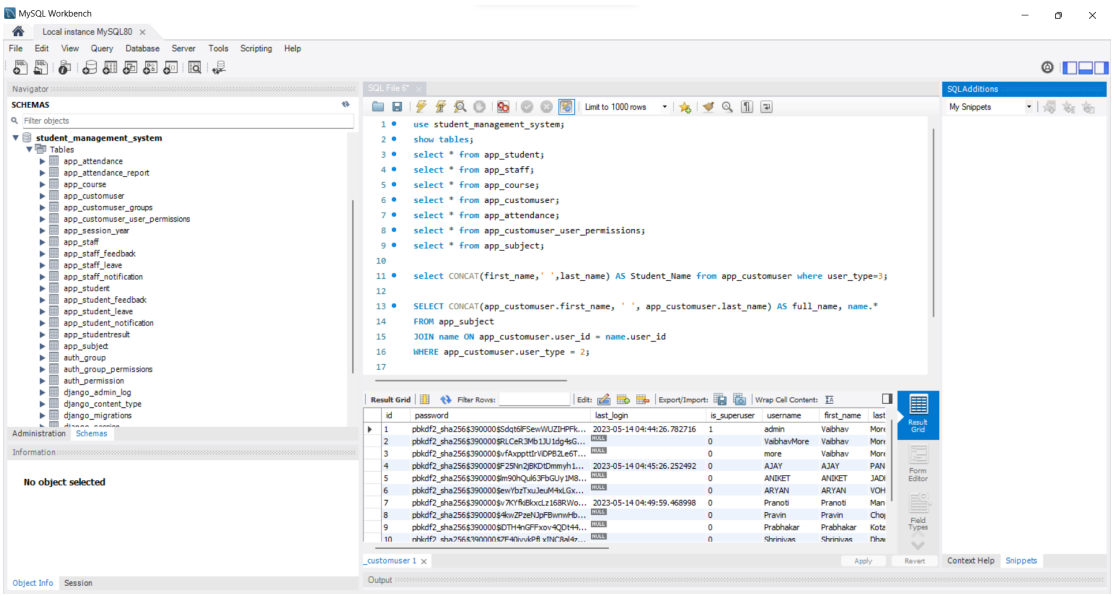


Figure 5.8: MYSQL

7. Define Models

Models represent the database tables and specify how your data is organised and related. Create a models.py file for each project, and use Django's model fields and relationships to specify the required models.

Models are essential in Django because they specify the structure and behaviour of data within a web application. By encapsulating the qualities and relationships of the data, they act as a bridge between the application and the database. The sorts of data stored, such as text, numbers, dates, or file uploads, are determined by the fields that models establish.

Additionally, they provide connections across various models that can be one-to-one, one-to-many, or many-to-many. Models may include techniques for applying business logic, performing operations on the data, or obtaining relevant data from a database. While adhering to the principles of database normalisation and maintainable code architecture, developers can easily manage and manipulate data by using models.

An academic monitoring system using Django can be implemented using a template that includes several files. Here is an explanation of each file:

- settings.py: This file contains the settings for the Django project, including the database settings, installed apps, middleware, and static files.
- urls.py: This file contains the URL patterns for the Django project, which map URLs to views.
- wsgi.py: This file contains the configuration for the web server gateway interface (WSGI) that runs the Django requirements.txt: This file contains a list of Python packages that are required for the Django project to run.
- manage.py: This file is used to manage the Django project, including running migrations, starting and stopping the development server, and creating superusers.
- models.py: This file contains the database models for the Django project, which define the structure of the data stored in the database.
- views.py: This file contains the views for the Django project, which handle HTTP requests and return HTTP responses.
- templates: This directory contains the HTML templates for the Django project, which define the structure and layout of the web pages.
- static: This directory contains the static files for the Django project, such as CSS, JavaScript, and images.
- admin.py: This file contains the admin interface for the Django project, which allows administrators to manage the database.
- migrations: This directory contains the migration files for the Django project, which are used to create and modify the database schema.
- tests.py: This file contains the unit tests for the Django project, which are used to ensure that the code is working correctly.

  Overall, the academic monitoring system using Django template includes several files that work together to create a functional academic monitoring system. By customizing and extending these files, developers can create a system tailored to their specific needs and requirements.

Essential Technologies used

HTML, CSS, and Python are used to design the front-end user interface, style the user interface, and manage the system's back-end logic in an academic monitoring project. Each technology is described below:

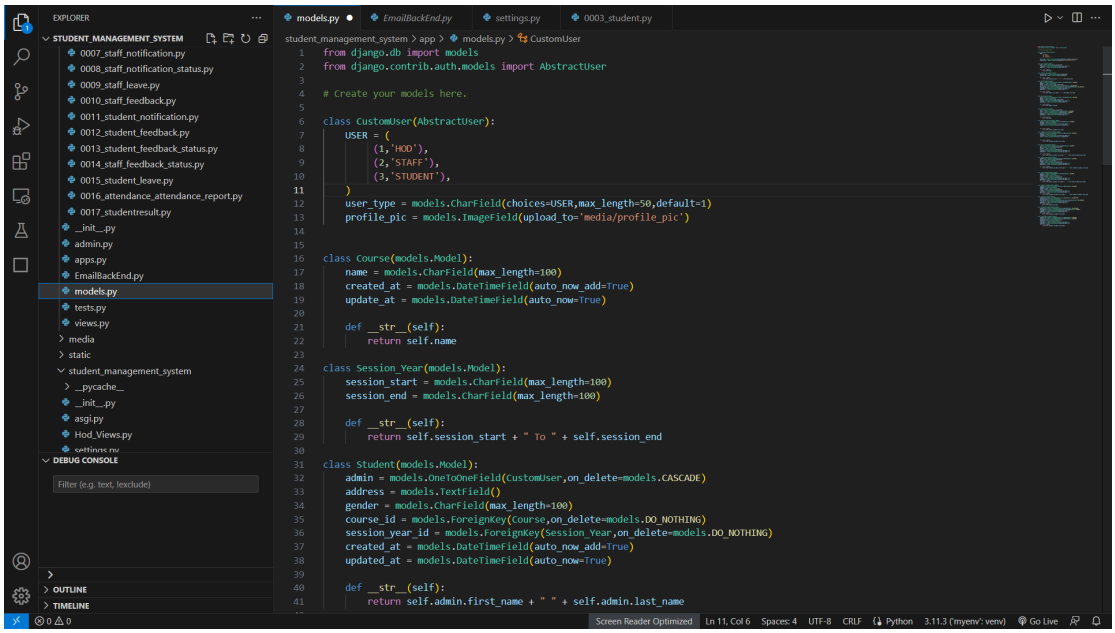
The structure and content of web pages are created using HTML (Hypertext Markup Language). The structure and content of a web page are defined using tags in the markup language HTML. For instance, the body tag is used to define the section of a web page that contains the content, while the head tag is used to define the section that contains the content. HTML is used to specify user interface elements like buttons, as well as the layout and design of web pages.

CSS: Web pages are styled using CSS (Cascading Style Sheets). A web page's visual style and layout are specified using the stylesheet language CSS. The colours, fonts, backgrounds, and other visual components of web pages are defined using CSS. In order to build a unified and aesthetically pleasing user experience across several web pages, CSS is utilised.

Python is used to manage the project's back-end logic for academic monitoring. Python is a simple to learn and use high-level programming language. The code that communicates with the database, fetches data, and processes it is written in Python. The code for user login, authorization, and other security features is likewise written in Python. Python is used to create reports and analytics as well as the programming that notifies students and teachers of events.

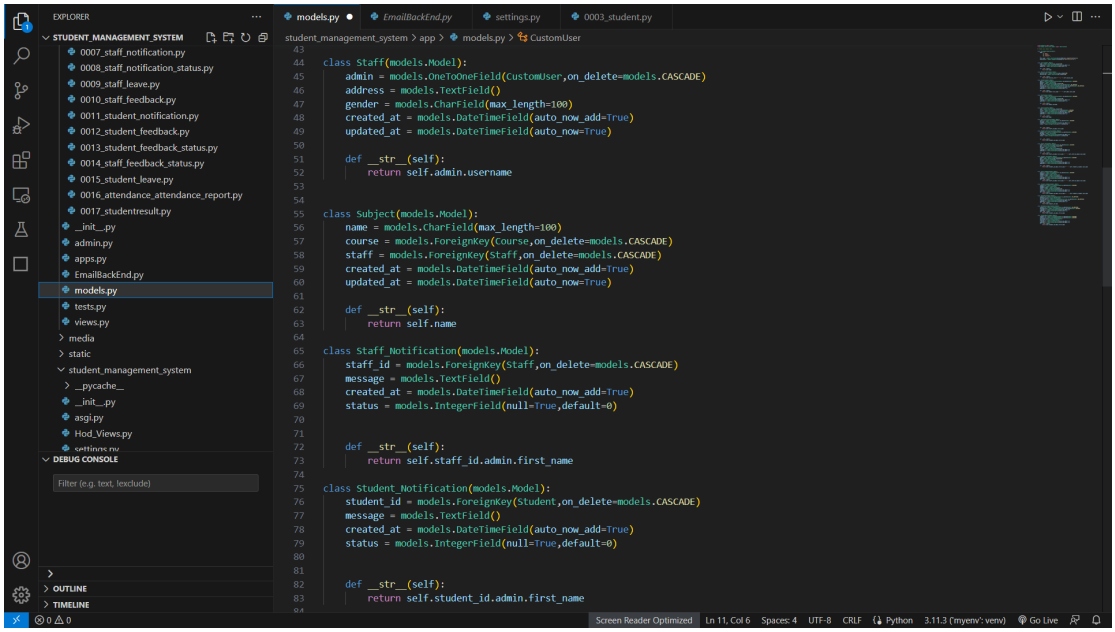
Overall, the technologies of HTML, CSS, and Python are crucial for developing an academic monitoring project. Developers may design a user interface that is aesthetically pleasing, user-friendly, and safe by utilising these technologies.

Models



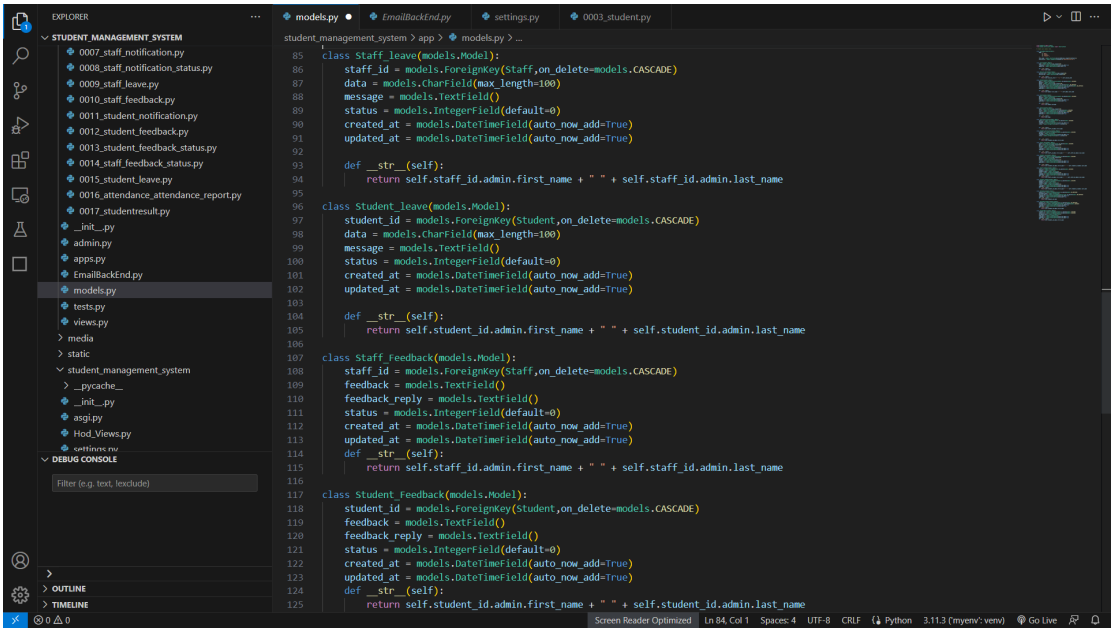
```python
from django.db import models
from django.contrib.auth.models import AbstractUser

# Create your models here.

class CustomUser(AbstractUser):
    USER = (
        (1,'HOD'),
        (2,'STAFF'),
        (3,'STUDENT'),
    )
    user_type = models.CharField(choices=USER,max_length=50,default=1)
    profile_pic = models.ImageField(upload_to='media/profile_pic')


class Course(models.Model):
    name = models.CharField(max_length=100)
    created_at = models.DateTimeField(auto_now_add=True)
    update_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.name

class Session_Year(models.Model):
    session_start = models.CharField(max_length=100)
    session_end = models.CharField(max_length=100)

    def __str__(self):
        return self.session_start + " To " + self.session_end

class Student(models.Model):
    admin = models.OneToOneField(CustomUser,on_delete=models.CASCADE)
    address = models.TextField()
    gender = models.CharField(max_length=100)
    course_id = models.ForeignKey(Course,on_delete=models.DO_NOTHING)
    session_year_id = models.ForeignKey(Session_Year,on_delete=models.DO_NOTHING)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.admin.first_name + " " + self.admin.last_name
```

Figure 5.9: Model



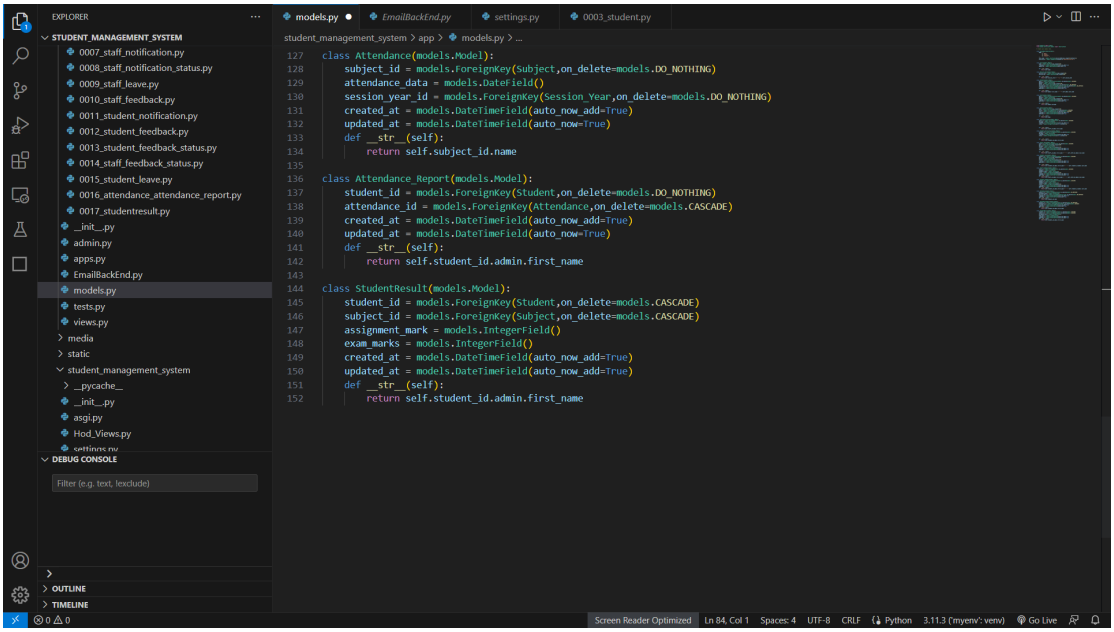
```python
class Staff(models.Model):
    admin = models.OneToOneField(CustomUser,on_delete=models.CASCADE)
    address = models.TextField()
    gender = models.CharField(max_length=100)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.admin.username


class Subject(models.Model):
    name = models.CharField(max_length=100)
    course = models.ForeignKey(Course,on_delete=models.CASCADE)
    staff = models.ForeignKey(Staff,on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.name

class Staff_Notification(models.Model):
    staff_id = models.ForeignKey(Staff,on_delete=models.CASCADE)
    message = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    status = models.IntegerField(null=True,default=0)


    def __str__(self):
        return self.staff_id.admin.first_name

class Student_Notification(models.Model):
    student_id = models.ForeignKey(Student,on_delete=models.CASCADE)
    message = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    status = models.IntegerField(null=True,default=0)


    def __str__(self):
        return self.student_id.admin.first_name
```

Figure 5.10: Model



```
class Staff_leave(models.Model):
    staff_id = models.ForeignKey(Staff,on_delete=models.CASCADE)
    data = models.CharField(max_length=100)
    message = models.TextField()
    status = models.IntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.staff_id.admin.first_name + " " + self.staff_id.admin.last_name

class Student_leave(models.Model):
    student_id = models.ForeignKey(Student,on_delete=models.CASCADE)
    data = models.CharField(max_length=100)
    message = models.TextField()
    status = models.IntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.student_id.admin.first_name + " " + self.student_id.admin.last_name

class Staff_Feedback(models.Model):
    staff_id = models.ForeignKey(Staff,on_delete=models.CASCADE)
    feedback = models.TextField()
    feedback_reply = models.TextField()
    status = models.IntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.staff_id.admin.first_name + " " + self.staff_id.admin.last_name

class Student_Feedback(models.Model):
    student_id = models.ForeignKey(Student,on_delete=models.CASCADE)
    feedback = models.TextField()
    feedback_reply = models.TextField()
    status = models.IntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.student_id.admin.first_name + " " + self.student_id.admin.last_name
```

Figure 5.11: Model



```
class Attendance(models.Model):
    subject_id = models.ForeignKey(Subject,on_delete=models.DO_NOTHING)
    attendance_data = models.DateField()
    session_year_id = models.ForeignKey(Session_Year,on_delete=models.DO_NOTHING)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    def __str__(self):
        return self.subject_id.name

class Attendance_Report(models.Model):
    student_id = models.ForeignKey(Student,on_delete=models.DO_NOTHING)
    attendance_id = models.ForeignKey(Attendance,on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    def __str__(self):
        return self.student_id.admin.first_name

class StudentResult(models.Model):
    student_id = models.ForeignKey(Student,on_delete=models.CASCADE)
    subject_id = models.ForeignKey(Subject,on_delete=models.CASCADE)
    assignment_mark = models.IntegerField()
    exam_marks = models.IntegerField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.student_id.admin.first_name
```

Figure 5.12: Model

How Model File Works

This is a Django model for academic monitoring system. It includes several entities, each with its own set of fields and relationships.

Here's a brief explanation of each entity:

CustomUser: This entity represents the user accounts in the system. It inherits from

Django's
AbstractUser :model and adds fields for the user's type, profile picture, and other user-related information.
Course: This entity represents the courses offered by the institution. It has fields for the course name, creation and update timestamps, and a many-to-many relationship with the Session_Year entity.
Session_Year: This entity represents the academic sessions, such as the first year, second year, or third year. It has fields for the start and end dates of the session.
Student: This entity represents the students enrolled in the courses. It has fields for the student's address, gender, course ID, session year ID, creation and update timestamps, and a many-to-one relationship with the CustomUser entity.
Staff: This entity represents the staff members who teach the courses in the academic session. It has fields for the staff member's address, gender, creation and update timestamps, and a many-to-one relationship with the CustomUser entity.
Subject: This entity represents the subjects taught in the courses. It has fields for the subject name, course ID, staff ID, creation and update timestamps, and a many-to-many relationship with the Course and Staff entities.
Staff_Notification: This entity represents the notifications sent to the staff members. It has fields for the staff ID, notification message, creation and update timestamps, and a status field to indicate whether the notification has been read or not.
Student_Notification: This entity represents the notifications sent to the students. It has fields for the student ID, notification message, creation and update timestamps, and a status field to indicate whether the notification has been read or not.
Staff_leave: This entity represents the leaves taken by the staff members. It has fields for the staff ID, leave date, leave reason, status field to indicate whether the leave has been approved or not, creation and update timestamps.
Student_leave: This entity represents the leaves taken by the students. It has fields for the student ID, leave date, leave reason, status field to indicate whether the leave has been approved or not, creation and update timestamps.
Staff_Feedback: This entity represents the feedback submitted by the staff members. It has fields for the staff ID, feedback message, feedback reply, status field to indicate whether the feedback has been read or not, creation and update timestamps.
Student_Feedback: This entity represents the feedback submitted by the students. It

has fields for the student ID, feedback message, feedback reply, status field to indicate whether the feedback has been read or not, creation and update timestamps.

Attendance: This entity represents the attendance records for the courses. It has fields for the subject ID, attendance date, session year ID, creation and update timestamps, and a many-to-one relationship with these Subject and Session_Year entities.

Attendance_Report: This entity represents the attendance reports for the students. It has fields for the student ID, attendance ID, creation and update timestamps, and a many-to-one relationship with the Student Attendance and Subject entities.

StudentResult: This entity represents the results of the students' courses. It has fields for the student ID, subject ID, assignment mark, exam mark, creation and update timestamps, and a many-to-one relationship with the Student, Subject, and Session_Year entities.

8. Run Migrations

Django uses migrations to control changes to the database schema. Use the python manage.py makemigrations command to create migrations after specifying models. Based on the modifications found in the models, migration files will be generated. Utilise Python's manage.py migrate to implement the migrations on the database.

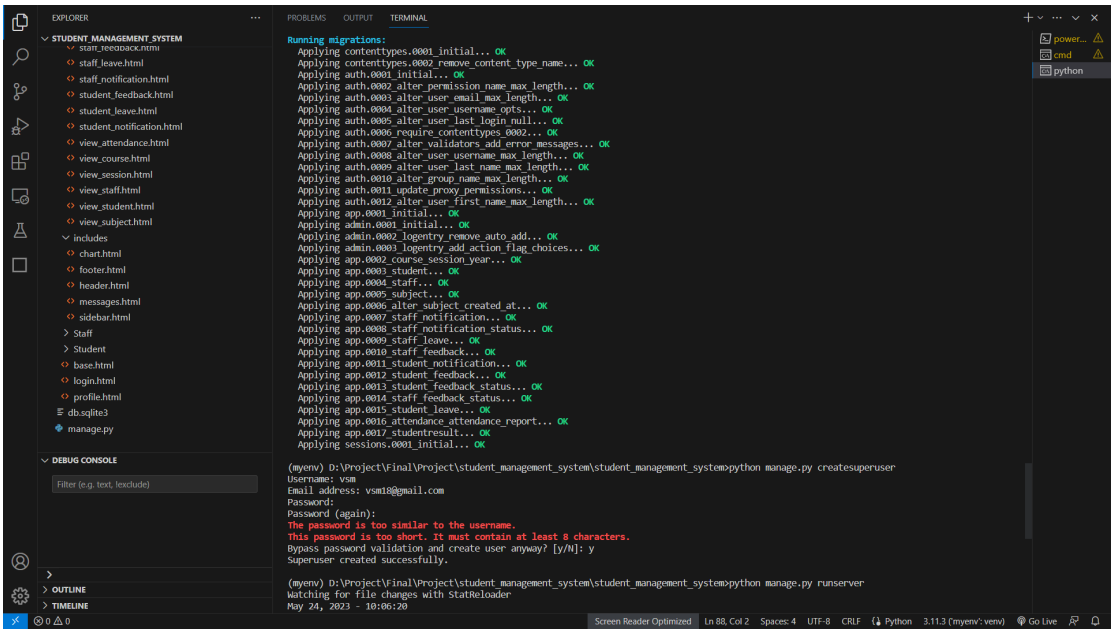


Figure 5.13: Migration

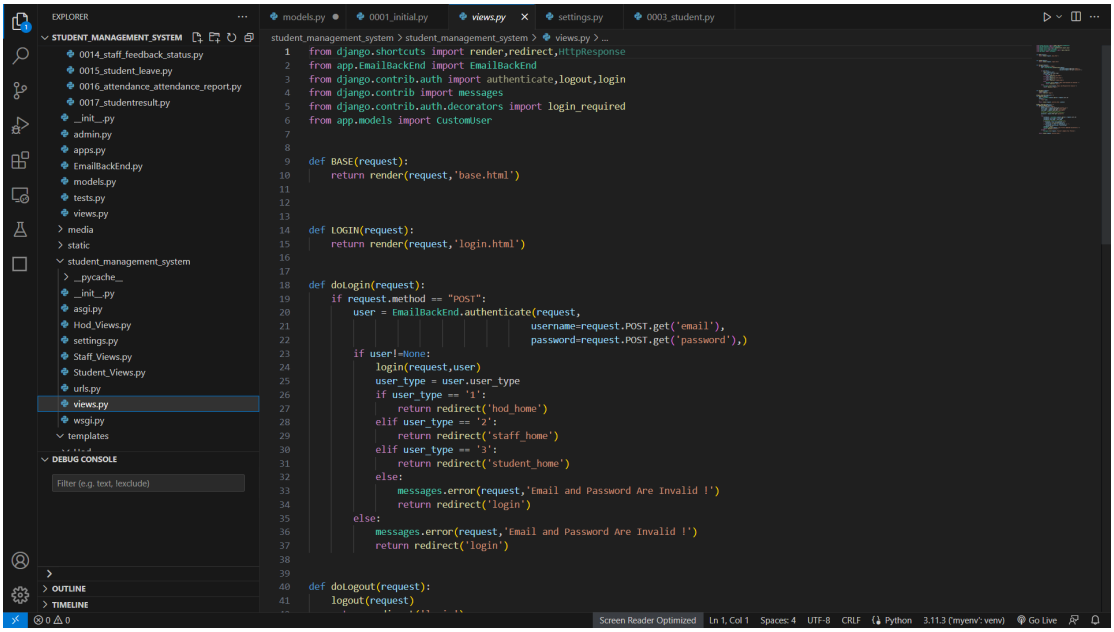
9. Create Views and Templates

Views communicate with models and templates while handling your application's logic. Within the app's views.py file, define functions or classes to create views. Templates provide presentation logic and HTML markup. In the app's templates directory, create templates.For constructing web applications in Django, views and templates are crucial elements. Views are responsible for handling the logic involved in handling requests and producing responses. They respond to incoming requests, work with models to retrieve or alter data, and then deliver the data to templates. On the other hand, the application's presentation layer is defined by templatesements.

Views are responsible for handling the logic involved in handling requests and producing responses. They respond to incoming requests, work with models to retrieve or alter data, and then deliver the data to templates. On the other hand, the application's presentation layer is defined by templates. For the purpose of presenting data dynamically, they contain HTML markup and might also have variables and template tags. In

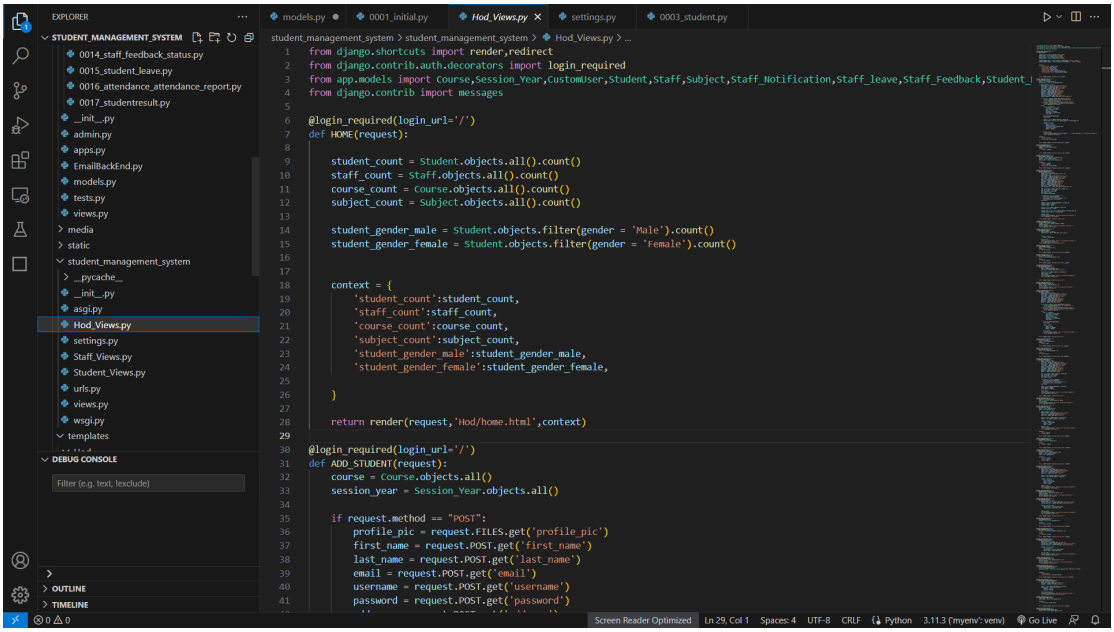
order to offer a smooth user experience, views and templates collaborate. Views gather the relevant information, transmit it to templates, and then render the finished HTML output, which is then sent back to the user's browser. Django enhances code organisation and reusability by separating the logic (views) from the presentation (templates).

All Views



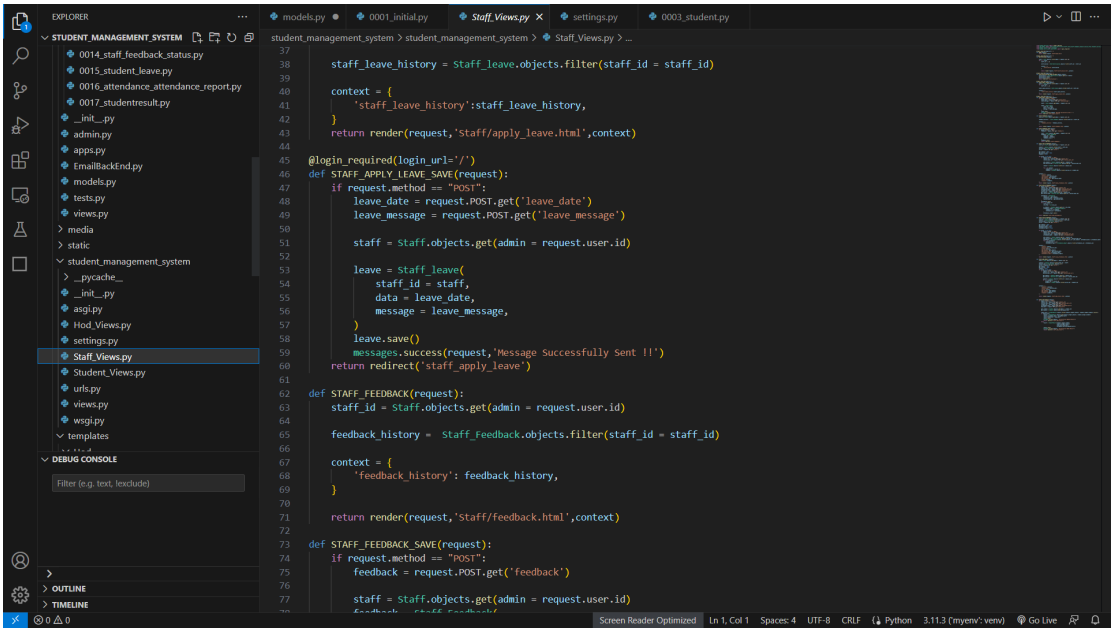
```python
from django.shortcuts import render,redirect,HttpResponse
from app.EmailBackEnd import EmailBackEnd
from django.contrib.auth import authenticate,logout,login
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from app.models import CustomUser


def BASE(request):
    return render(request,'base.html')



def LOGIN(request):
    return render(request,'login.html')


def doLogin(request):
    if request.method == "POST":
        user = EmailBackEnd.authenticate(request,
                                         username=request.POST.get('email'),
                                         password=request.POST.get('password'),)
        if user!=None:
            login(request,user)
            user_type = user.user_type
            if user_type == '1':
                return redirect('hod_home')
            elif user_type == '2':
                return redirect('staff_home')
            elif user_type == '3':
                return redirect('student_home')
            else:
                messages.error(request,'Email and Password Are Invalid !')
                return redirect('login')
        else:
            messages.error(request,'Email and Password Are Invalid !')
            return redirect('login')


def doLogout(request):
    logout(request)
```

Figure 5.14: Views



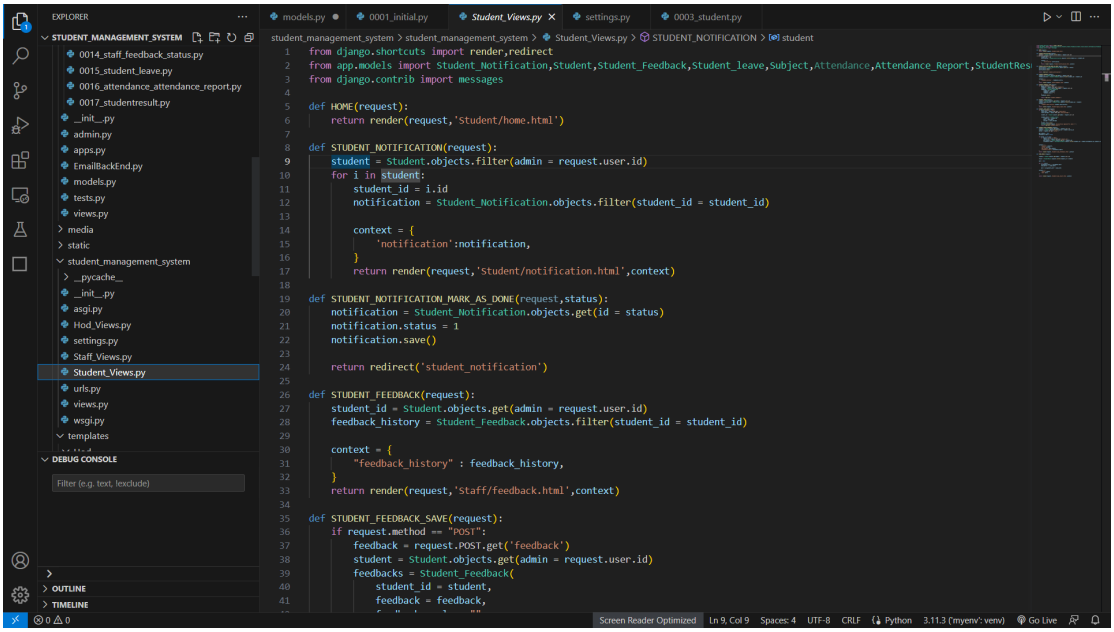
```python
from django.shortcuts import render,redirect
from django.contrib.auth.decorators import login_required
from app.models import Course,Session_Year,CustomUser,Student,Staff,Subject,Staff_Notification,Staff_leave,Staff_Feedback,Student_
from django.contrib import messages

@login_required(login_url='/')
def HOME(request):

    student_count = Student.objects.all().count()
    staff_count = Staff.objects.all().count()
    course_count = Course.objects.all().count()
    subject_count = Subject.objects.all().count()

    student_gender_male = Student.objects.filter(gender = 'Male').count()
    student_gender_female = Student.objects.filter(gender = 'Female').count()


    context = {
        'student_count':student_count,
        'staff_count':staff_count,
        'course_count':course_count,
        'subject_count':subject_count,
        'student_gender_male':student_gender_male,
        'student_gender_female':student_gender_female,

    }

    return render(request,'Hod/home.html',context)

@login_required(login_url='/')
def ADD_STUDENT(request):
    course = Course.objects.all()
    session_year = Session_Year.objects.all()

    if request.method == "POST":
        profile_pic = request.FILES.get('profile_pic')
        first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')
        email = request.POST.get('email')
        username = request.POST.get('username')
        password = request.POST.get('password')
```

Figure 5.15: HOD(Admin) Views



```
    staff_leave_history = Staff_leave.objects.filter(staff_id = staff_id)

    context = {
        'staff_leave_history':staff_leave_history,
    }
    return render(request,'Staff/apply_leave.html',context)

@login_required(login_url='/')
def STAFF_APPLY_LEAVE_SAVE(request):
    if request.method == "POST":
        leave_date = request.POST.get('leave_date')
        leave_message = request.POST.get('leave_message')

        staff = Staff.objects.get(admin = request.user.id)

        leave = Staff_leave(
            staff_id = staff,
            data = leave_date,
            message = leave_message,
        )
        leave.save()
        messages.success(request,'Message Successfully Sent !!')
    return redirect('staff_apply_leave')

def STAFF_FEEDBACK(request):
    staff_id = Staff.objects.get(admin = request.user.id)

    feedback_history =  Staff_Feedback.objects.filter(staff_id = staff_id)

    context = {
        'feedback_history': feedback_history,
    }

    return render(request,'Staff/Feedback.html',context)

def STAFF_FEEDBACK_SAVE(request):
    if request.method == "POST":
        feedback = request.POST.get('feedback')

        staff = Staff.objects.get(admin = request.user.id)
```

Figure 5.16: Staff Views



```
from django.shortcuts import render,redirect
from app.models import Student_Notification,Student,Student_Feedback,Student_leave,Subject,Attendance,Attendance_Report,StudentRes
from django.contrib import messages

def HOME(request):
    return render(request,'Student/home.html')

def STUDENT_NOTIFICATION(request):
    student = Student.objects.filter(admin = request.user.id)
    for i in student:
        student_id = i.id
        notification = Student_Notification.objects.filter(student_id = student_id)

        context = {
            'notification':notification,
        }
        return render(request,'Student/notification.html',context)

def STUDENT_NOTIFICATION_MARK_AS_DONE(request,status):
    notification = Student_Notification.objects.get(id = status)
    notification.status = 1
    notification.save()

    return redirect('student_notification')

def STUDENT_FEEDBACK(request):
    student_id = Student.objects.get(admin = request.user.id)
    feedback_history = Student_Feedback.objects.filter(student_id = student_id)

    context = {
        "feedback_history" : feedback_history,
    }
    return render(request,'Staff/Feedback.html',context)

def STUDENT_FEEDBACK_SAVE(request):
    if request.method == "POST":
        feedback = request.POST.get('feedback')
        student = Student.objects.get(admin = request.user.id)
        feedbacks = Student_Feedback(
            student_id = student,
            feedback = feedback,
```

Figure 5.17: Student Views

Templates

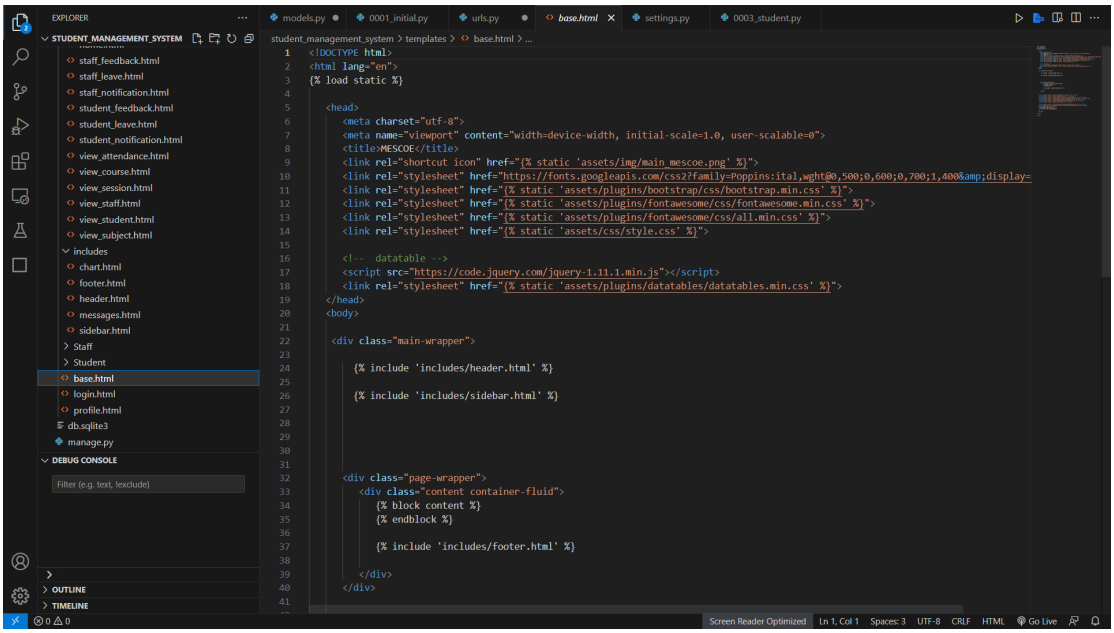


Figure 5.18: Base Template

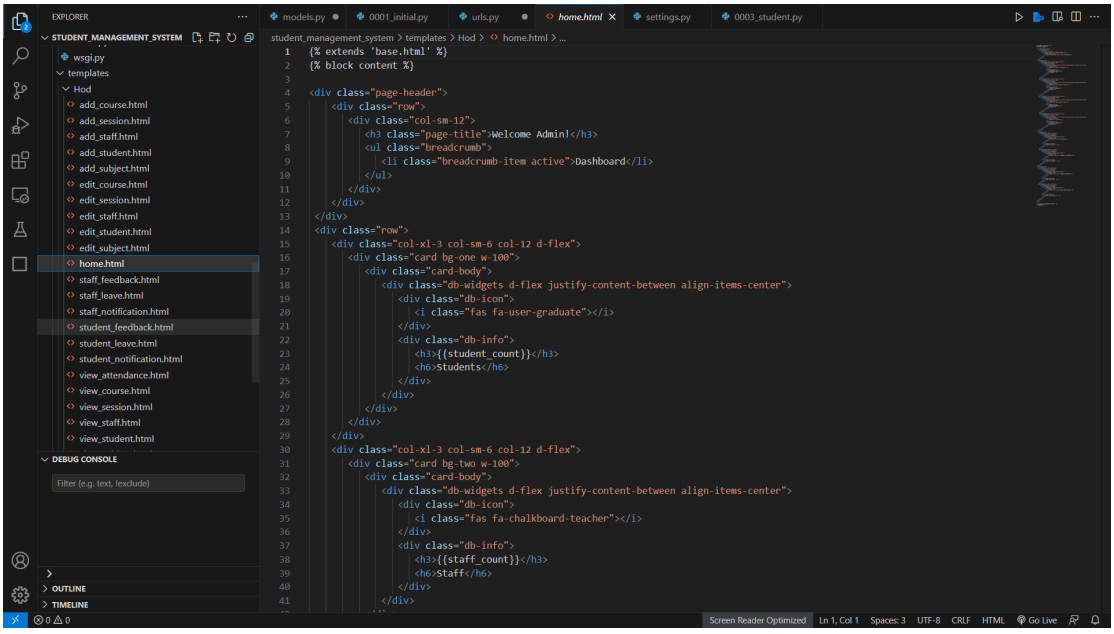


Figure 5.19: Admin Template

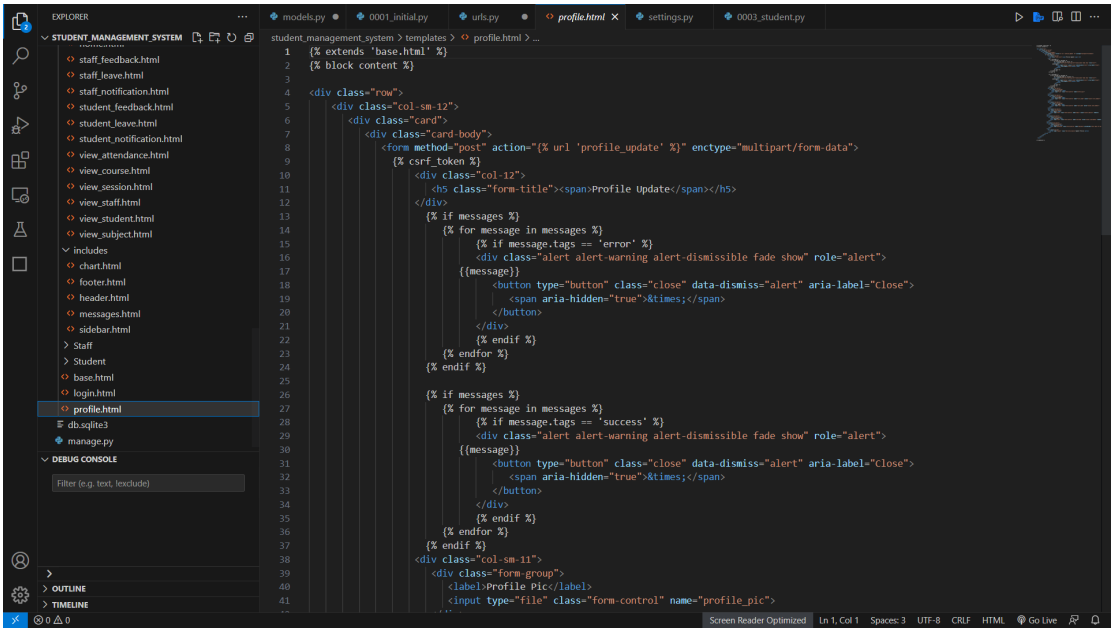


Figure 5.20: Profile Template

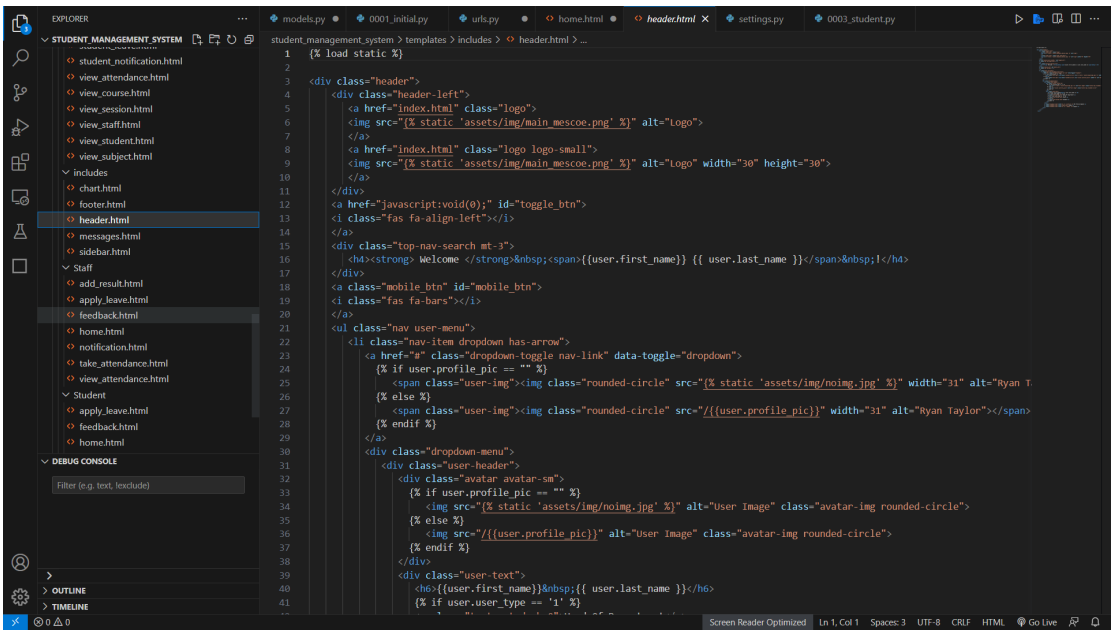


Figure 5.21: Header Template

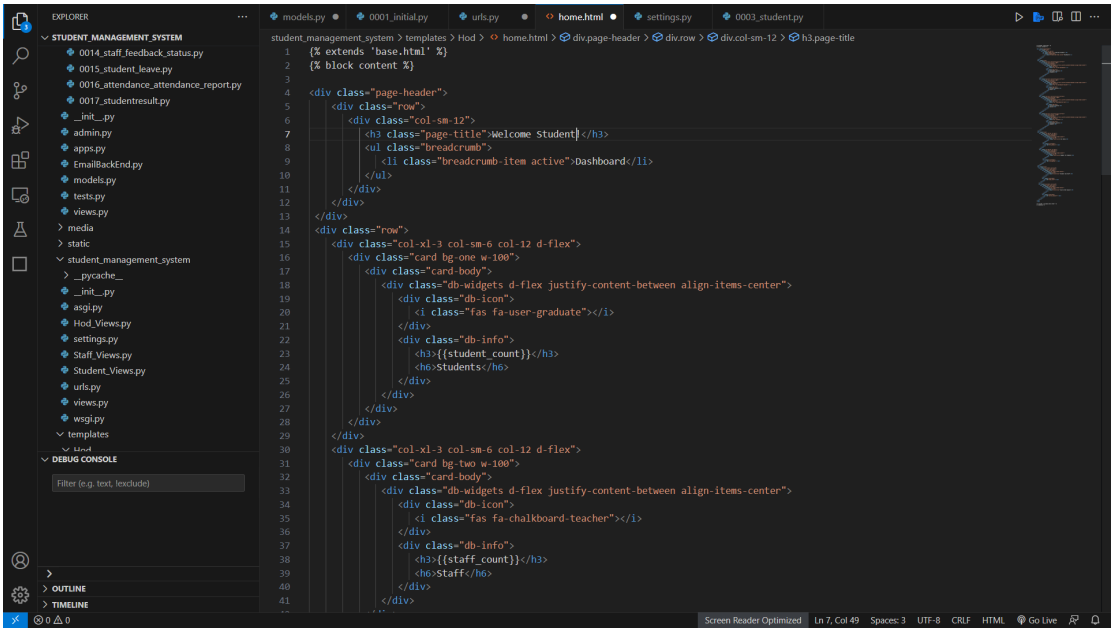


Figure 5.22: Student Template

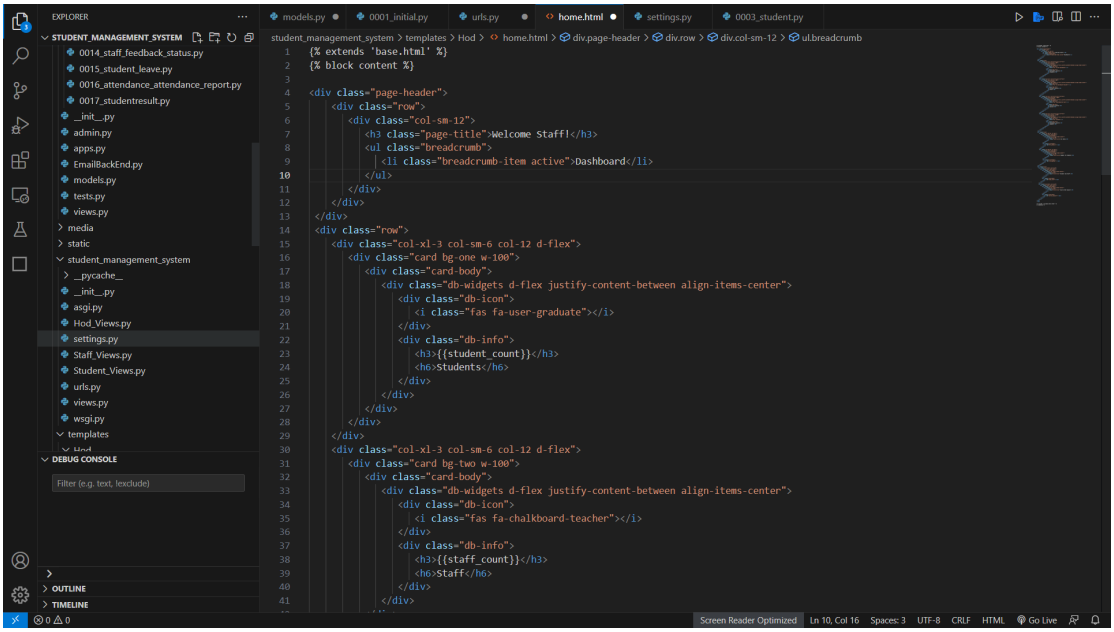
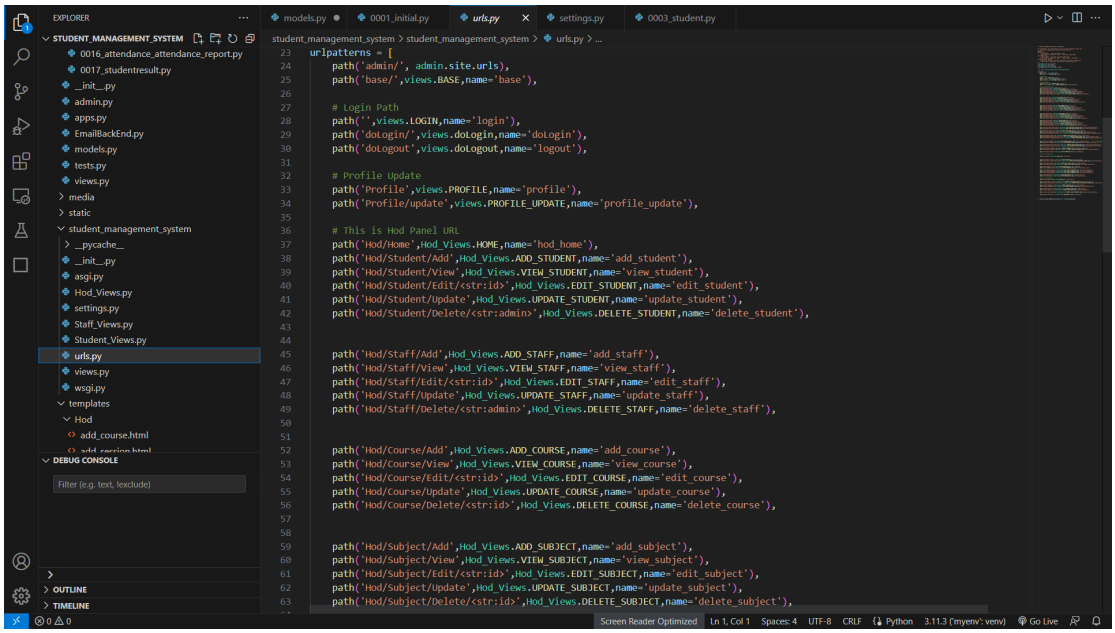


Figure 5.23: Staff Template

10. Define URL Patterns

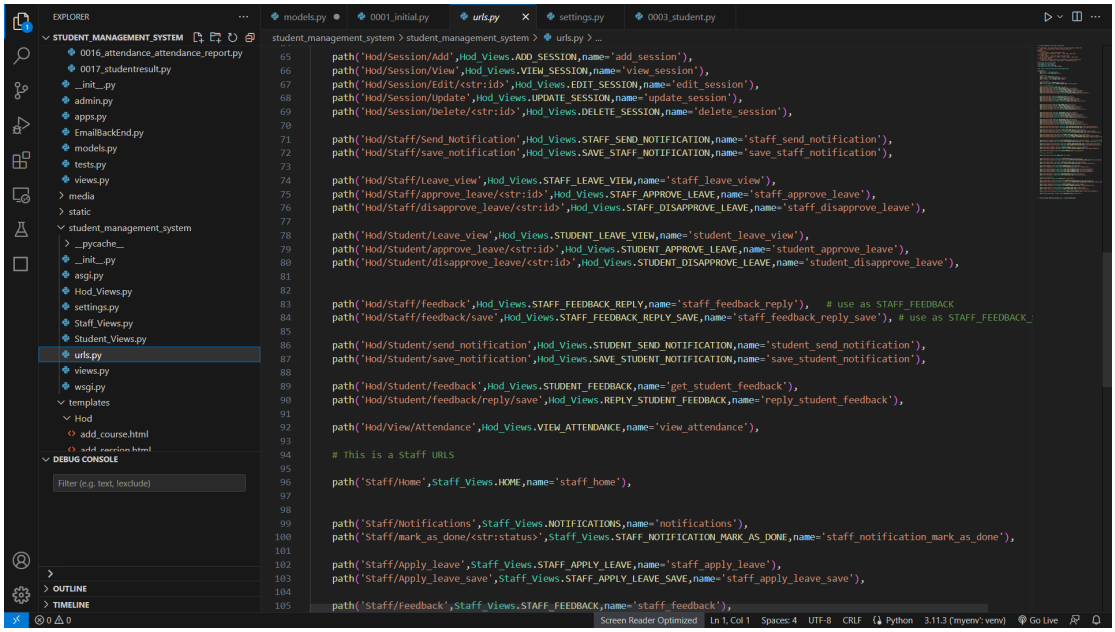
Requests are sent to certain views using URLs. Define URL patterns using regular expressions and link them to the associated views in the app's urls.py file.

URL



```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('base/',views.BASE,name='base'),

    # Login Path
    path('',views.LOGIN,name='login'),
    path('doLogin/',views.doLogin,name='doLogin'),
    path('doLogout',views.doLogout,name='logout'),

    # Profile Update
    path('Profile',views.PROFILE,name='profile'),
    path('Profile/update',views.PROFILE_UPDATE,name='profile_update'),

    # This is Hod Panel URL
    path('Hod/Home',Hod_Views.HOME,name='hod_home'),
    path('Hod/Student/Add',Hod_Views.ADD_STUDENT,name='add_student'),
    path('Hod/Student/View',Hod_Views.VIEW_STUDENT,name='view_student'),
    path('Hod/Student/Edit/<str:id>',Hod_Views.EDIT_STUDENT,name='edit_student'),
    path('Hod/Student/Update',Hod_Views.UPDATE_STUDENT,name='update_student'),
    path('Hod/Student/Delete/<str:admin>',Hod_Views.DELETE_STUDENT,name='delete_student'),


    path('Hod/Staff/Add',Hod_Views.ADD_STAFF,name='add_staff'),
    path('Hod/Staff/View',Hod_Views.VIEW_STAFF,name='view_staff'),
    path('Hod/Staff/Edit/<str:id>',Hod_Views.EDIT_STAFF,name='edit_staff'),
    path('Hod/Staff/Update',Hod_Views.UPDATE_STAFF,name='update_staff'),
    path('Hod/Staff/Delete/<str:admin>',Hod_Views.DELETE_STAFF,name='delete_staff'),


    path('Hod/Course/Add',Hod_Views.ADD_COURSE,name='add_course'),
    path('Hod/Course/View',Hod_Views.VIEW_COURSE,name='view_course'),
    path('Hod/Course/Edit/<str:id>',Hod_Views.EDIT_COURSE,name='edit_course'),
    path('Hod/Course/Update',Hod_Views.UPDATE_COURSE,name='update_course'),
    path('Hod/Course/Delete/<str:id>',Hod_Views.DELETE_COURSE,name='delete_course'),


    path('Hod/Subject/Add',Hod_Views.ADD_SUBJECT,name='add_subject'),
    path('Hod/Subject/View',Hod_Views.VIEW_SUBJECT,name='view_subject'),
    path('Hod/Subject/Edit/<str:id>',Hod_Views.EDIT_SUBJECT,name='edit_subject'),
    path('Hod/Subject/Update',Hod_Views.UPDATE_SUBJECT,name='update_subject'),
    path('Hod/Subject/Delete/<str:id>',Hod_Views.DELETE_SUBJECT,name='delete_subject'),
```

Figure 5.24: HOD URL



```
    path('Hod/Session/Add',Hod_Views.ADD_SESSION,name='add_session'),
    path('Hod/Session/View',Hod_Views.VIEW_SESSION,name='view_session'),
    path('Hod/Session/Edit/<str:id>',Hod_Views.EDIT_SESSION,name='edit_session'),
    path('Hod/Session/Update',Hod_Views.UPDATE_SESSION,name='update_session'),
    path('Hod/Session/Delete/<str:id>',Hod_Views.DELETE_SESSION,name='delete_session'),

    path('Hod/Staff/Send_Notification',Hod_Views.STAFF_SEND_NOTIFICATION,name='staff_send_notification'),
    path('Hod/Staff/save_notification',Hod_Views.SAVE_STAFF_NOTIFICATION,name='save_staff_notification'),

    path('Hod/Staff/Leave_view',Hod_Views.STAFF_LEAVE_VIEW,name='staff_leave_view'),
    path('Hod/Staff/approve_leave/<str:id>',Hod_Views.STAFF_APPROVE_LEAVE,name='staff_approve_leave'),
    path('Hod/Staff/disapprove_leave/<str:id>',Hod_Views.STAFF_DISAPPROVE_LEAVE,name='staff_disapprove_leave'),

    path('Hod/Student/Leave_view',Hod_Views.STUDENT_LEAVE_VIEW,name='student_leave_view'),
    path('Hod/Student/approve_leave/<str:id>',Hod_Views.STUDENT_APPROVE_LEAVE,name='student_approve_leave'),
    path('Hod/Student/disapprove_leave/<str:id>',Hod_Views.STUDENT_DISAPPROVE_LEAVE,name='student_disapprove_leave'),


    path('Hod/Staff/feedback',Hod_Views.STAFF_FEEDBACK_REPLY,name='staff_feedback_reply'),   # use as STAFF_FEEDBACK
    path('Hod/Staff/feedback/save',Hod_Views.STAFF_FEEDBACK_REPLY_SAVE,name='staff_feedback_reply_save'), # use as STAFF_FEEDBACK_

    path('Hod/Student/send_notification',Hod_Views.STUDENT_SEND_NOTIFICATION,name='student_send_notification'),
    path('Hod/Student/save_notification',Hod_Views.SAVE_STUDENT_NOTIFICATION,name='save_student_notification'),

    path('Hod/Student/feedback',Hod_Views.STUDENT_FEEDBACK,name='get_student_feedback'),
    path('Hod/Student/feedback/reply/save',Hod_Views.REPLY_STUDENT_FEEDBACK,name='reply_student_feedback'),

    path('Hod/View/Attendance',Hod_Views.VIEW_ATTENDANCE,name='view_attendance'),

    # This is a Staff URLS

    path('Staff/Home',Staff_Views.HOME,name='staff_home'),


    path('Staff/Notifications',Staff_Views.NOTIFICATIONS,name='notifications'),
    path('Staff/mark_as_done/<str:status>',Staff_Views.STAFF_NOTIFICATION_MARK_AS_DONE,name='staff_notification_mark_as_done'),

    path('Staff/Apply_leave',Staff_Views.STAFF_APPLY_LEAVE,name='staff_apply_leave'),
    path('Staff/Apply_leave_save',Staff_Views.STAFF_APPLY_LEAVE_SAVE,name='staff_apply_leave_save'),

    path('Staff/Feedback',Staff_Views.STAFF_FEEDBACK,name='staff_feedback'),
```

Figure 5.25: Staff URL

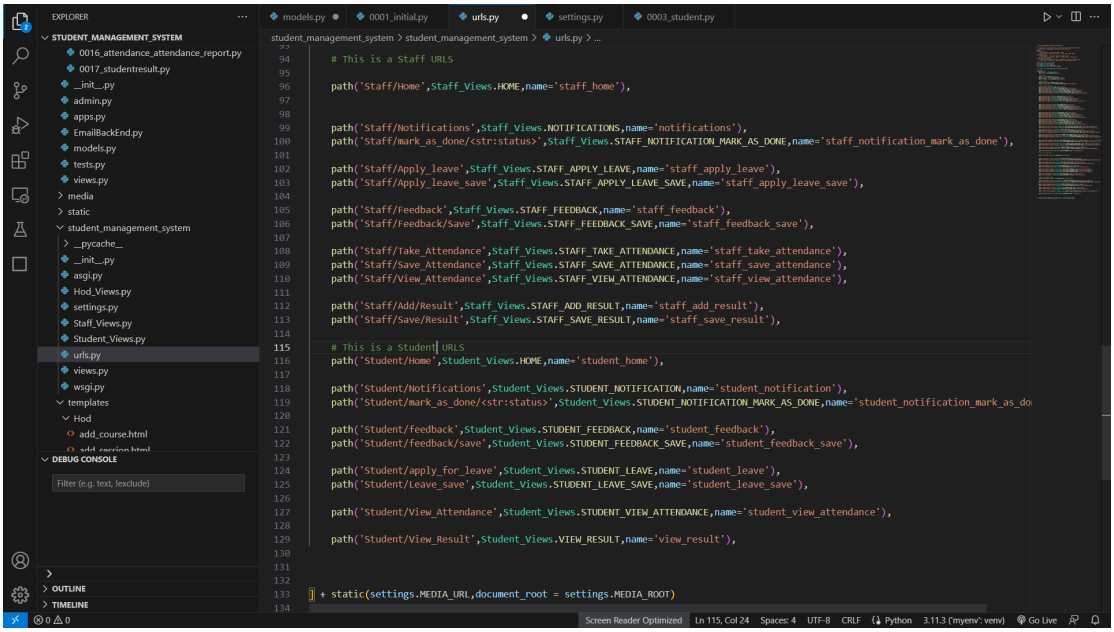


Figure 5.26: Student URL

11. Run the Development Server

You may see your changes take effect in the web application by starting or restarting the development server using the python manage.py runserver command.

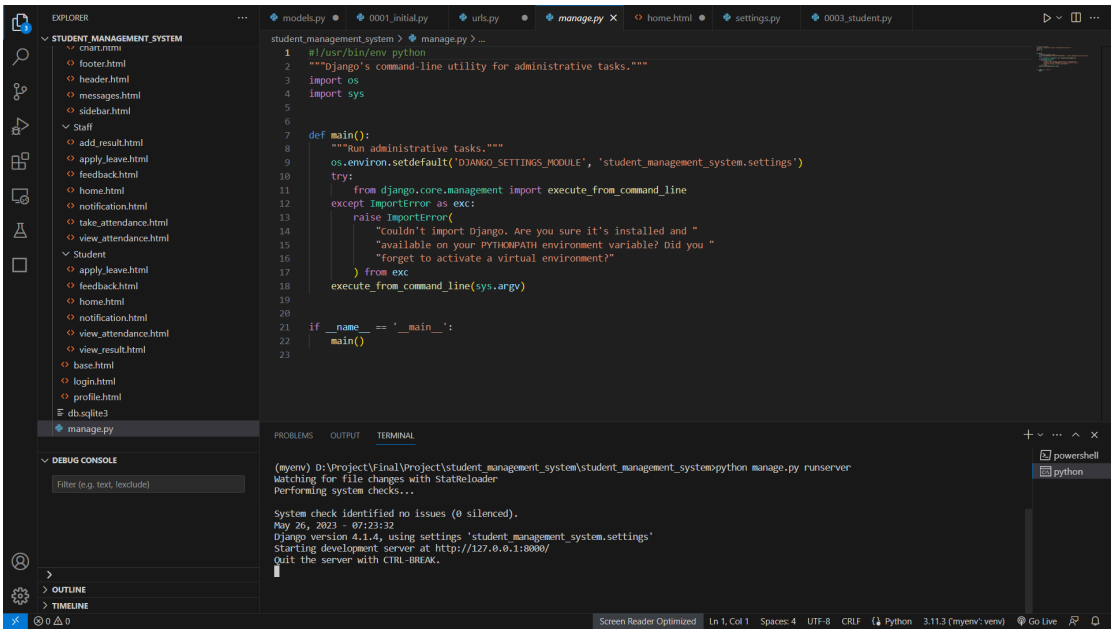


Figure 5.27: Run Server

User Authentication

Any system needs to have user authentication. Through built-in functionality, Django makes user authentication simple. User registration, login, and password-reset functionality are all handled by the system. Only users who have been granted access to the system and permission to act in accordance with their roles can do so thanks to proper authentication.

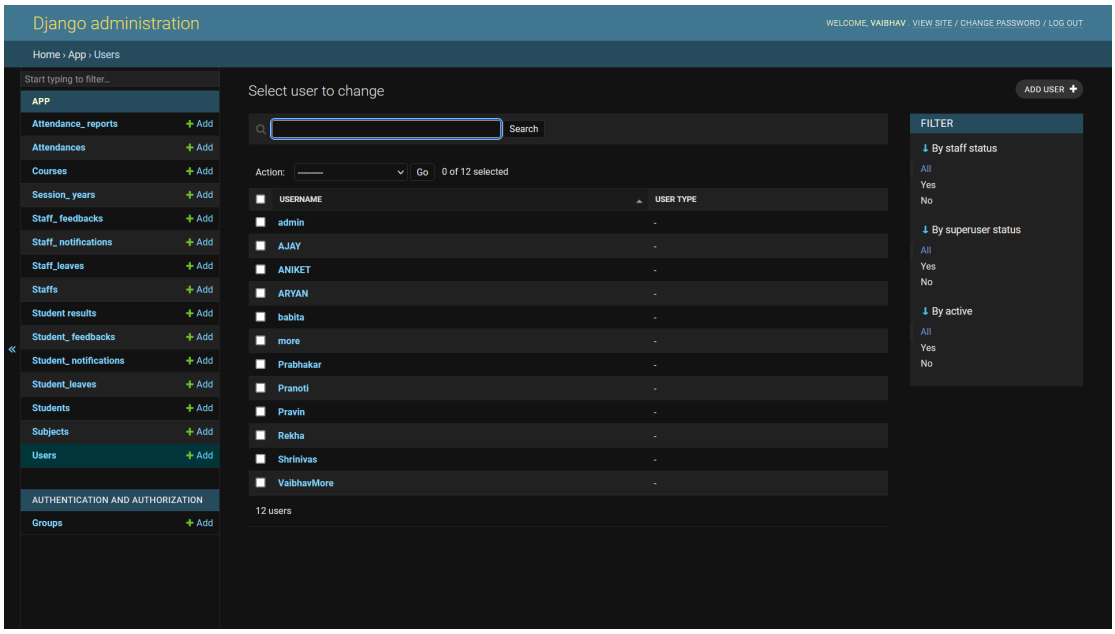


Figure 5.28: Authentication and Authorization

Web applications must have user authentication, and Django comes with a powerful authentication mechanism by default. Developers may quickly add user registration, login, logout, password management, and other authentication-related functionality using Django's authentication framework.

The idea of users and user authentication backends serves as the foundation for Django's authentication mechanism. Users can be created, kept, and managed securely in the database together with their authentication information, such as usernames and passwords.

Django manages the authentication procedure by comparing the submitted credentials to the user data that has been stored when a user logs in. If the credentials are legitimate, Django establishes a session for the user, enabling them to access the application's secured parts.

Forms and Validation

Forms are essential for data input and validation. Django has form-handling tools that make designing and verifying forms easier. Forms can be customised based on unique data requirements and are defined using Python classes. Developers may guarantee data integrity and enforce business rules by utilising Django's form validation features.

Managing user input, validating data, and producing HTML form elements are all made simple in Django by the use of forms. In order to connect the user interface with the underlying data models, forms are used.

Developers can create forms by using Python classes with Django's form system. Text fields, checkboxes, and dropdown menus are examples of form fields, which are the HTML form's input elements. Such field types as CharField, EmailField, IntegerField, and others are available in Django with built-in validation. The definition of custom fields by developers is another option for dealing with certain data types or validation needs.

Form validation ensures that the information entered by the user is accurate and satisfies the requirements. Based on the field types and validation criteria specified in the form class, Django's form validation automatically performs data cleaning and validation. It verifies data types, looks for needed fields, and applies custom validation logic.

The form system in Django supports both client-side and server-side validation. JavaScript is used for client-side validation, which gives consumers immediate feedback without involving a server request. Since client-side validation can be evaded, server-side validation is crucial to guaranteeing the security and integrity of data.

Django processes the form input after it is submitted, validates it, and alerts the user if there are any validation issues. The cleaned and validated data is accessible to developers in the form's cleaned data attribute, where they may use it to carry out additional processing or store it in the database.

Business Logic

Implementing the application's basic features and workflows is part of the Academic Monitoring System's business logic. This involves managing student registration, course enrollment, grade submission, attendance tracking, producing reports, and any other particular requirements specified in the beginning. Typically, the views and models of the Django application include the business logic.

Business logic is the term used to describe the procedures and laws that control how a company runs and manages its data. Business logic, when referring to web applications, is the execution of certain procedures and rules within the application's code. It contains the precise procedures, calculations, and workflows that are peculiar to a given industry or field.

Business logic is often embedded in Django applications within the views and models. Views have the logic needed to process user requests, communicate with models, and produce responses. They handle jobs like validating data, authenticating users, granting access, and enforcing any other business rules that may be necessary.

On the other hand, the structure and behaviour of the data in the application are defined by models. They may contain techniques that encapsulate calculations, sophisticated actions on the data, or business logic relating to data manipulation.

Developers can also design unique modules or classes to house reusable parts of business logic. Because common business rules may be separated into distinct modules and incorporated into numerous views or models, this encourages code reuse and maintainability.

Business logic is essential for ensuring that the application functions in accordance with the unique needs and processes of the business. It helps the programme manage sophisticated data operations, enforce rules, make computations, generate reports, and more. Developers can construct specialised and effective solutions that fit the particular requirements of the business or domain by integrating business logic into the Django application.

Admin Interface

For administrators to manage system data, Django's admin interface offers a user-friendly interface. Administrators can conduct CRUD operations on the data using this automatically produced interface, which is based on the provided models. Administrators may manage the system more easily without having to interact with the database directly by using the admin interface to add, change, and delete records.

The Django admin interface is an effective tool that enables developers and authorised users to control and interact with the data of the application using a simple interface. It offers a ready-made administrative interface that streamlines routine administrative procedures and lowers the amount of work needed to manage the application's data.

Based on the registered models in the application, the admin interface is automatically constructed. It gives users a straightforward way to carry out CRUD (Create, Read, Update, Delete) actions on the database-stored data. Users with the necessary permissions can browse and search for specific data, as well as add, amend, and remove records, through the admin interface.

A variety of tools are available in Django's admin interface to improve the user experience and productivity. Filtering, sorting, and data pagination are all features that can be customised for listing views. Users can edit connected objects immediately inside the admin interface thanks to its support for inline editing.

The behaviour and appearance of the admin interface are completely under developer control. They can define fieldsets, alter the displayed fields, and set access restrictions for particular user groups or groups of users. By altering templates and including custom CSS styles, they can also alter how the admin interface looks and feels.

It is also possible to add customised features to the admin interface. Custom admin actions—functions that carry out particular activities on particular objects—can be created by developers. Additionally, they can create unique admin views to show extra data or carry out sophisticated data processing.

DASHBOARD

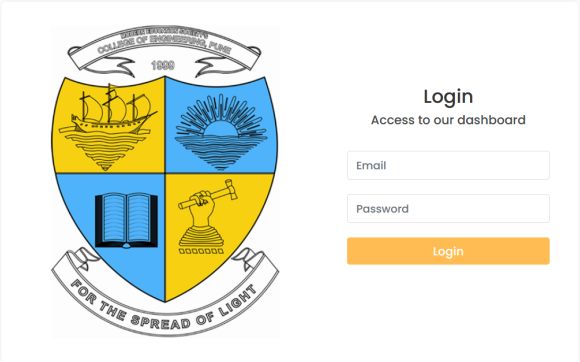


Figure 5.29: Login Page

Figure 5.30: Admin DashBoard

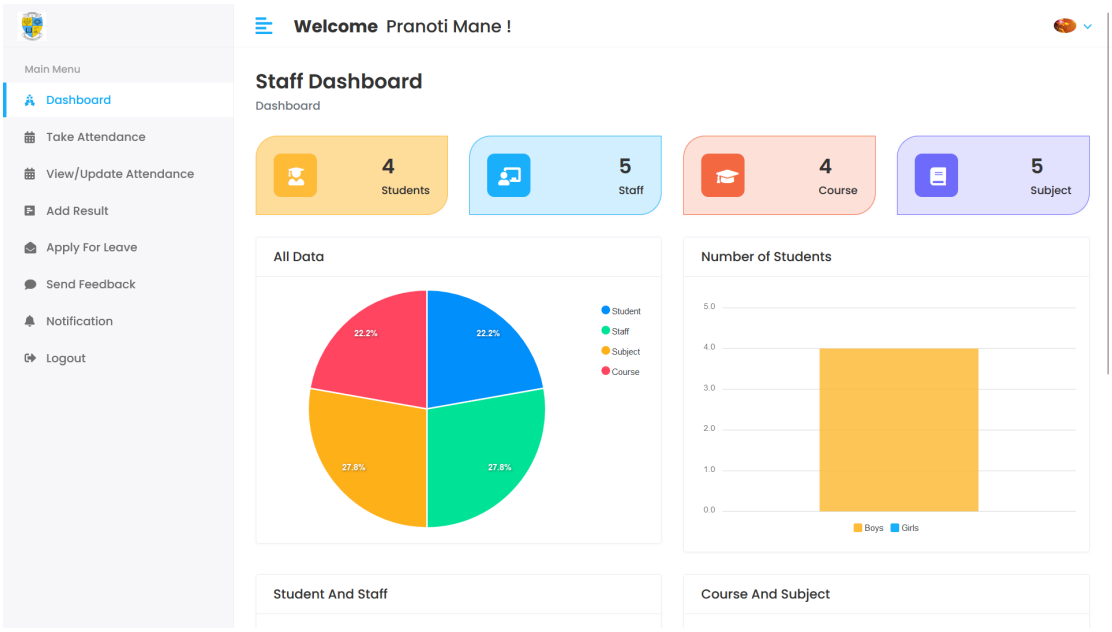


Figure 5.31: Staff DashBoard

Figure 5.32: Student DashBoard

Testing and Debugging

The Academic Monitoring System must undergo extensive testing to guarantee proper operation and dependability. Developers can write test cases to verify the behaviour of the system using the testing framework that Django offers. To ensure that the system works as planned and to find any faults or problems, automated tests can be built to cover a variety of scenarios.

In order to guarantee the dependability, functionality, and quality of a programme, testing and debugging are essential components of software development. Developers can more easily find and fix problems because Django comes with tools and features that help with testing and debugging procedures.

Using the Django testing framework makes testing in Django easier. This framework enables programmers to create test cases to ensure the accuracy of their code. In addition to testing the integration of different components, test cases can be built to validate specific pieces of code, such as models, views, or forms. Assertions and helper functions are provided by Django to make testing easier and more expressive. By executing all specified tests and reporting the results, the manage.py test command can be used to perform test cases.

Django also supports unit tests, functional tests, and integration tests, among other testing paradigms. The primary goal of unit tests is to validate the correct operation of isolated pieces of code. Tests that imitate user interactions with the application help validate its behaviour. To make sure the application's various parts function together without any issues, integration tests examine interactions between them. Comprehensive test cases help developers maintain code stability over time by helping them find and repair errors early in the development cycle.

When it comes to debugging, Django offers a strong debugging framework that aids developers in locating and resolving problems throughout the development process. Detailed error reporting, stack trace details, and interactive debugging sessions are just a few of the capabilities offered by the framework. Developers can analyse variables, walk through code execution, and put breakpoints in their code to help them find the root of faults.

By adding third-party tools and extensions, Django's debugging capabilities can be improved even further. When debugging an application, tools like the Django Debug Toolbar can give you more information about the SQL queries, request and response data, and application performance.

Debugging and testing are fundamental techniques in Django development. While debugging aids in the rapid identification and resolution of problems, testing verifies the accuracy and dependability of the codebase. Developers can create solid, error-free apps by employing a methodical and exhaustive testing strategy and using the debugging tools and functionalities offered by Django.

Deployment

The Academic Monitoring System must undergo extensive testing to guarantee proper operation and dependability. Developers can write test cases to verify the behaviour of the system using the testing framework that Django offers. To ensure that the system works as planned and to find any faults or problems, automated tests can be built to cover a variety of scenarios.

It is crucial to set up the system for deployment before deploying it to a live environment. To do this, the Django application must be configured to run on web servers like Apache or Nginx. Along with setting up a hosting environment, such as a virtual private server or cloud platform, and configuring key infrastructure parts like databases and caching systems, deployment also involves these steps. The protection of sensitive data and preventing unauthorised access also depend on implementing suitable security measures, such as HTTPS encryption and access controls.

A developed Django application is made accessible for usage in a production environment using the procedure known as deployment in the language. It includes setting up databases, managing dependencies, establishing servers, and making sure users can access the application.

The Django application is often hosted during deployment on a web server like Apache or Nginx. Receiving incoming requests and sending them to the Django application for processing is the responsibility of the web server. Setting up virtual hosts, configuring ports, and establishing server rules to handle various request types are all part of configuring the web server.

For the application's data to be stored and retrieved, a database server is also necessary in addition to the web server. Oracle, PostgreSQL, MySQL, and SQLite are just a few of the database backends that Django supports. In addition to preparing the Django application to connect to the database, the deployment procedure also entails configuring the selected database server, creating the required databases and database users, and building the required databases.

The deployment process should also include management of the Django application's dependencies and libraries. To separate the application's dependencies from the

system-level packages, it is standard procedure to employ virtual environments. Virtual environments can be created, managed, and the necessary packages installed using programmes like pipenv or virtualenv.

Deployment must take security seriously. In addition to acquiring and installing SSL certificates, it also entails setting up secure connections using protocols like HTTPS. Sensitive data should be protected, and only authorised users should be able to access particular areas of the application; therefore, appropriate authentication and authorization procedures should be put in place.

Scalability and performance issues must also be taken into account during deployment. Using load balancers or installing on scalable infrastructure can help evenly distribute the strain for the application, which may need to handle increased traffic and demand. To boost performance and cut down on database requests, use caching methods.

The deployment process can be automated using Continuous Integration and Continuous Deployment (CI/CD) practises, enabling frequent upgrades and releases. CI/CD pipelines can be configured to automatically create the application, run tests, and deploy it to production environments, resulting in a streamlined and effective deployment process.

In general, Django deployment entails configuring servers, establishing databases, managing dependencies, ensuring security, and putting monitoring and error handling methods into place. It is essential for making the Django application accessible to users and ensuring its dependable and effective performance in a production setting.

Maintenance and Updates

Once the Academic Monitoring System has been implemented, regular upkeep and upgrades are required to fix any faults, add new features, and enhance functionality. Regular monitoring makes it easier to spot problems and remedy them quickly. A strategy for gathering customer feedback and implementing it into the next update is crucial. Through this iterative approach, the system is kept reliable, user-friendly, and in line with changing requirements.

Any software system, including an academic monitoring system, must have regular maintenance and updates. It's crucial to have a strategy in place for keeping the system operational, secure, and up-to-date beyond the first deployment. Here are some extra details on upkeep and upgrades for an academic monitoring system:

Maintenance

The academic monitoring system requires maintenance in order to operate effectively. This covers commonplace duties like maintaining data integrity, fixing problems, and monitoring system performance. To guarantee optimal system performance, routine maintenance duties may include database optimisation, server monitoring, and performance tuning. In order to protect data in the event of any unanticipated disasters, it is also crucial to build suitable backup and disaster recovery systems.

Updates

To improve the functioning and security of the academic monitoring system, updates are required. Updates may include introducing fresh functionality, enhancing current features, or addressing software flaws. To successfully handle updates, a version control system must be in place. This makes it possible for developers working on various system components to collaborate and track changes, as well as provide rollback options if necessary. To reduce the chance of introducing bugs or disruptions, updates should be fully tested in a staging environment before being released to the production system.

User Feedback and Support

User feedback is essential for pinpointing problem areas and addressing user needs. It is crucial to create channels for users to submit suggestions and support requests. User forums, ticketing systems, and online help desks can all be used for this. Review user comments frequently and prioritise improvements or bug solutions according to user needs. In order to create a great user experience, it is also critical to offer users rapid support and help while promptly responding to any questions or issues they may have.

Security Updates

The academic monitoring system deals with sensitive student data, so maintaining its security is crucial. The best way to find possible security problems is to conduct regular security audits and vulnerability assessments. All parts of the system, including the supporting frameworks, libraries, and server architecture, need to be kept up-to-date with the most recent security patches and updates. To guard against potential security concerns, it is crucial to implement secure coding practises, data encryption, and user authentication procedures.

Documentation and Knowledge Transfer

For the academic monitoring system to be lasting, current documentation must be maintained. System architecture, workflows, configurations, and any customizations should be documented to facilitate knowledge sharing and easier maintenance among team members. In order to ensure that system administrators, instructors, and other users can efficiently utilise and operate the system, documentation and training materials are also crucial.

Documentation and Knowledge Transfer

Based on changing educational requirements and technological breakthroughs, an academic monitoring system should develop and improve. Determine what needs to be improved by routinely evaluating the system's performance, getting user input, and carrying out frequent evaluations. To improve the system's usability and functionality, think about integrating it with other educational systems, adopting cutting-edge fea-

tures, or incorporating new technologies. **Data Security and Privacy**

Academic monitoring systems handle private student and academic data, necessitating stringent security measures. To safeguard the system from unauthorised access and data breaches, suitable access controls, user authentication, and data encryption must be implemented. To protect data privacy and compliance, it is important to follow privacy laws like the Family Educational Rights and Privacy Act (FERPA) and the General Data Protection Regulation (GDPR).

An academic monitoring system must prioritise data security and privacy in order to safeguard the private data of students, teachers, and staff. Here is some further information on data security and privacy in the context of a system for tracking academic performance:

1. Data Encryption

It is essential to use reliable encryption techniques for data storage and transport. As a result, critical data, including student attendance records, grades, and personal information, is securely guarded against unauthorised access. Its security is improved by encryption both during network transmission and while it is stored in databases or other storage devices.

2. User Authentication and Access Control

To guarantee that only authorised users may access the system, reliable user authentication measures should be in place. These include username/password combinations and multi-factor authentication. Based on their roles and responsibilities inside the academic institution, role-based access control (RBAC) can be designed to limit user access to particular features and data.

3. Data Integrity

The integrity of the data kept in the academic monitoring system should be guaranteed by appropriate measures. Utilising procedures for data validation and verification can aid in preventing unauthorised data manipulation. To guard against data loss or corruption, regular backups and data redundancy procedures should also be in place.

4. Compliance with Data Protection Regulations

It is crucial to abide by data protection laws like the Family Educational Rights and

Privacy Act (FERPA) and the General Data Protection Regulation (GDPR). These laws control how personal data is gathered, stored, and used. Institutions must comply with these laws by putting in place suitable security measures and obtaining user consent before processing any personal data.

5. Secure Infrastructure

The academic monitoring system's hosting infrastructure needs to be secure. The use of secure servers, firewalls, intrusion detection systems, and routine security updates are all part of this. To find and fix potential system weaknesses, routine security audits and vulnerability assessments should be carried out.

6. Data Minimization and Anonymization

Only gather and save the information that is required for monitoring academic progress. To protect individual privacy, use personally identifiable information (PII) as little as possible and anonymize data whenever you can. This may entail hiding or encrypting identifiable data and substituting unique identifiers.

7. Secure Communication Channels

Make certain that the lines of communication between users and the academic monitoring system are safe. For web-based interfaces, secure protocols like HTTPS can be used, and secure communication standards can be implemented for APIs or other integrations.

8. Privacy Policies and Consent

Users should be informed of the sorts of data that are collected, how they are used, and who has access to them in the system's privacy policies. Obtain user approval before processing any data, and give consumers tools to control their privacy preferences and data sharing settings.

9. Regular Security Assessments

To find vulnerabilities and potential hazards, do frequent security assessments and penetration testing. This facilitates the proactive fixing of security flaws and the application of required security fixes or updates.

CHAPTER 6

Block Diagram

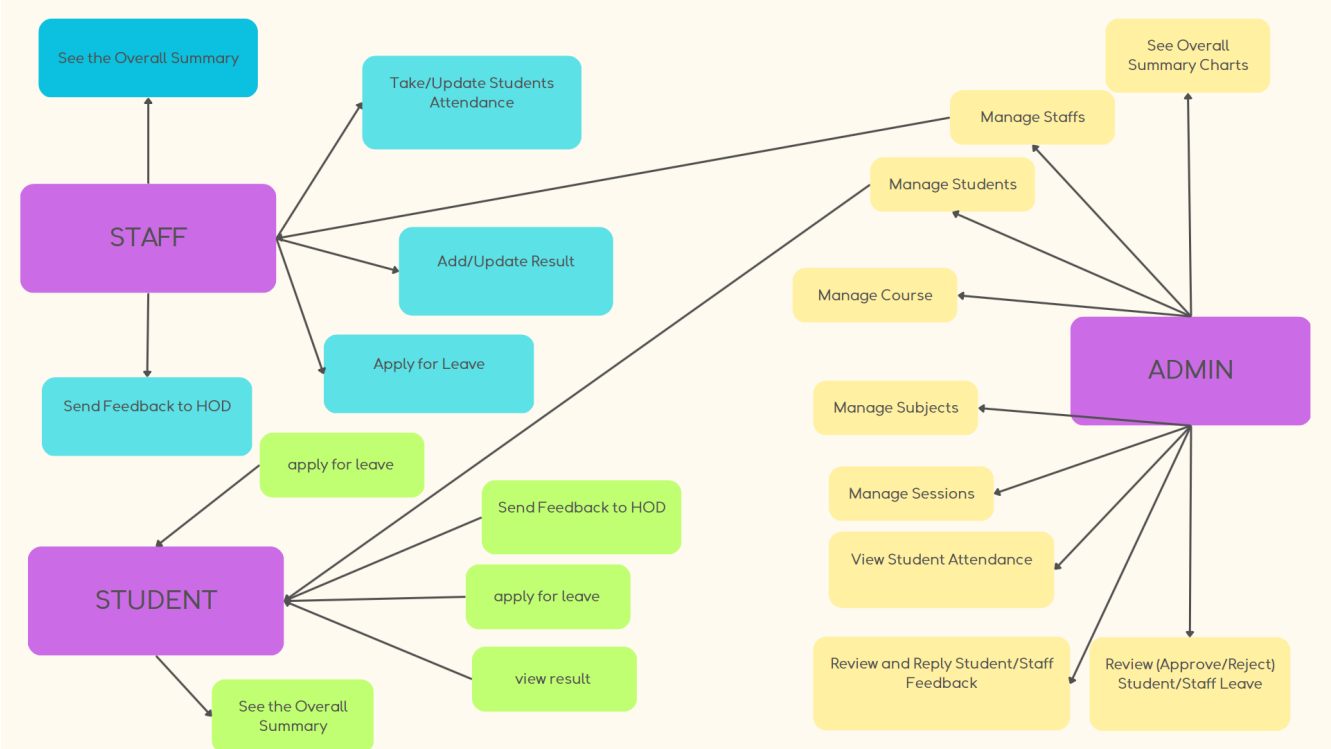


Figure 6.1: BLOCK DIAGRAM

CHAPTER 7

WORKING

A web-based tool called an academic monitoring system enables educational institutions to oversee and track the academic development of its pupils. The system is built to offer a number of capabilities, including course management, leave management, student attendance tracking, and feedback management. The academic monitoring system in this instance is created using the Django web framework, and it includes features like Admin/Staff/Student Login, Add/Edit Course, Add/Edit Staff, Add/Edit Student, Add/Edit Subject, Upload Staff's Picture, Upload Student's Picture, Views Permission , Attendance and Update Attendance, Apply For Leave, Students Can Check Attendance, Reply to Leave Applications, Reply to Feedback, Admin View Attendance.

Users are able to access the system based on their roles thanks to the login capability. Staff and students only have restricted access to the system depending on their authorization, however administrators have unlimited access. Admins can add or edit courses that the institution offers by using the Add/Edit Course capability. Effective course management is made possible by this feature. Similar to that, administrators can add or change staff details like name, designation, contact information, etc. using the Add/Edit Staff function. Administrators can add or amend student information such as name, roll number, contact information, etc. using the Add/Edit Student feature.

Admins can add or edit the subjects the institution offers by using the Add/Edit Subject capability. Effective subject management is made possible by this feature. Admins can upload images of employees and students using the relevant functions, Upload employees Picture and Upload Student Picture. This feature makes it easier to distinguish between staff and students.

Students can apply for leave online by using the Apply For Leave capability. Effectively managing leave requests is made easier by this feature. Students can check their attendance online thanks to the capability that allows them to do so. This function aids

in informing pupils of their attendance status.

Staff members can respond to student leave requests and feedback via the functionality Reply to Leave Applications and Reply to Feedback, respectively. Maintaining contact between faculty and students is made easier by this feature. Administrators can access all students' attendance records using the Admin access Attendance capability. Effectively tracking student attendance is made possible by this function.

Users can update their passwords anytime necessary by using the Password update for Admin, Staff, and Students using setpassword() feature. This function aids in ensuring the security and privacy of data. Users can change profile features including their name and contact information using the Admin Profile Edit, Staff Profile Edit, and Student Profile Edit functions

CHAPTER 8

RESULT

A software programme called an academic monitoring system is used in educational institutions to monitor and control student behaviour, attendance, and performance. Teachers, administrators, and parents can use the system to track pupils' development and spot areas where they can benefit from extra help.

Features like grade tracking, attendance tracking, behaviour management, and communication tools are frequently included in academic monitoring systems. Teachers can submit grades and attendance data into the system, and administrators can utilise it to produce reports on student performance and spot trends over time. The system also allows parents to contact with teachers and keep track of their child's progress.

The fact that an academic monitoring system offers real-time information on student performance is one of its main advantages. As a result, teachers can see difficult children early on and offer them extra help before they fall behind. Additionally, it assists administrators in discovering potential areas for curriculum or instruction improvement at the institution.

An further advantage of an academic monitoring system is that it encourages accountability and transparency. In order to keep them updated on their child's progress, parents can always check their child's grades and attendance records. The technology can be used by teachers and administrators to connect with parents more efficiently, giving them regular updates on their children's progress and answering any queries or concerns they may have.

Dashboard

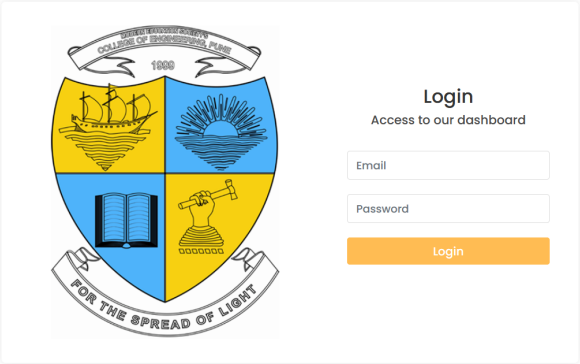


Figure 8.1: Login Page

Figure 8.2: Admin DashBoard

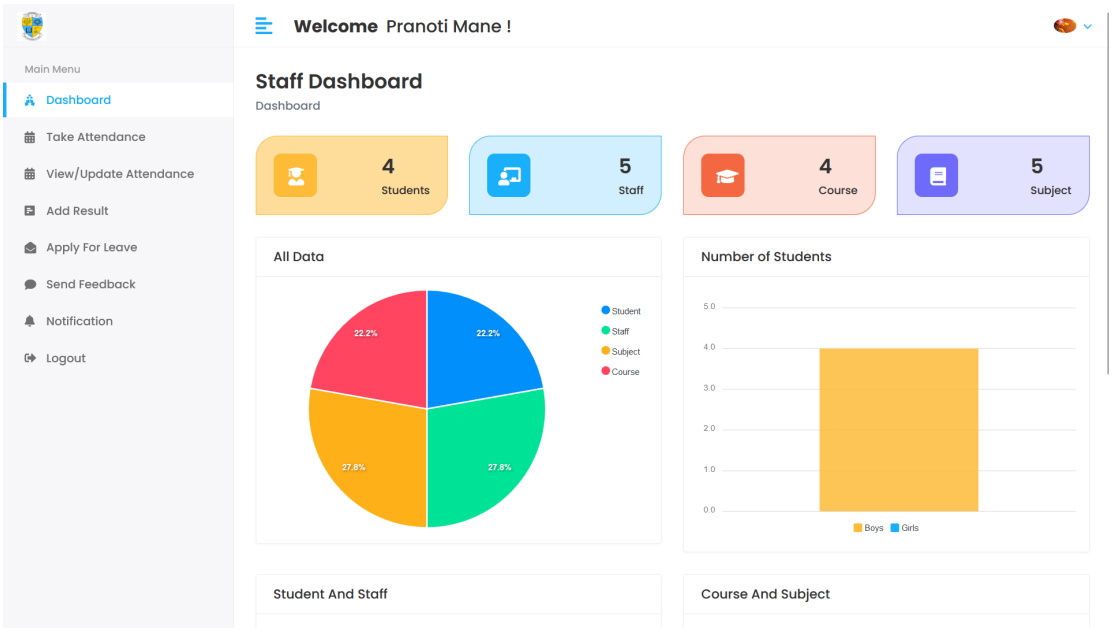


Figure 8.3: Staff DashBoard

Figure 8.4: Student DashBoard

Admin Features

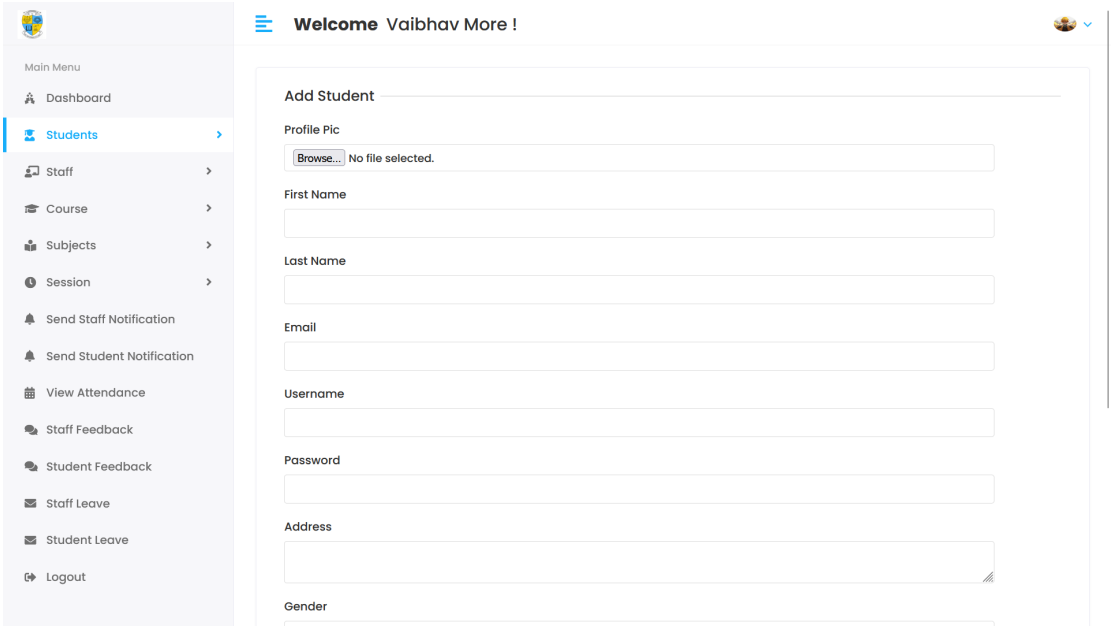


Figure 8.5: Add Student

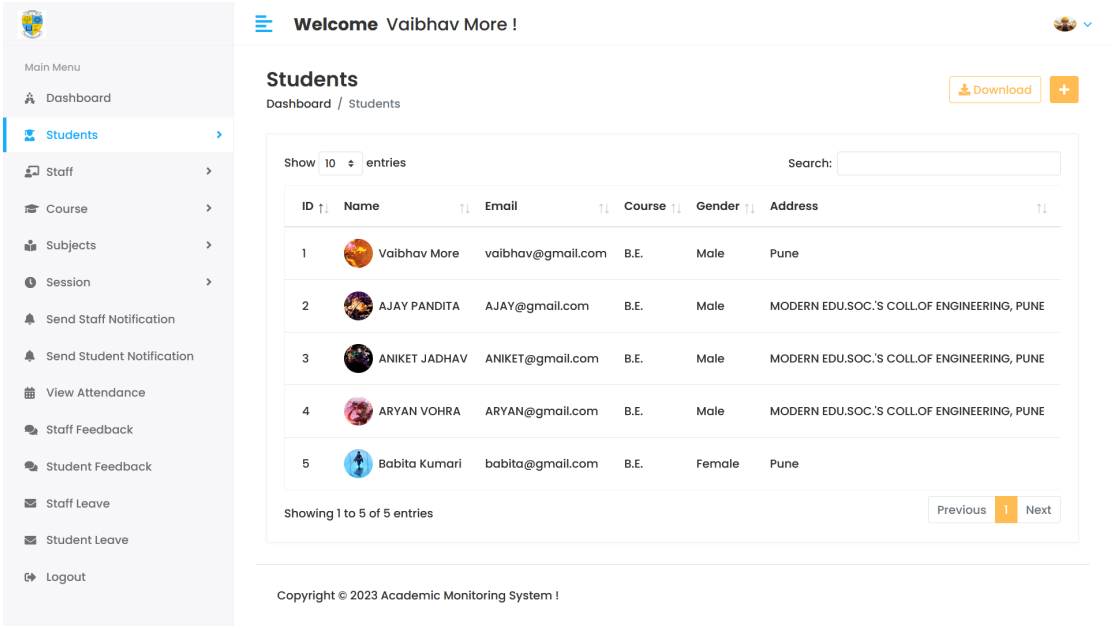


Figure 8.6: View Student

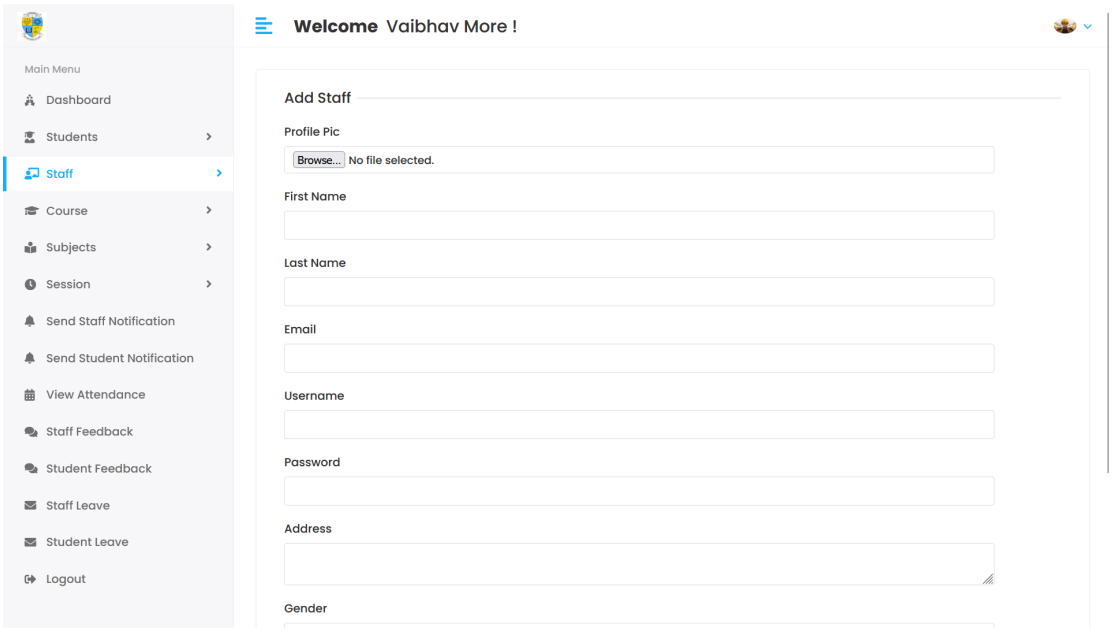


Figure 8.7: Add Staff

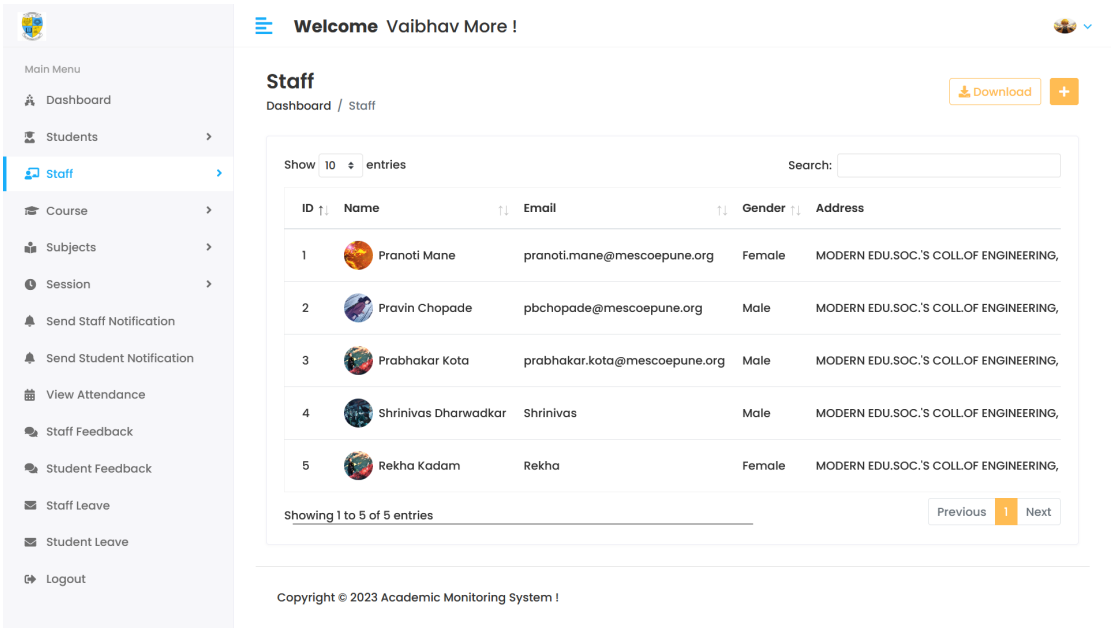


Figure 8.8: View Staff

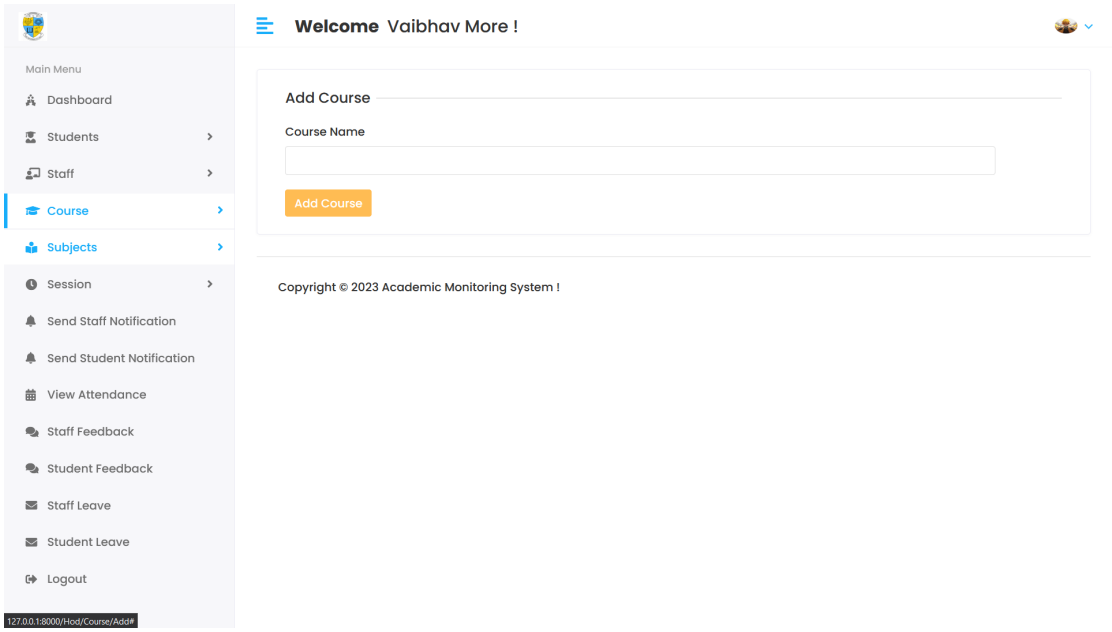


Figure 8.9: Add Course

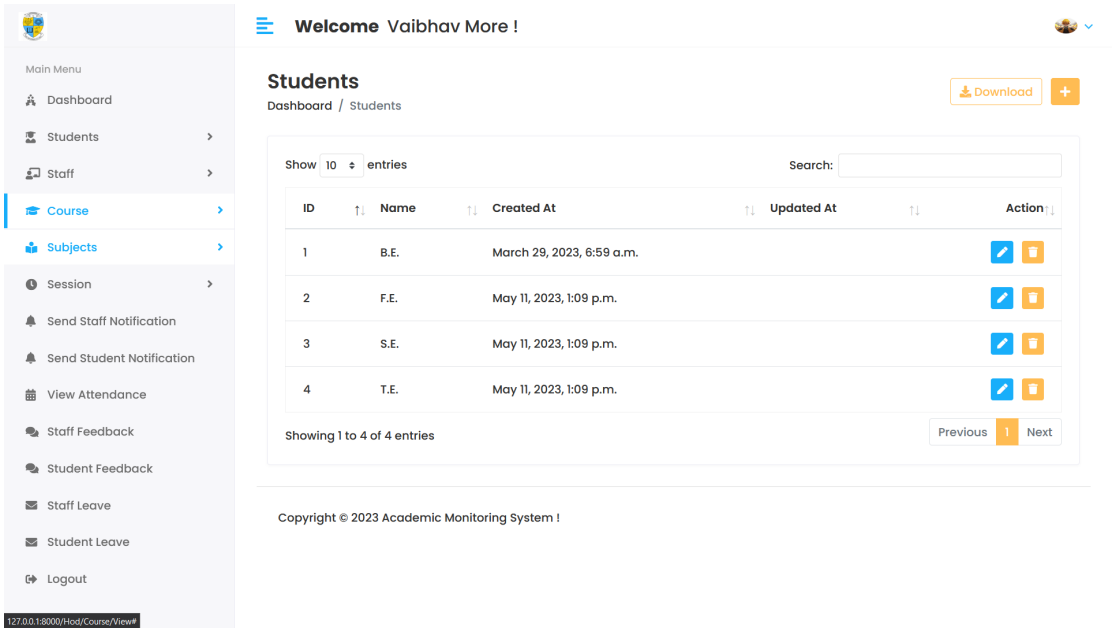


Figure 8.10: View Course

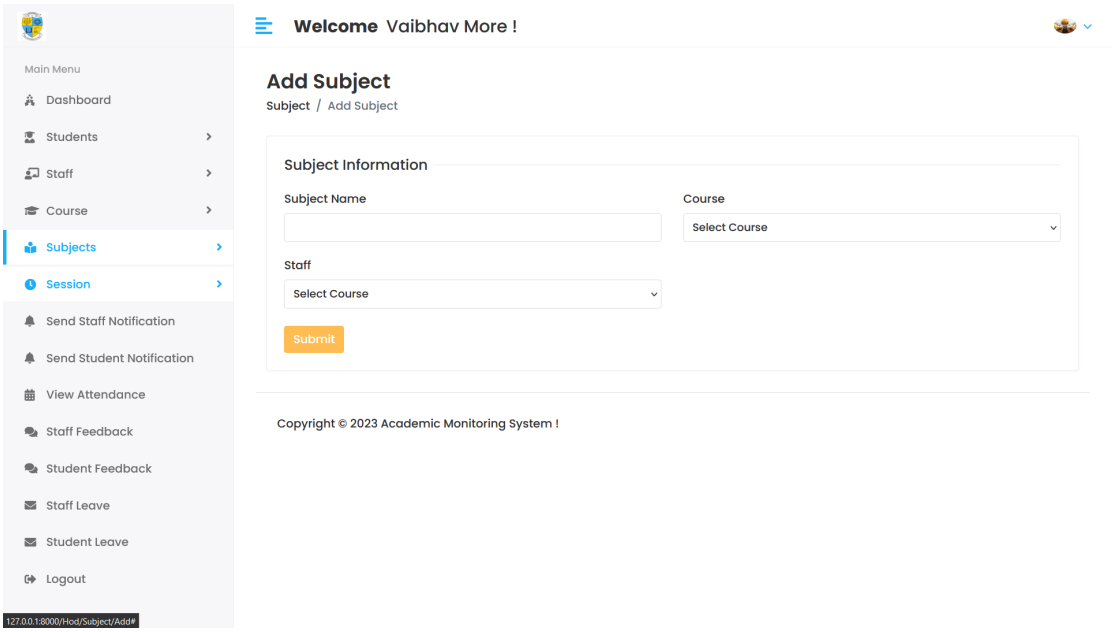


Figure 8.11: Add Subject

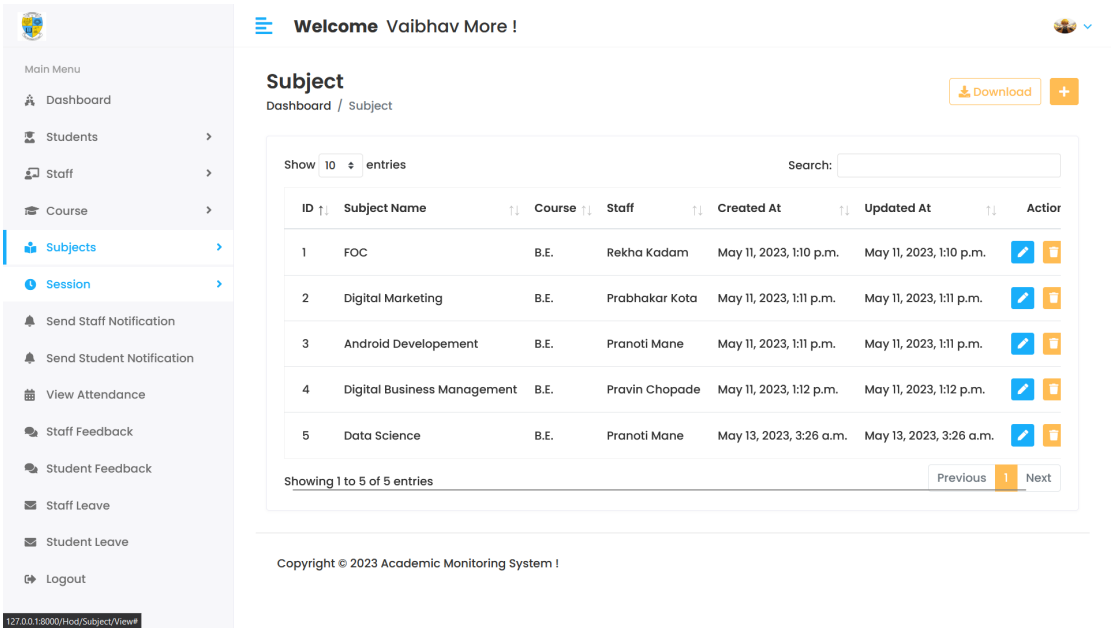


Figure 8.12: View Subject

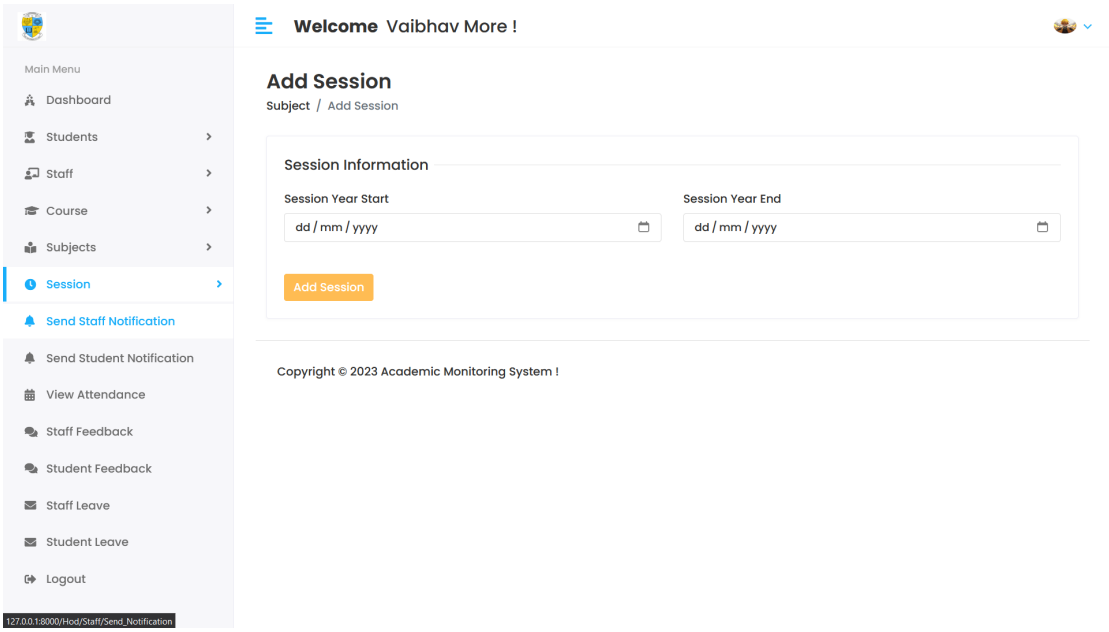


Figure 8.13: Add Session

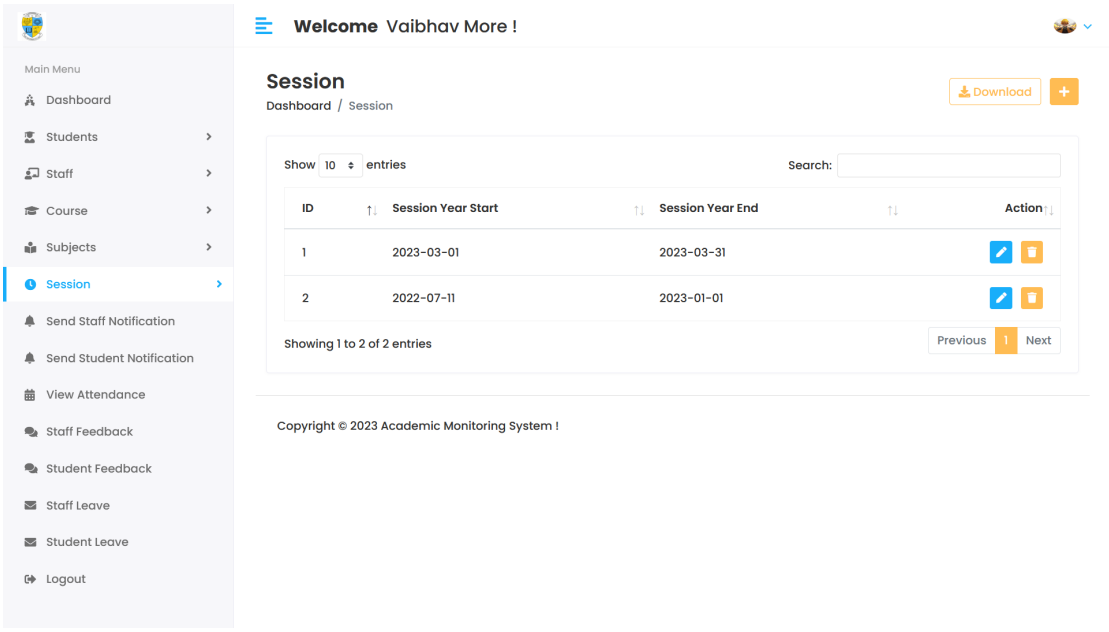


Figure 8.14: View Session

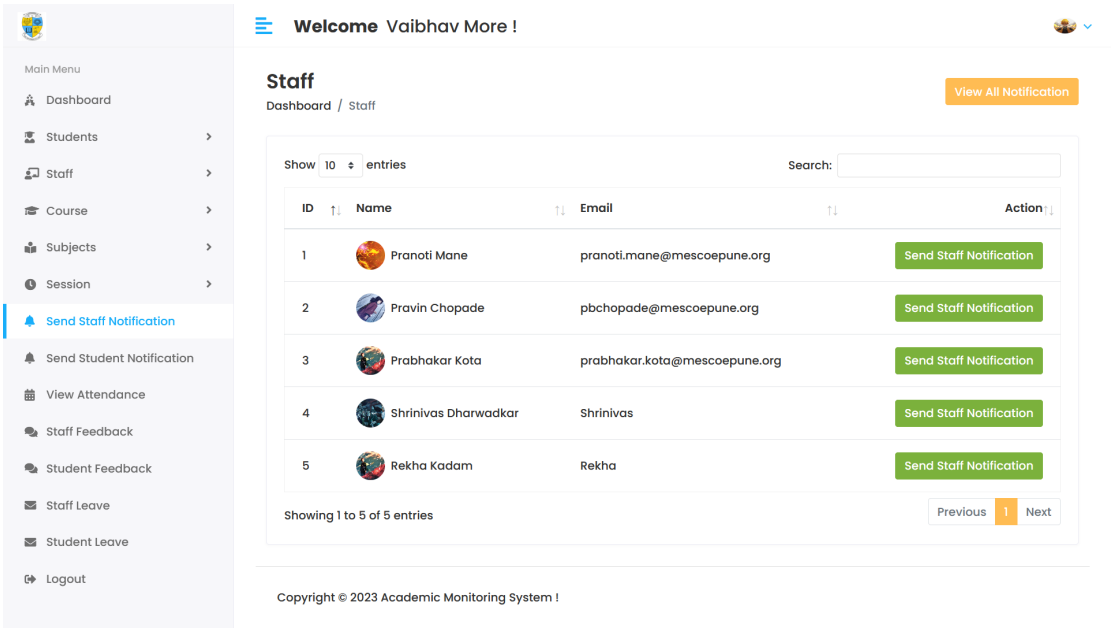


Figure 8.15: Send Staff Notification

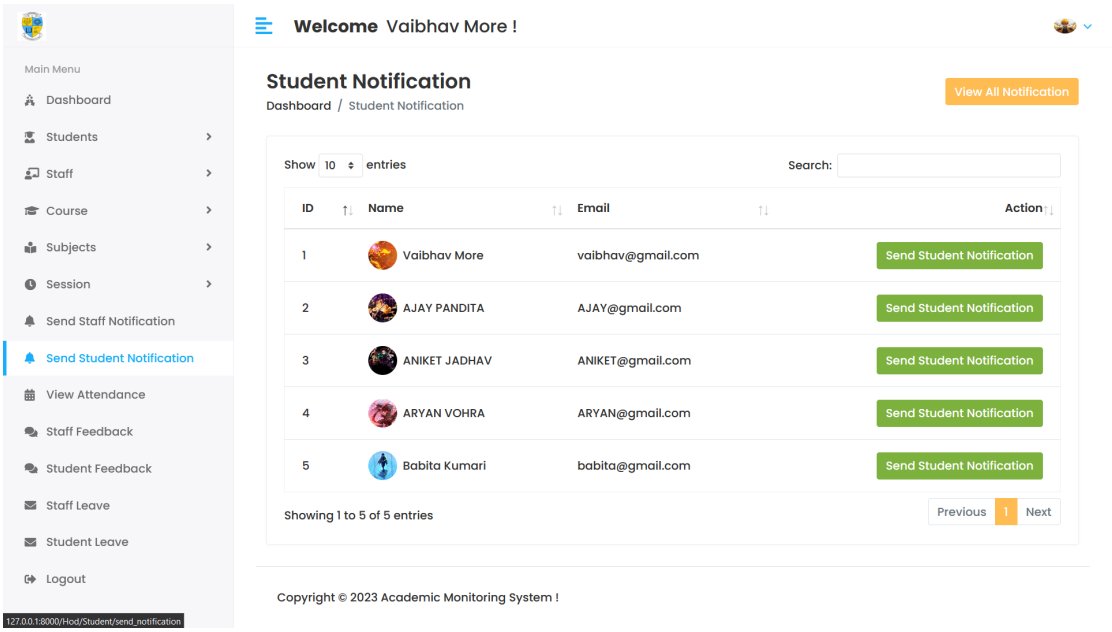


Figure 8.16: Send Student Notification

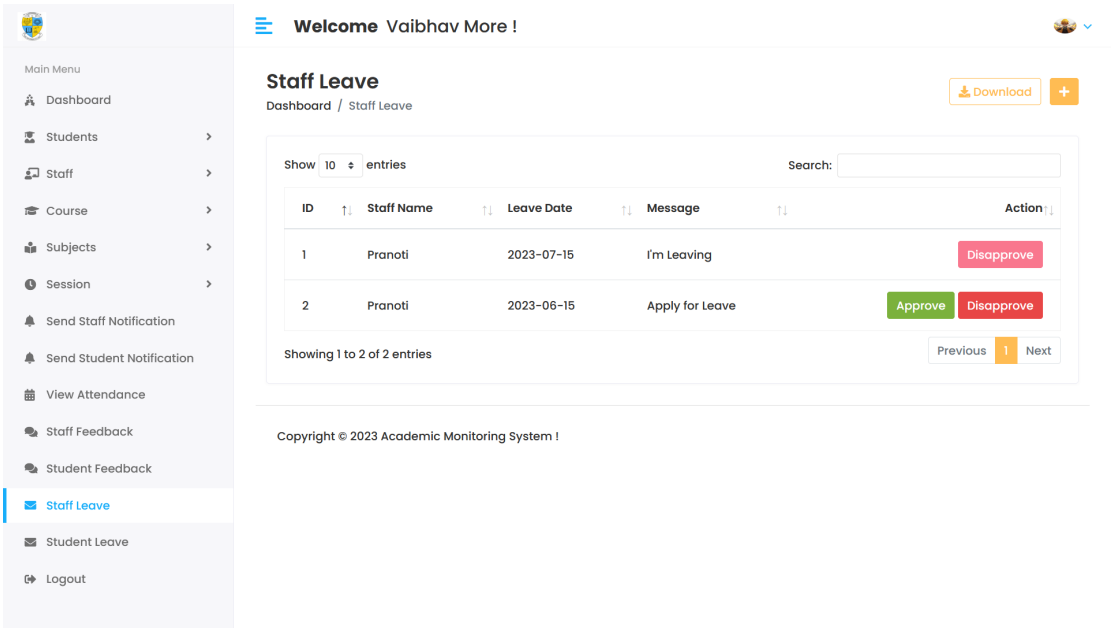


Figure 8.17: Staff Leave

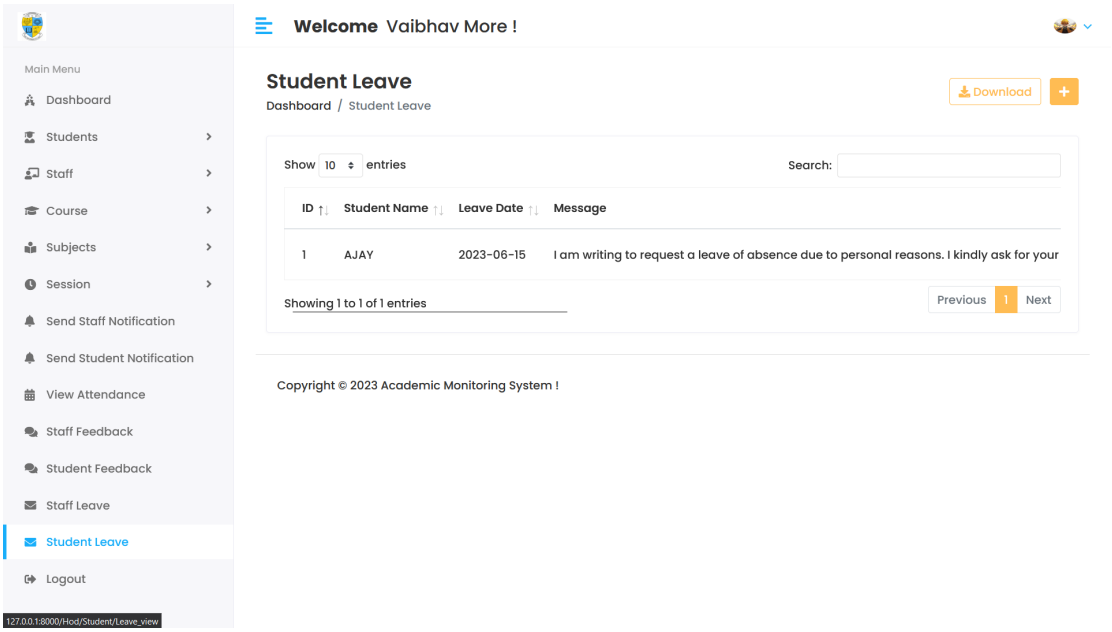


Figure 8.18: Student Leave

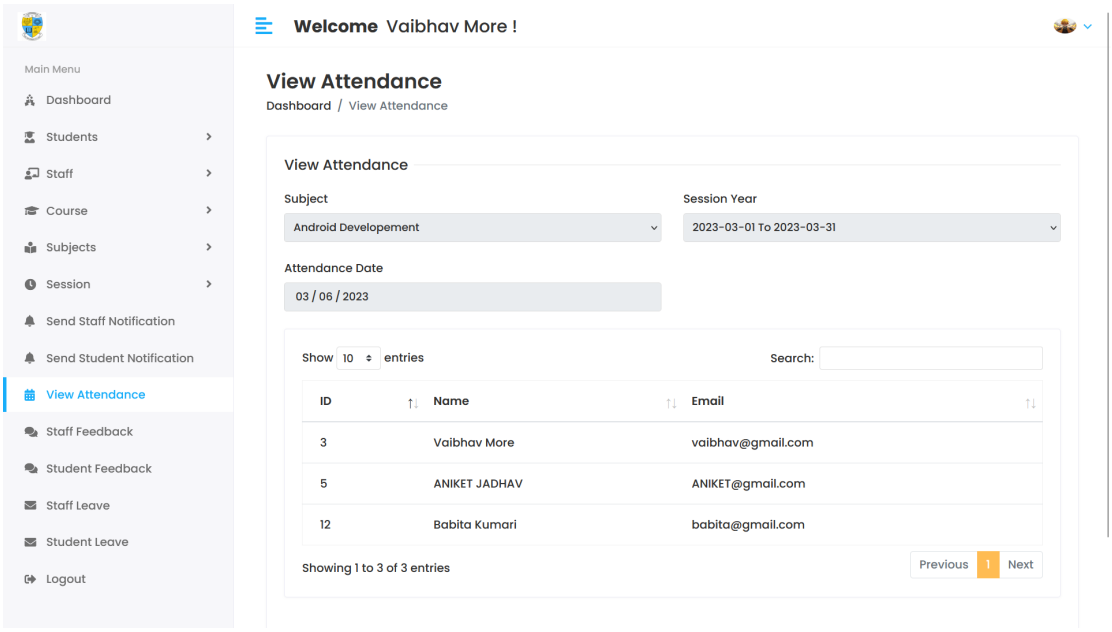


Figure 8.19: Admin View Attendance

Staff Features

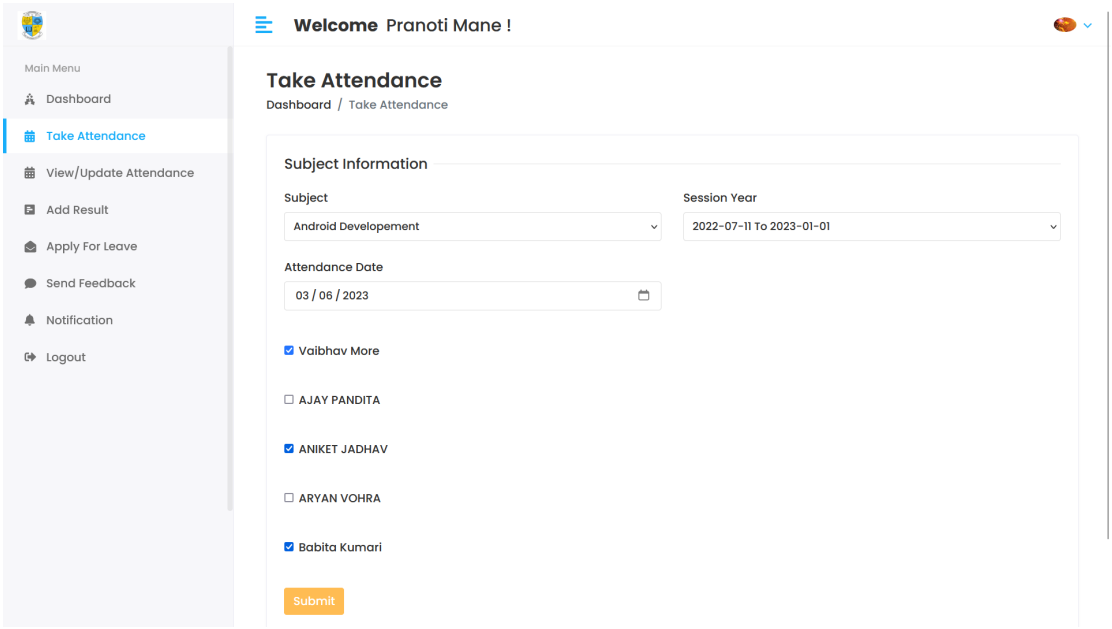


Figure 8.20: Staff Take Attendance

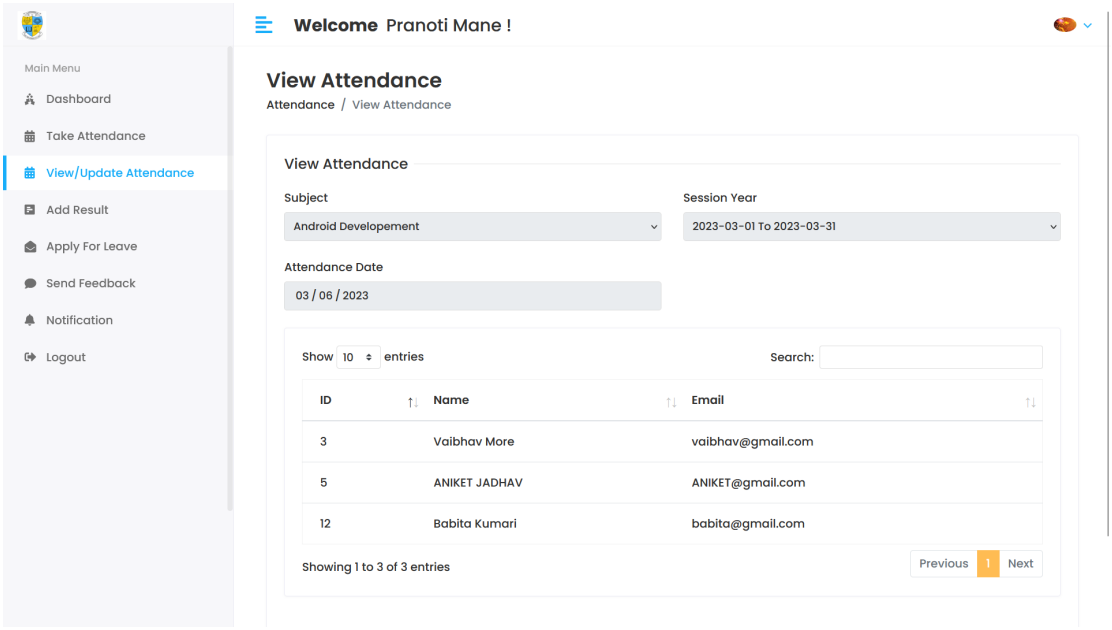


Figure 8.21: Staff View/Update Attendance

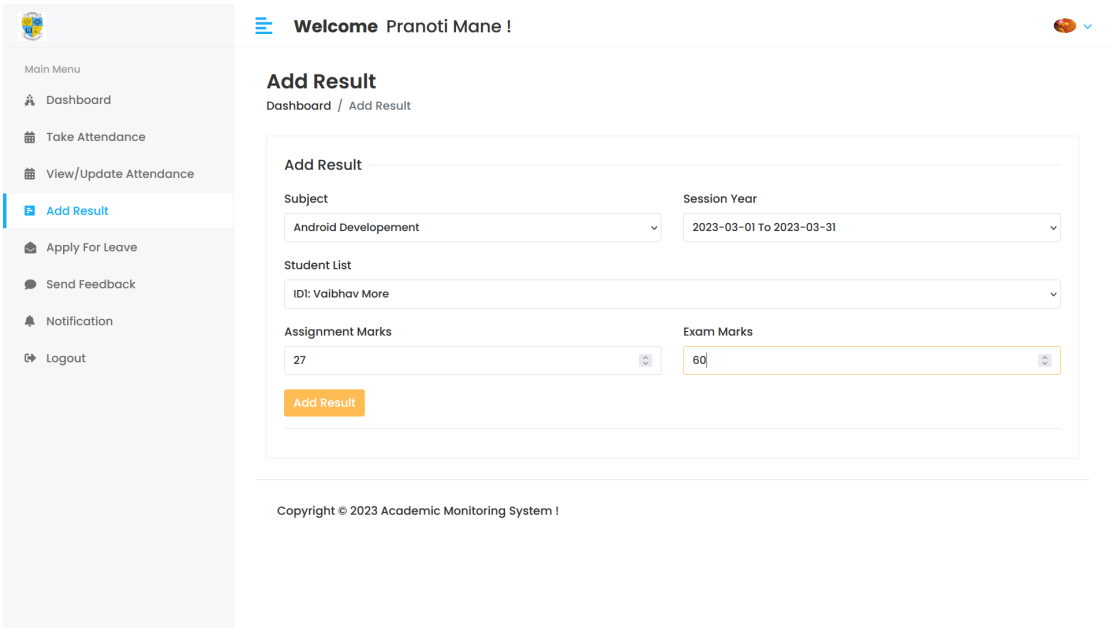


Figure 8.22: Staff Add Result

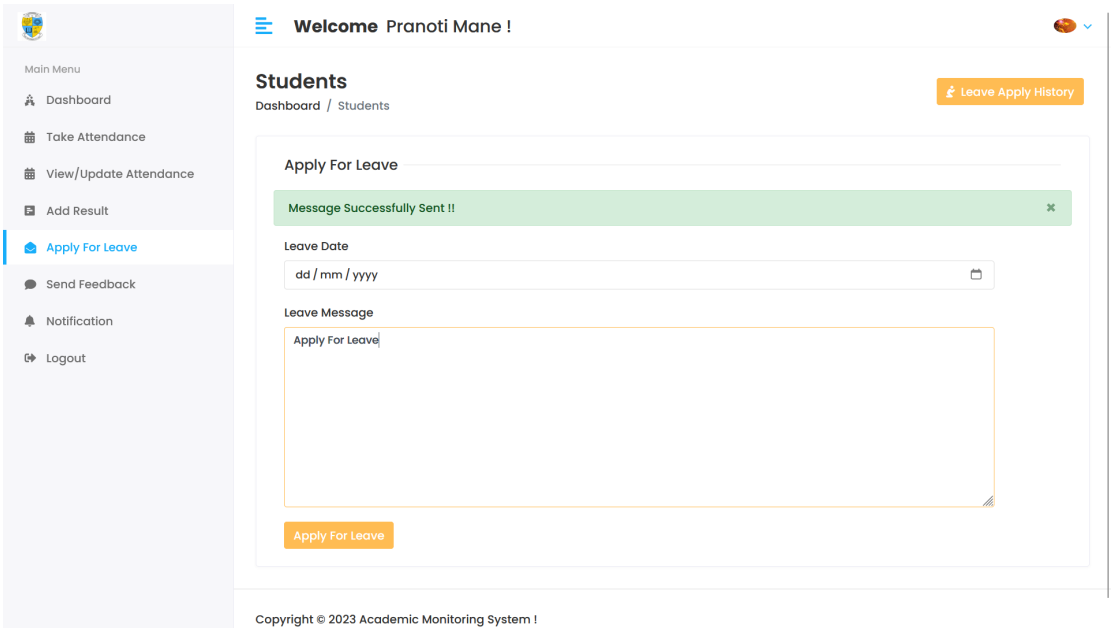


Figure 8.23: Staff Apply For Leave

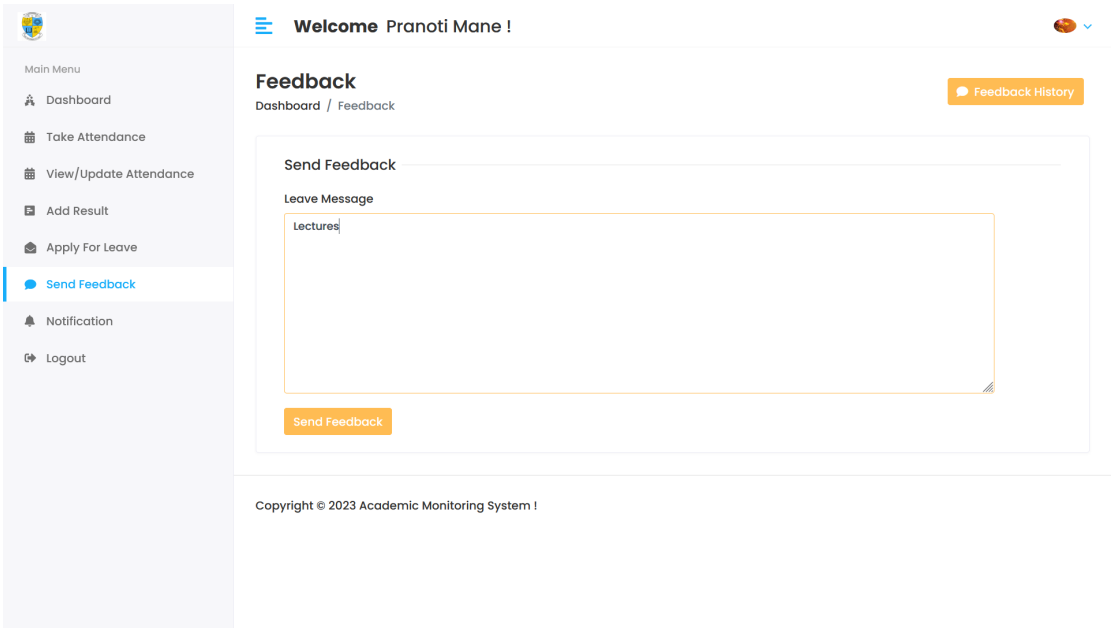


Figure 8.24: Staff Send FeedBack

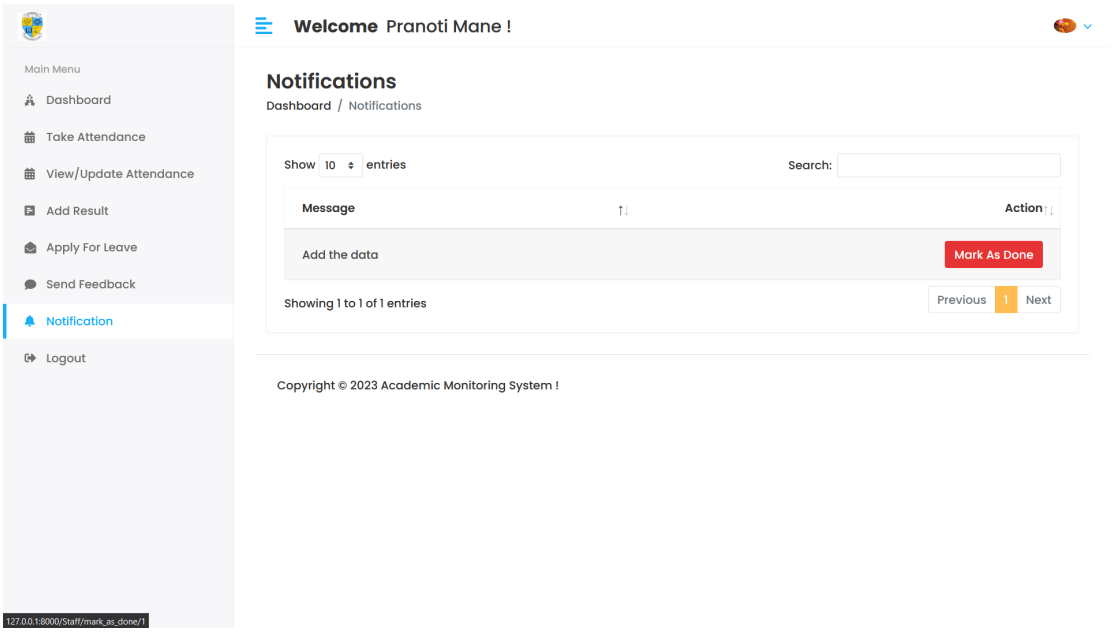


Figure 8.25: Staff Notification

Student Features

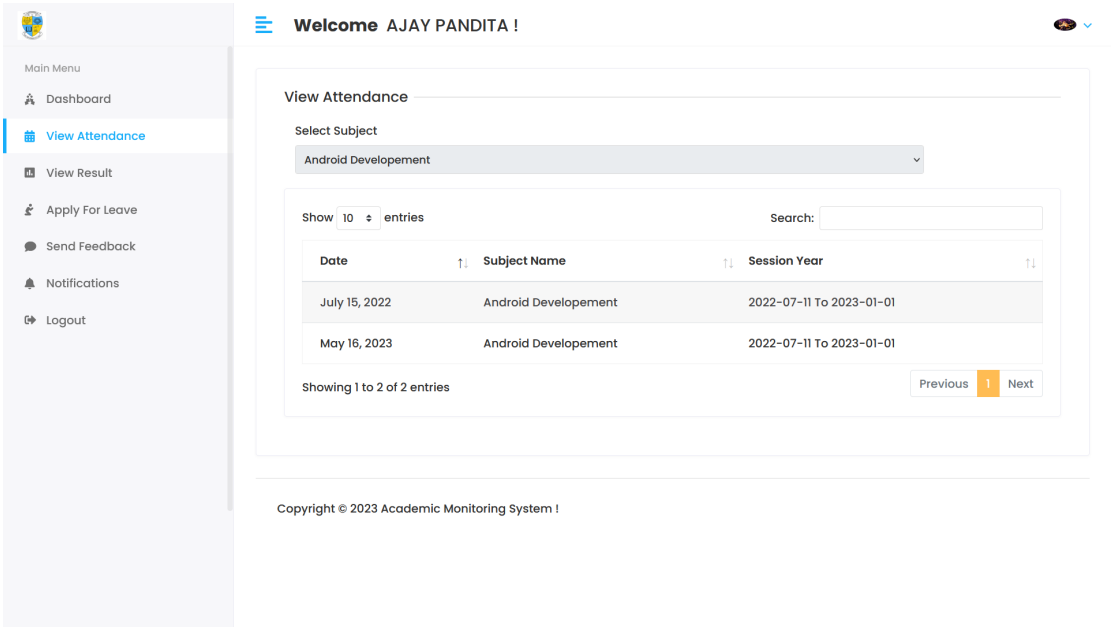


Figure 8.26: Student View Attendance

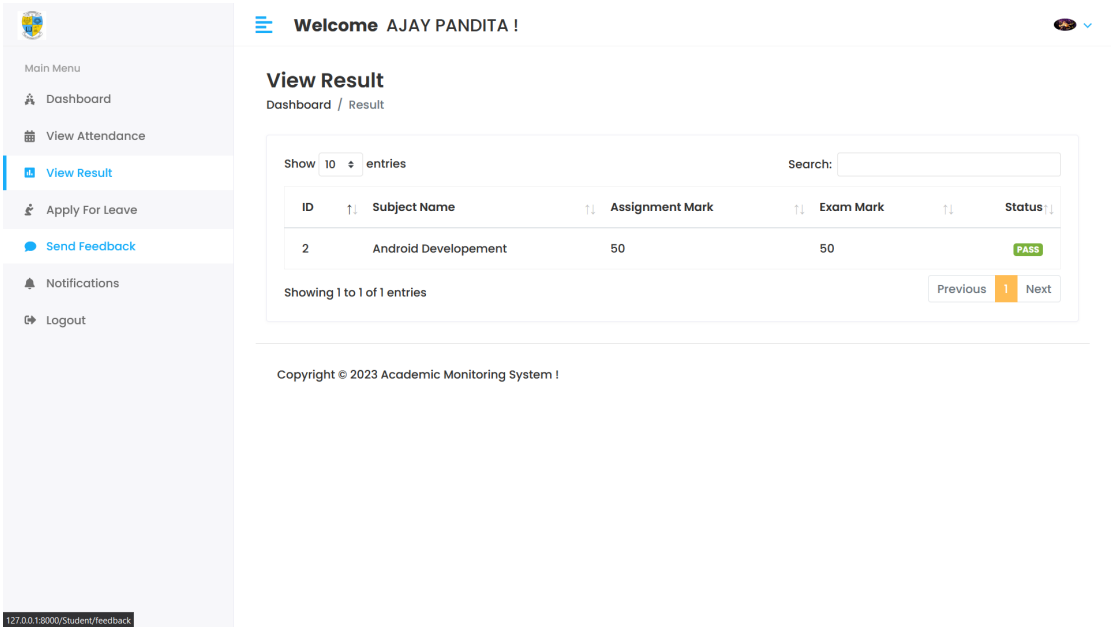


Figure 8.27: Student View Attendance

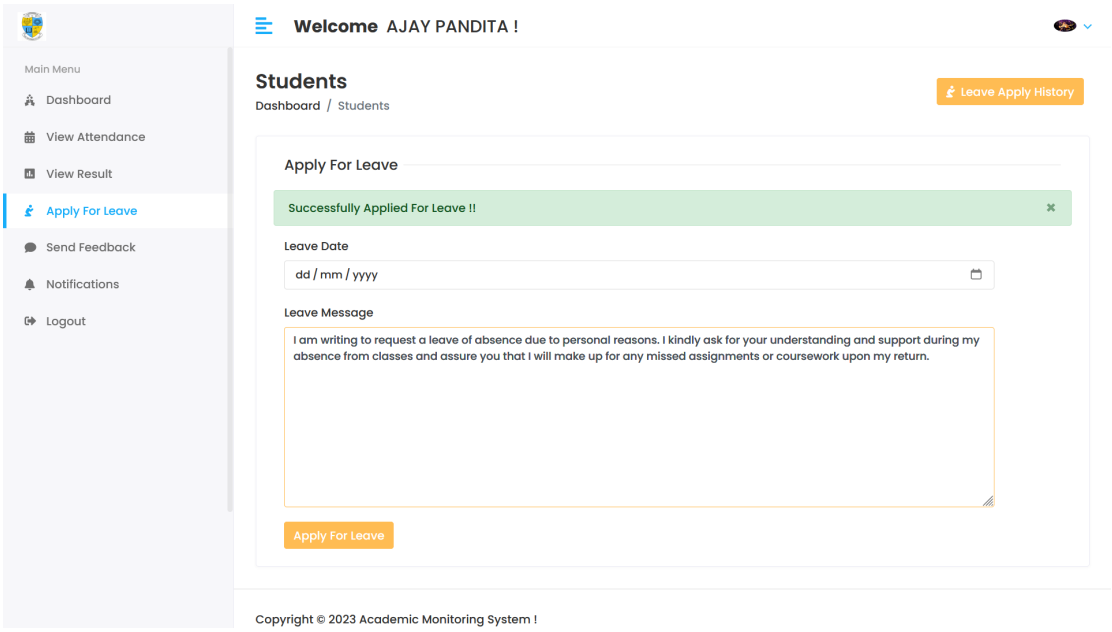


Figure 8.28: Student Apply For Leave

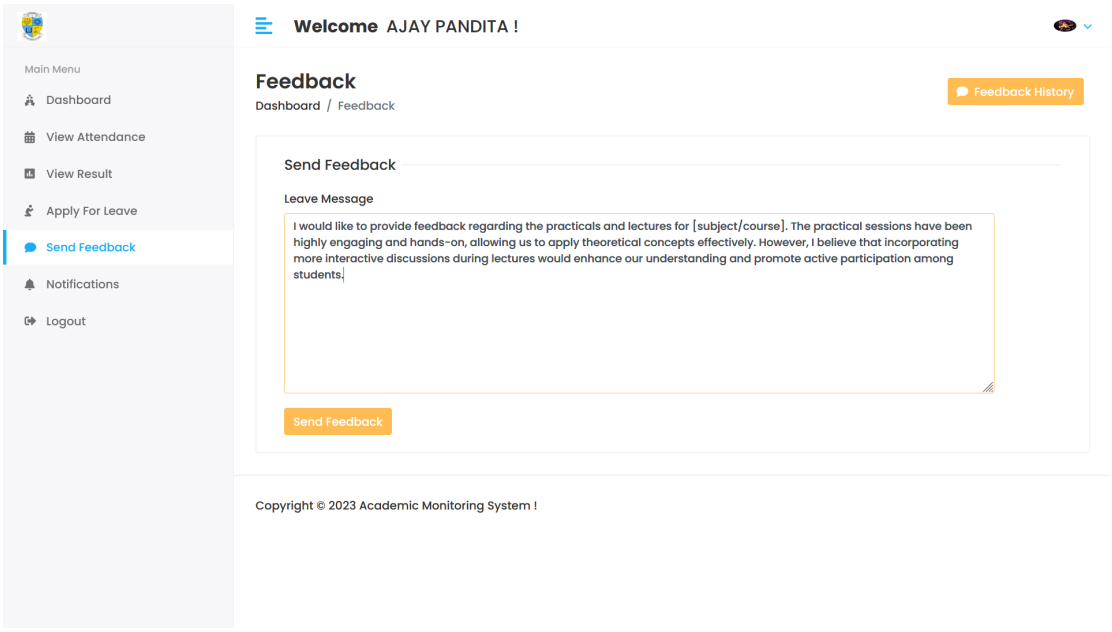


Figure 8.29: Student Send FeedBack

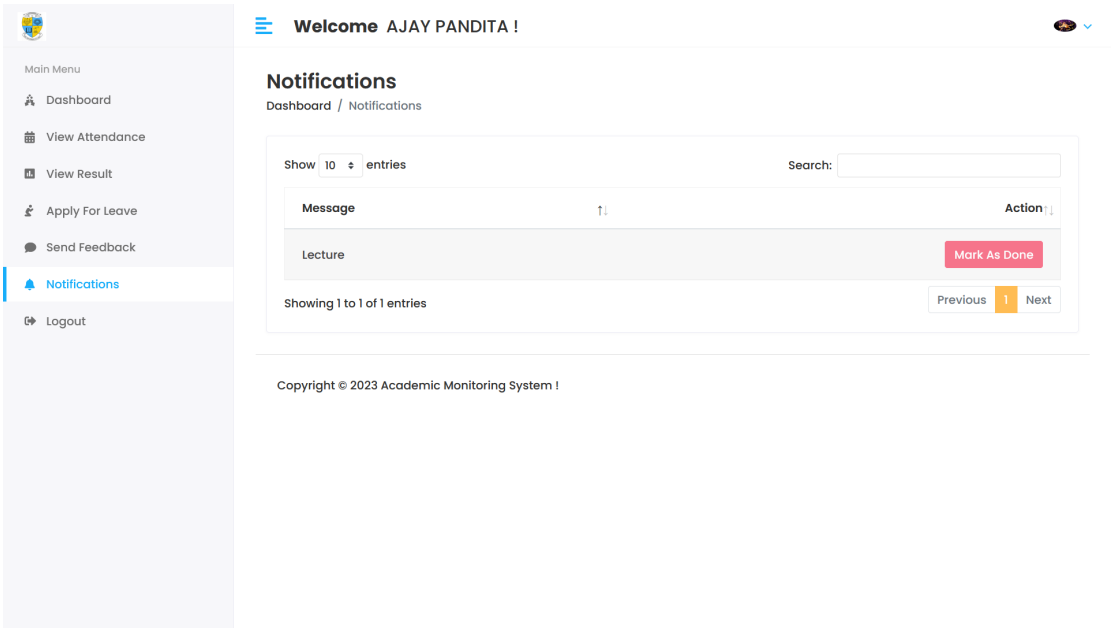


Figure 8.30: Student Notifications

An academic monitoring system's deployment can have a substantial positive impact on educational institutions. Enhanced administrative process efficiency is one of the main outcomes. The technology decreases the amount of manual labour necessary, streamlines workflows, and helps staff members and teachers make better use of their limited time by automating operations like attendance monitoring, grading, and report preparation.

Enhancing data management and accuracy is a significant additional outcome. The digital infrastructure of the system allows for precise recording and management of student data, including attendance logs, grades, and academic progress. As a result, there is no chance of human error, and academic data is guaranteed to be accurate and reliable.

Additionally beneficial to the project are real-time monitoring and reporting capabilities. The system makes it possible for teachers and administrators to get access to the most recent data on student behaviour, performance, and attendance. Effective communication with parents and other stakeholders is made possible by this timely information, which also enables proactive interventions, early issue detection, and communication of problems.

Additionally, the academic monitoring system fosters better interaction and cooperation among members of the educational community. Features like message services and web portals promote smooth communication between educators, students, and parents. This improves communication, makes it possible to provide feedback quickly, and promotes teamwork to support the achievement of students.

The project also results in data-driven decision-making. Informed decisions about student interventions, academic assistance, and resource allocation may be made by educators and administrators thanks to the system's extensive data collection, analysis, and pattern-finding capabilities. The efficacy of educational interventions and methods is increased by this evidence-based approach.

In terms of effectiveness, data accuracy, real-time monitoring, communication, and data-driven decision-making, the adoption of a monitoring system for academic institutions generally produces favourable outcomes. These results assist student performance and the general enhancement of educational procedures.

CHAPTER 9

CONCLUSION

Academic monitoring systems often seek to enhance educational results by tracking and analysing student performance, attendance, behaviour, and other pertinent data. These systems can be put into use on a variety of scales, including the classroom, school, district, and even the national level.

The goals and objectives of the system, the data types to be collected and analysed, the methods of data collection and analysis, the privacy and security concerns of stakeholders, and the feedback mechanisms for stakeholders must all be taken into account when designing an effective academic monitoring system.

Technology utilisation is a vital component of academic monitoring systems. Real-time data collection and analysis can be done using a variety of software programmes and technologies. Learning management systems (LMS), for instance, can monitor student development and offer tailored feedback in accordance with their performance. Similar to this, student data such as attendance records, grades, and behavioural problems can be stored and managed via student information systems (SIS).

The involvement of stakeholders like teachers, parents, administrators, and the actual pupils themselves is another crucial factor. The system's benefits and intended usage in enhancing educational results must be explained to these stakeholders. Additionally, they require access to pertinent information that might help them make decisions, such as reports and data.

CHAPTER 10

APPLICATION

Our project have many real-life applications, some of which are:

- An academic monitoring system is a piece of software that enables educational institutions to track and keep tabs on the behaviour, attendance, and performance of their students. The technique can be used to spot pupils who are having difficulties academically or behaviorally and give them the help they need to succeed.
- A system for academic monitoring has several uses. It can be used to monitor student progress in schools, colleges, and universities. Teachers may simply keep track of attendance, grades, and other performance indicators with the use of an academic monitoring system. Then, using this information, reports can be created to assist administrators in making decisions about curriculum development, teacher preparation, and student support services.
- An academic monitoring system can be used to track student behaviour in addition to academic success. This entails monitoring sanctions, spotting trends of misbehaviour, and, if required, initiating early intervention.
- An academic monitoring system's ability to help teachers spot kids who are at risk of falling behind early on is one of its main advantages. This implies that teachers can step in before a student's behaviour or academic performance become severe issues. Schools can enhance overall results and lower dropout rates by giving challenging students specialised attention and resources.
- The method makes it possible to efficiently and precisely track students' attendance. Teachers can use the digital platform it offers to track attendance, note absences, and create attendance reports. With the help of this programme, manual attendance management is no longer necessary, and attendance data is kept current and conveniently accessible.
- Teachers and administrators can efficiently monitor students' academic progress thanks to the academic monitoring system. It offers a central database where grades, tasks, and tests may be kept track of. Teachers can enter grades, monitor student progress, and produce thorough performance reports. This software enables prompt interventions and focused help for students who might be having academic difficulties.
- The system is also capable of managing and tracking student behaviour and disciplinary records. Teachers are able to keep a record of student behaviour as well as occurrences and disciplinary measures. Using this programme makes it easier to see patterns of misconduct, monitor disciplinary developments, and take the necessary steps to preserve a supportive learning environment.

- The academic monitoring system promotes regular communication between professors, students, and parents. It provides for the compilation of progress reports that provide an overview of students' academic performance, attendance, and behaviour. Teachers can share these reports with parents electronically, enabling an open and transparent dialogue regarding kids' success and areas for growth.
- The system offers reporting and data analysis features. It enables administrators to produce a number of reports, including performance trends, grade distributions, and attendance summaries. These reports aid in finding trends, evaluating the success of educational initiatives, and coming to data-driven conclusions on how to enhance instruction.
- The programme acts as a centralised database for student data. It keeps track of crucial information like contact information, academic records, and personal data. With the help of this application, data management is made easier while also ensuring the security and accessibility of student information.

CHAPTER 11

ADVANTAGES AND LIMITATIONS

Advantages:

- Academic monitoring systems can assist teachers in identifying areas where students' academic performance can be improved, such as those where they are having difficulty or where they can use more effective teaching strategies.
- Increased student engagement: By offering feedback on tasks, presenting chances for self-reflection, and motivating students to participate in the learning process, academic monitoring systems can help students engage more deeply in the academic process.
- Improved teaching strategies: By detecting areas where students are having difficulty and giving them specific feedback and support, academic monitoring systems can assist teachers in improving their teaching strategies.
- Increased transparency: By giving information on the academic performance of students and teachers, academic monitoring systems can aid in enhancing the transparency of the academic process.
- Improved communication: Academic monitoring systems can help to improve communication between teachers and students, by providing regular updates on the academic performance of students and providing feedback on assignments. Communication between teachers and students can be improved thanks to academic monitoring systems, which give regular updates on students' academic performance and give comments on assignments.

Limitations:

- Installing academic monitoring systems might be costly since they need more hardware and software resources and possibly more teacher and administrative training. Implementing academic monitoring systems can be challenging since it calls for a high level of technological know-how and may necessitate the creation of new hardware and software.
- Limited functionality: Academic monitoring solutions could not offer all the features necessary for a thorough examination of the academic process, such as automatic grading and feedback or system integration.
- Limited data: Academic monitoring systems might not provide enough information to give a thorough picture of how well students and teachers are doing academically, which could reduce the system's usefulness.
- Limited analysis: Academic monitoring systems may not offer enough analytic capabilities to pinpoint areas where the academic process needs improvement, which could reduce the system's usefulness.

CHAPTER 12

FUTURE SCOPE

Future applications for a Django-based academic monitoring system include the following: Integration with other systems: To provide a more thorough picture of students' academic performance, the academic monitoring system can be combined with other systems including learning management systems, student information systems, and communication systems. Automated feedback and grading: The system can be set up to automatically grade assignments and give students feedback, which would lighten the load on teachers and boost system effectiveness.

Real-time evaluation: The system can be set up to evaluate students' academic achievement in real-time, giving teachers and students immediate feedback. Student participation: By offering feedback on tasks and opportunities for introspection, the system can be set up to encourage students to take part in the educational process. Course planning and scheduling: The system may be configured to include tools for course planning and scheduling, enabling instructors to efficiently plan and schedule their classes. Analytics and reporting: The system may be set up to give teachers the ability to assess their pupils' academic progress and pinpoint areas in need of improvement. Mobile app: The system may be set up to offer a mobile app that students and teachers could use to access it from their smartphones and other portable devices, increasing both their use and accessibility. Overall, a Django-based academic monitoring system can be used to increase the effectiveness of the academic process, provide a thorough perspective of student academic achievement, and foster student involvement and motivation.

REFERENCES

[1] B. J. Mendonca, G. D'mello, R. D'souza, and J. More, "Automated attendance using android devices," Int. J. Appl. Inf. Syst., vol. 8, no. 6, pp. 21–26, 2015.

[2] H. Sutar, S. Chaudhari, P. Bhopi, and D. Sonavale, "Automated Attendance System," Int. Res. J. Mod. Eng. Technol. Sci., vol. 04, no. 04, 2022

[3] Y. N. S, A. Kumar, and N. R. Kumar, "Location Based Smart Attendance System Using GPS," Ann. Rom. Soc. Cell Biol., vol. 25, no. 2, pp. 4510–4516, 2021, [Online]. Available: http://annalsofrscb.ro

[4] B. Chandramouli, S. A. Kumar, C. V. Lakshmi, G. B. Harish, and P. A. Khan, "Face Recognition Based Attendance System Using Jetson Nano," Int. Res. J. Mod. Eng. Technol. Sci., vol. 3, no. 8, 2021.

[5] D. Prangchumpol, "Face Recognition for Attendance Management System Using Multiple Sensors," in Journal of Physics: Conference Series, 2019, vol. 1335, no. 1, doi: 10.1088/1742-6596/1335/1/012011

[6] M. S. M. Alburaiki, G. M. Johar, R. A. A. Helmi, and M. H. Alkawaz, "Mobile Based Attendance System: Face Recognition and Location Detection using Machine Learning," in 2021 IEEE 12th Control and System Graduate Research Colloquium, ICSGRC 2021 - Proceedings, 2021, pp. 177–182. doi: 10.1109/ICSGRC53186.2021.9515221.

[7] P. S. H. Smitha, "Face Recognition based Attendance Management System.," Int. J. Eng. Research and Technology, vol. 9, no. 05, 2020.

[8] D. Sunaryono, J. Siswantoro, and R. Anggoro, "An Android Based Course Attendance System using Face Recognition," J. King Saud Univ. - Comput. Inf. Sci., vol. 33, no. 3, pp. 304–312, 2021, doi: 10.1016/j.jksuci.2019.01.006.

[9] J. D. W. S. Souza, S. Jothi, and A. Chandrasekar, "Automated Attendance Marking and Management System by Facial Recognition Using Histogram," in 2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS 2019, 2019, doi: 10.1109/ICACCS.2019.8728399.

[10] M. A. J. Hameed, "Android-based Smart Student Attendance System," Int. Res. J. Eng. Technol., vol. 12, pp. 2356–2395, 2017.

[11] F. Susanto, F. Fauziah, and A. Andrianingsih, "Lecturer Attendance System using Face Recognition Application an Android-Based," J. Comput. Networks, Archit. High Perform. Comput., vol. 3, no. 2, pp. 167– 173, 2021, doi: 10.47709/cnahpc.v3i2.981.

[12] Adagale, A., Agrawal, D., Dane, S. (2016). Academic Monitoring System. IJCSN International Journal of Computer Science and Network, 5(2), April 2016. ISSN 2277-5420

[13] V. Somasundaram, M. Kannan, and V. Sriram, "Mobile Based Attendance Management System," Indian J. Sci. Technol., vol. 9, no. 35, pp. 1–4, 2016, doi: 10.17485/ijst/2016/v9i35/101807.

[14] B. K. P. Mohamed and C. V. Raghu, "Fingerprint Attendance System for Classroom Needs," in 2012 Annual IEEE India Conference, INDICON 2012, 2012, doi: 10.1109/INDCON.2012.6420657.

[15] B. Soewito, F. L. Gaol, E. Simanjuntak, and F. E. Gunawan, "Attendance System on Android Smartphone," in ICCEREC 2015 - International Conference on Control, Electronics, Renewable Energy and Communications, 2015, pp. 208–211. doi: 10.1109/ICCEREC.2015.7337046.

[16] S. A. M. Noor, N. Zaini, M. F. A. Latip, and N. Hamzah, "Android-Based Attendance Management System," in 2015 IEEE Conference on Systems, Process and Control (ICSPC), 2015, pp. 118–122.

[17] H. Sutar, S. Chaudhari, P. Bhopi, and D. Sonavale, "Automated Attendance System," Int. Res. J. Mod. Eng. Technol. Sci., vol. 04, no. 04, 2022.

[18] E. Varadharajan, R. Dharani, S. Jeevitha, B. Kavinmathi, and S. Hemalatha, "Automatic Attendance Management System using Face Detection," in 2016 Online international conference on green engineering and technologies, 2016.

[19] A. A. Kumbhar, K. S. Wanjara, D. H. Trivedi, A. U. Khairatkar, and D. Sharma, "Automated Attendance Monitoring System using Android Platform," Int. J. Curr. Eng. Technol., vol. 4, no. 2, pp. 1096–1099, 2014, [Online]. Available: International Journal of Current Engineering and Technology

[20] J. D. W. S. Souza, S. Jothi, and A. Chandrasekar, "Automated Attendance Marking and Management System by Facial Recognition Using Histogram," in 2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS 2019, 2019, doi: 10.1109/ICACCS.2019.8728399.

[21] M. M. Islam, M. K. Hasan, M. M. Billah, and M. M. Uddin, "Development of Smartphone-Based Student Attendance System," in 5th IEEE Region 10 Humanitarian Technology Conference 2017, 2018, vol. 2018, doi: 10.1109/R10-HTC.2017.8288945.