STUDENT MANAGEMENT SYSTEM WITH CHATBOT AI WEBZY

A PROJECT REPORT

Submitted By

U. MASIF AHAMED (922021104013)

S. GUNASEKARAN (922021104302)

P. SAKTHIVEL (922021104305)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SRI VIDYA COLLEGE OF ENGINEERING AND TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this Project report titled **"STUDENT MANAGEMENT SYSTEM WITH CHATBOT AI WEBZY"** is the bonafide work of "**U. MASIF AHEMAD (922021104013), S. GUNASEKARAN (922021104302), P. SAKTHIVEL (922021104305)"** who carried out the Project under my supervision.

SIGNATURE	**SIGNATURE**
Dr.P.KRISHNAVENI	**Dr.P.KRISHNAVENI**
HEAD OF THE DEPARTMENT,	**HEAD OF THE DEPARTMENT,**
Associate Professor,	Associate Professor,
Department of CSE,	Department of CSE,
Sri Vidya College of	Sri Vidya College of
Engineering & Technology,	Engineering & Technology,
Virudhunagar – 626 005.	Virudhunagar – 626 005.

Submitted for the project viva – voice held on ____________________ at Sri Vidya college of Engineering & Technology, Virudhunagar.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost we thank the Almighty for this gracious guidance throughout the project.

We express our sincere and respectful regards to chairman **Er. R. THIUVENKADA RAMANUJA DOSS B.E.,** for providing necessary in carrying for his work.

We endow our thank to Principal, **Dr. T. LOUIE FRANGO BE., ME., PhD.,** of Sri Vidya College of Engineering and Technology, who has given us permission and extending facilities for the successful completion of the project.

At the same time, we wish to record our deep sense of gratitude to **Dr. P. KRISHNAVENI BE., ME., PhD.,** Associate Professor and Head of the Department, Computer Science and Engineering for the constant encouragement and valuable suggestion throughout this project.

We wish to thank **Dr. P. KRISHNAVENI BE., ME., PhD.,** Supervisor and Head of the Department, generously offered valuable suggestions and guidance throughout the project.

Finally, we also thank our parents, friends for their moral support in our project.

ABSTRACT

This project, titled **“STUDENT MANAGEMENT SYSTEM WITH CHATBOT AI WEBZY”**, aims to create an efficient and automated system for managing student data with the integration of an AI-powered chatbot. The system is designed to streamline various administrative tasks related to student information management, such as attendance tracking, grades, and personal details.

The project utilizes a **MySQL database** for storing student information, which is accessed through a **web-based interface**. The **Webzy AI Chatbot** interacts with users through both **text and voice inputs**, providing assistance and answering queries related to the student management system. The chatbot uses **Gemini API** to generate responses, with fallback logic to handle untrained questions.

The system is developed using **Python, Django**, and **HTML/CSS/JavaScript** for the frontend. The integration of **pyttsx3** allows Webzy to respond with a **male voice**, making it a user-friendly system, particularly for visually impaired users.

This system reduces administrative workload, increases efficiency, and improves user experience with the help of AI-based interaction. The successful implementation of the Student Management System demonstrates the potential of AI chatbots in educational and administrative settings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER – 1

INTRODUCTION

1.1 Student Management System

In today's digital era, educational institutions are increasingly adopting technology to manage academic and administrative tasks more efficiently. A Student Management System (SMS) is one such solution, designed to automate and centralize the management of student information throughout their academic lifecycle. From the moment a student enrolls to the time they graduate, an SMS handles all essential processes including admissions, course registration, attendance tracking, performance evaluation, fee payments, timetable scheduling, and communication.

The traditional approach of managing student data through registers, spreadsheets, or disconnected applications often results in data inconsistency, inefficiency, and excessive workload for staff. An integrated SMS overcomes these challenges by storing all student-related information in a secure, centralized database accessible through role-based logins for admins, teachers, and students. This leads to better data accuracy, quick information retrieval, and streamlined academic operations.

Modern SMS platforms also support report generation, real-time notifications, and analytics to help management make data-driven decisions. They improve transparency by allowing students and parents to view academic progress, attendance, and announcements online. However, most existing systems are designed with fixed interfaces and require users to manually navigate through menus and forms to access information, which can be time-consuming and confusing for non-technical users.

With evolving expectations in digital education, there is a demand for systems that are not only functional but also intuitive, responsive, and interactive. This is where the integration of smart technologies like chatbots can significantly enhance the usability and effectiveness of SMS platforms.

1.2 Chatbot AI – WebZy

To meet the growing need for intelligent and responsive student support within academic platforms, the project introduces Webzy, an AI-powered chatbot designed specifically for integration into the Student Management System. Webzy serves as a virtual assistant that interacts with users in natural language, offering real-time responses to common queries and guiding them through various system features without manual browsing.

Webzy leverages Natural Language Processing (NLP) techniques to understand and respond to user queries in conversational form. For example, students can ask questions like "Show my attendance," "What are my internal marks?" or "When is the next exam?" and receive immediate, accurate replies. This simplifies the user experience, especially for those unfamiliar with navigating complex software interfaces.

One of Webzy's key advantages is its 24/7 availability. Unlike human staff, the chatbot can respond to queries at any time, reducing dependency on office hours and administrative personnel. It not only answers academic questions but can also assist with procedural guidance, such as how to apply for leave, register for courses, or check fee due dates. For institutions, this means reduced workload on staff and improved service to students.

Webzy is also customizable and scalable. Admins can easily update its knowledge base, train it with new questions, and configure its responses based on evolving academic policies. Its backend integration ensures that the chatbot delivers real-time, student-specific information pulled directly from the SMS database.

In summary, Webzy transforms the traditional Student Management System into a more interactive, accessible, and intelligent platform, aligning with the expectations of modern educational environments. It enhances student engagement, improves system usability, and supports institutions in delivering smarter digital services.

CHAPTER – 2

LITERATURE SURVEY

2.1 General Introduction

A literature survey is a critical and comprehensive summary of previous research on a particular topic. It serves as the foundation for any academic research by providing insights into existing work, identifying research gaps, and supporting the justification for new investigations. Conducting a literature survey allows researchers to build upon the knowledge established by others, avoid duplication of efforts, and refine their research questions.

The main objective of a literature survey is to gather relevant information from scholarly sources such as journal articles, books, conference papers, and theses. It involves evaluating the methodologies, findings, and conclusions of previous studies to understand the current state of research in the chosen field. This process also highlights different approaches and perspectives, enabling researchers to compare and contrast various theories and results.

A well-structured literature survey not only demonstrates the researcher's understanding of the subject but also helps in developing a conceptual framework for the study. It identifies trends, patterns, and limitations in earlier works, thereby guiding the selection of suitable research methods and objectives. Furthermore, it establishes the significance of the study by situating it within the broader academic discourse.

In summary, a literature survey is a crucial step in the research process. It contributes to academic rigor by ensuring that the new research is relevant, original, and informed by a strong understanding of past work. By analyzing and synthesizing existing literature, researchers can confidently proceed with their investigations and make meaningful contributions to their field of study.

I have researched this project across various institutes and universities. Some of the colleges that have implemented a Student Management System are listed below.

Sathyabama University

This project focuses on developing an SMS for Sathyabama University using HTML, CSS, and JavaScript for the frontend and Node.js for the backend. The system will include:

- User Management – Admin, staff, and student portals.
- Automated Attendance – QR code-based or manual marking.
- Academic Performance Tracking – Secure grade and report management.
- Real-time Communication – Messaging system for students and faculty.
- Database Management – Secure student records using MongoDB or MySQL.

Arka Jain University

This project is designed for Arka Jain University, utilizing HTML, CSS, and JavaScript for the frontend and Python with Django for the backend. The system will include:

- Role-Based Access – Separate portals for students, faculty, and administrators.
- Online Attendance Management – Automated attendance using RFID-based or manual entry.
- Course and Exam Management – Faculty can upload course materials, assignments, and exam schedules.
- Student Progress Tracking – Performance analytics with graphical reports.
- Secure Database – Student records managed using PostgreSQL or MySQL.

2.2 Student Management System

2.2.1 Introduction

A Student Management System (SMS) is a software application designed to manage and streamline the academic, administrative, and financial operations of educational institutions. With the rapid growth of educational technology, various SMS solutions have been developed to enhance the efficiency of student data management, communication, and academic progress tracking.

2.2.2 Systems Overview

Numerous SMS solutions have been implemented across schools, colleges, and universities. Some notable systems include Fedena, Blackboard, Moodle, and PowerSchool. These platforms offer features such as student registration, attendance tracking, grade management, timetable scheduling, and communication tools.

For example, Fedena is an open-source ERP system that supports modules like admission, exams, and HR management. Moodle, although primarily a learning management system, incorporates many SMS features like course enrollment and grade tracking. Despite their capabilities, these systems often require customization to meet the specific needs of individual institutions.

2.2.3 Technologies Used

Most modern SMS platforms are built using web technologies like HTML, CSS, JavaScript, PHP, Python (Django), and Java. Databases such as MySQL, PostgreSQL, and MongoDB are commonly used for data storage. Cloud computing and mobile app integration are also becoming increasingly important for accessibility and scalability.

2.2.4 Limitations of Systems

- While existing systems offer many features, they also have some limitations:
- High cost of proprietary software
- Limited customization for specific institutional needs

- Complex user interfaces for non-technical users
- Inadequate support for real-time analytics and AI-based automation

These limitations highlight the need for customized and user-friendly systems that can be tailored to specific institutional workflows.

2.2.5 Research Gap

Though many SMS applications are available, most are either too generic or too complex for small and medium-sized institutions. There is a lack of simple, efficient, and cost-effective solutions that focus on core student-related functionalities while offering room for modular expansion.

2.3 Chatbot

2.3.1 Introduction

A Student Management System (SMS) is designed to simplify and automate the academic and administrative processes of educational institutions. With increasing digitization in education, SMS solutions have become essential for maintaining student records, monitoring academic progress, and improving communication between students, staff, and administration. To further enhance user interaction, the integration of an AI-based chatbot like Webzy offers a more dynamic and accessible support system.

2.3.2 Technologies Used

Most SMS platforms are developed using web development frameworks such as HTML, CSS, JavaScript, and server-side languages like PHP, Python (Django), and Java. Databases like MySQL and PostgreSQL are used for data storage. For chatbot development, frameworks such as Dialogflow, Rasa, or custom-built NLP models using Python are commonly used to create intelligent, interactive assistants.

2.3.3 Role of Chatbots

Chatbots have emerged as useful tools in the education sector. Studies show that chatbots can handle frequently asked questions, guide students through procedures (like admission or exam schedules), and offer instant support 24/7. Notable educational bots include Jill Watson (Georgia Tech) and chatbots in platforms like Coursera and edX, which help improve user engagement and reduce human workload.

2.3.4 Introducing "Webzy" – A Custom Chatbot for SMS

Webzy is a chatbot designed specifically for this SMS project to enhance communication and user assistance. Unlike general-purpose bots, Webzy focuses on answering queries related to student attendance, internal marks, class schedules, and exam updates. Its integration within the SMS enables students to quickly retrieve data without manually browsing through menus.

2.3.5 Research Gap

There is a clear need for lightweight, interactive, and institution-specific SMS platforms that include AI features like chatbots. Webzy addresses this gap by improving user experience through natural language interactions, reducing dependency on manual navigation, and helping institutions save time and resources.

CHAPTER – 3

EXISTING SYSTEM

3.1 General Introduction

In many educational institutions, the management of student-related data and academic processes is still handled through manual or semi-digital systems. These traditional methods often involve paperwork, spreadsheets, and basic software that are time-consuming, error-prone, and inefficient. Even in institutions that use software solutions, the systems are frequently outdated or lack integration between different modules such as attendance, exam records, and student communication.

Several commercial Student Management Systems (SMS) are currently available in the market, including platforms like Fedena, PowerSchool, Blackboard, and Moodle. These systems provide a range of features such as student registration, attendance tracking, grade management, timetable creation, and parent-teacher communication. However, these platforms are often designed for large institutions, making them expensive, complex, and difficult to customize for smaller colleges or schools.

Another limitation of existing systems is their reliance on static user interfaces. Students and staff must navigate multiple menus and forms to access information or perform tasks, which can be confusing for non-technical users. These systems also typically lack intelligent support mechanisms, such as real-time help or chat-based guidance, leading to increased dependency on administrative staff.

Moreover, most existing SMS platforms do not support mobile-friendly interfaces or chatbot integration, which are becoming increasingly important in today's fast-paced, mobile-first world. Without real-time assistance or smart features, students may miss important updates, deadlines, or notifications, leading to communication gaps and administrative delays.

In summary, while the existing systems offer many useful features, they often fall short in terms of flexibility, ease of use, cost-effectiveness, and intelligent support. These gaps highlight the need for a simplified, customizable, and interactive Student

Management System—enhanced with tools like a chatbot—to meet the evolving needs of educational institutions.

3.2 Student Management System with Chatbot AI – Webzy

To overcome the limitations of traditional student management systems, the proposed Student Management System (SMS) with Chatbot "Webzy" introduces a smart, interactive, and user-friendly platform that combines core administrative features with the power of conversational AI. This system is designed specifically for educational institutions looking for a simple yet efficient solution to manage student data and provide instant support through automation.

3.2.1 Overview of the System

The system includes all essential modules required by an academic institution, such as:

- Student Admission & Profile Management
- Attendance Tracking
- Internal and External Marks Entry
- Timetable and Exam Scheduling
- Staff and Student Login Portals
- Notifications and Announcements

In addition to these, the system integrates Webzy, an AI-powered chatbot designed to enhance user interaction. Webzy acts as a virtual assistant for students and staff, making the platform more dynamic and accessible.

3.2.2 Key Features of Webzy

- 24/7 Query Support: Webzy can answer frequently asked questions related to attendance, internal marks, upcoming exams, and holidays.
- Quick Data Access: Students can ask Webzy things like "What is my attendance percentage?" or "Show my internal marks" and get instant responses.
- User-Friendly Interface: Instead of navigating through complex menus, users can simply chat with Webzy to find what they need.

- Admin Integration: Admins can configure responses, add question categories, and control Webzy’s behavior through a backend dashboard.

3.2.3 Advantages Over Systems

- **Interactive Communication** → Unlike traditional SMS platforms, Webzy provides a conversational layer for real-time interaction.
- **Reduced Workload** → Webzy handles routine queries, reducing the workload for administrative and teaching staff.
- **Mobile & Web Compatibility** → The chatbot is accessible through both web browsers and mobile devices for anytime, anywhere support.
- **Customization** → The system can be tailored to meet the specific needs of different institutions without the complexity of enterprise solutions.

CHAPTER – 4

SYSTEM REQUIREMENTS

4.1 Software Requirements

The Student Management System (SMS) is a web-based application developed using the Django framework in Python. It requires Python 3.10 or higher and Django 4.0+ for server-side development. The frontend uses HTML, CSS, and JavaScript, and optionally Bootstrap for responsive UI design. The backend database is SQLite, which comes pre-configured with Django, simplifying setup and testing. The project runs in a local development environment, so Visual Studio Code or PyCharm is recommended for code editing and debugging. To run the project, pip is used to install required libraries such as Django, speech recognition, pyttsx3, and scikit-learn. For voice-based chatbot functionality, a microphone-enabled device is required, along with an internet connection for Google's speech recognition API. The application is hosted locally via Django’s built-in server using python manage.py run server. Additionally, the project may require Google Chrome or any modern browser for user interface testing. The overall system is platform-independent and can run on Windows, Linux, or macOS, provided the necessary Python environment is set up.

4.1.1 Front end

We utilized HTML, CSS, and JavaScript as the primary front-end technologies to design and develop an interactive, user-friendly interface for the Student Management System. HTML structured the content across various modules, CSS provided consistent styling and responsiveness, and JavaScript added dynamic features for enhanced user interaction and validation.

HTML (HyperText Markup Language) – Structure of the Website → HTML is used to build the layout and content of each page in the SMS portal. Examples include:

- login.html – creates a form for users (Admin, Staff, Student) to log in using username and password.

- dashboard.html – displays the dashboard after login, showing links to modules like "Add Session", "Manage Students", "Apply Leave", etc.
- add_student.html – contains a form to input student details like name, roll number, department, etc.
- manage_attendance.html – includes tables for viewing and updating attendance records.

HTML acts as the **skeleton** of the web pages, displaying headings, forms, tables, buttons, and navigation links.

CSS (Cascading Style Sheets) – Presentation and Styling → CSS styles all the HTML content, making the interface user-friendly and attractive. Key uses:

- Styling the login page with background color, center-aligned form, and hover effects on buttons.
- Giving the dashboard a clean layout with a **side navigation bar**, cards, and dropdown menus.
- Customizing table styles (borders, colors, hover effects) in pages like "Manage Students" or "View Results".
- Responsive design – adjusting the layout for mobile, tablet, and desktop views.

External stylesheets or frameworks like Bootstrap may be linked for a consistent and modern design.

JavaScript – Functionality and Interactivity → JavaScript adds dynamic behaviour to the SMS interface. Common uses include:

• **Form validation** – checking whether all fields are filled before submitting (e.g., when adding a student or applying leave).

• **Real-time notifications** – alerting users on successful actions like "Session Added Successfully".

• **Dropdown filtering** – filtering options (e.g., by department or semester) in result/attendance management pages.

• Enabling/disabling buttons, showing/hiding fields dynamically (e.g., based on user role).

• Using **AJAX** (if implemented) to fetch or update data without reloading the entire page.

4.1.2 Back end

In the backend of our Student Management System (SMS), we have utilized Python as the core programming language due to its readability and robust ecosystem, Django as the web framework to streamline development through its built-in features like ORM, authentication, and admin interface, and SQLite as the database system for its simplicity, lightweight nature, and seamless integration with Django during development.

Python → Python is the core programming language used to develop the backend logic of the SMS. It powers the Django framework and is used to define models (database structure), views (business logic), and forms (data handling). Python's simplicity and readability make it efficient for writing clean, maintainable code. It handles functionalities such as user authentication, session management, leave application processing, attendance recording, and more. Custom Python scripts may also handle chatbot interaction and voice recognition features integrated into the project.

Django (Web Framework) → Django is the main web framework used to build the Student Management System. It follows the MVT (Model-View-Template) architecture. Django handles URL routing, admin interface generation, form validation, user authentication, and security features. Django’s ORM (Object-Relational Mapping) allows seamless interaction with the SQLite database without writing raw SQL. The framework enables easy creation of different user roles (Admin, Staff, Student) with specific access permissions. It also supports the chatbot module integration, email functionalities, and file handling.

SQLite (Database) → SQLite is a lightweight, file-based database used in the SMS project. It stores all the backend data, including user details (students, staff, admin), subject lists, attendance records, session years, exam results, leave applications, and feedback entries. Since SQLite does not require a separate server, it is ideal for development and small-scale deployment. Django's ORM maps Python models directly to SQLite tables, allowing developers to work with Python objects instead of SQL queries. Migrations are used to create and update the tables in the SQLite database based on changes in the Django models.

4.2 Hardware Requirements

To effectively run the Student Management System (SMS) project, a system with moderate hardware specifications is sufficient. The application is developed using Django, which operates well on most standard configurations. A minimum of an Intel Core i3 processor or equivalent, 4 GB of RAM, and 500 GB of hard disk space is recommended for development and testing purposes. For better performance, especially when handling larger datasets or multiple users, an Intel Core i5 or higher processor with 8 GB of RAM is preferred. The system should also have a stable internet connection, a modern web browser, and a microphone if the chatbot's voice functionality is used. These hardware requirements ensure smooth functioning and responsiveness of the SMS application during both development and deployment.

4.2.1 8 GB Ram

An 8 GB RAM (Random Access Memory) configuration plays a crucial role in ensuring the smooth and efficient operation of the Student Management System (SMS) project. With 8 GB of memory, the system can handle multiple processes simultaneously without lag, such as running the Django server, managing the database, processing chatbot inputs, and operating the web browser for frontend interactions. This amount of RAM allows developers and users to experience faster application performance, quicker data retrieval, and smoother multitasking, especially when managing large volumes of student data or accessing multiple modules like attendance, results, and notifications at the same time. It ensures the system remains responsive and avoids crashes during high workloads.

4.2.2 500 GB of hard disk

500 GB of hard disk storage provides sufficient space to install and run all the essential components required for the Student Management System (SMS), such as the operating system, development tools (like Python, Django, and PyCharm), databases (like SQLite or MySQL), and project files. It also allows ample room to store related documents, backups, user data, media files, and logs generated by the system. For

student record management, even with thousands of entries, a significant portion of this space remains available, ensuring smooth operation without frequent concerns about disk space limitations. While 500 GB is more than adequate for development and deployment in a small to medium-scale setup, larger-scale use may benefit from additional or cloud-based storage solutions.

4.2.3 Intel core i3 processor

An Intel Core i3 processor provides a reliable and cost-effective computing platform for developing and running the Student Management System (SMS) project. It delivers adequate processing power to handle essential tasks such as executing the Django backend server, interacting with the SQLite or MySQL database, running the chatbot logic, and rendering the web interface in a browser. While not as powerful as higher-end processors, the Core i3 is sufficient for educational and lightweight applications like the SMS system, ensuring smooth performance during development, testing, and real-time usage without excessive power consumption or heat generation. Its dual-core or quad-core capabilities help in managing multiple tasks concurrently, supporting a responsive user experience.

CHAPTER – 5

SYSTEM IMPLEMENTATION

5.1 General Implementation

The Student Management System (SMS) is implemented as a web-based application to efficiently manage student records, attendance, and academic data. The implementation is divided into frontend, backend, and database development.

1. Frontend Implementation:
    - Technologies Used: HTML, CSS, JavaScript
    - Features:
        - User-friendly dashboards for Admin, Staff, and Students
        - Forms for student registration, attendance, and grade management
        - Responsive design for mobile and desktop access
2. Backend Implementation:
    - Technology Used: Python (Django)
    - Features:
        - User authentication and role-based access
        - API integration for attendance tracking and report generation
        - Secure handling of student records and academic data
3. Database Implementation:
    - Database Used: MySQL
    - Features:

- Tables for storing student details, attendance, grades, and user roles
- Secure data encryption for sensitive information
- Optimized queries for fast data retrieval

5.2 Student Management System

Main Python file →

manage.py is a command-line utility created automatically when a Django project is initialized. It simplifies managing the project by wrapping django-admin commands. Developers use it to run the development server, apply database migrations, create superusers, and more. It ensures the correct project settings are loaded without manually configuring environment variables. This makes it essential for smooth development and testing of Django applications.

Login.html →

A Login HTML page is designed to authenticate users by collecting their login credentials, typically a username or email and a password. It contains a form with input fields for the user to enter this data, and a submit button to send the data to the server for verification. The structure is created using HTML for content and layout, CSS for styling, and optionally JavaScript for validation or interactivity. Once submitted, the form directs the data to a backend endpoint (like a Django view) for processing. A successful login redirects the user to their respective dashboard (admin, staff, or student), while a failed login shows an error message.

Admin Module →

The Admin Module is a central component of the Student Management System that provides authorized administrators with complete control over the platform. This module allows the admin to manage core operations such as adding, updating, or deleting student and staff records, assigning subjects, creating classes, managing attendance, viewing reports, and controlling system settings. Admins also have access to dashboards that display important statistics and updates, helping them make quick decisions. In a Django-based system, this module is often built using Django's admin interface or custom templates with backend integration. Security features like login authentication ensure that only authorized users can access admin functionalities.

Column Name	**Data Type**	**Description**
id	INT (Primary Key, Auto Increment)	Unique identifier for each admin
username	VARCHAR(50)	Unique username used for login
password	VARCHAR(255)	Encrypted password for secure access
full_name	VARCHAR(100)	Full name of the admin
email	VARCHAR(100)	Email address for communication

TABLE 5.2.1 Admin Module

Staff Module →

The Staff Module in a Student Management System allows faculty members to perform academic and administrative tasks related to their assigned subjects and students. Through this module, staff can take daily attendance, enter internal marks, manage assignments, upload study materials, view their class timetables, and send notifications or feedback to students. Each staff member logs in with a unique ID and password, ensuring secure access to their respective data. This module helps reduce paperwork, improve accuracy, and enhance communication between teachers and students. It is typically integrated with the backend so that all data entered is stored and reflected in real-time.

Field Name	Data Type	Description
staff_id	INT (PK)	Unique staff ID
user_id	INT (FK)	Linked to Users table
name	VARCHAR	Staff full name
department	VARCHAR	Teaching department
email	VARCHAR	Email
phone	VARCHAR	Contact number

TABLE 5.2.2 Staff Module

Student Module →

The Student Module is a dedicated section in the Student Management System that allows students to access and manage their academic information. After logging in with their credentials, students can view their personal profile, attendance records, internal and external marks, exam schedules, class timetables, and receive announcements or notifications from staff or admin. They can also download study materials, submit assignments, and interact with the system through features like feedback forms or integrated chatbots (e.g., Webzy). This module ensures transparency, improves engagement, and provides students with quick access to their academic progress in a user-friendly interface.

Field Name	Data Type	Description
student_id	INT (PK)	Unique student ID
user_id	INT (FK)	Linked to Users table
name	VARCHAR	Student's full name
dob	DATE	Date of birth
department	VARCHAR	Branch/Department
semester	INT	Current semester
email	VARCHAR	Email ID
phone	VARCHAR	Contact number

TABLE 5.2.3 Student Module

5.3 Chatbot AI – Webzy

The Chatbot AI – Webzy Module is an intelligent virtual assistant integrated into the Student Management System to provide real-time, automated support to students and staff. Webzy is built using Natural Language Processing (NLP) and AI algorithms to understand user queries and respond conversationally. It is embedded within the SMS interface, allowing users to ask questions such as "What is my attendance?", "Show internal marks", or "When is my next exam?" and get instant, accurate answers.

This module reduces the need for users to manually navigate through menus, improving the system's usability and efficiency. Webzy is available 24/7 and can be trained to answer new queries as needed. Admins can manage its responses and update its knowledge base through the backend. By providing smart, user-friendly interaction, the Webzy module enhances user experience and helps streamline communication within the institution.

Attribute	**Description**
Module Name	Chatbot AI – Webzy
Technology Used	Python, SpeechRecognition, pyttsx3, Scikit-learn (TF-IDF & Cosine Similarity)
Purpose	To provide voice-enabled assistance to users (admin, staff, students)
Inputs	User speech (converted to text), role-based prompts
Outputs	Text-to-speech replies with predefined or smart responses
Functions	Recognize speech, process input, identify role, generate appropriate reply
Integration	Integrated into the SMS web portal; dynamically understands user roles
Login Awareness	Fetches current user session (admin/staff/student) from the database

TABLE 5.3.1 Chatbot AI Webzy Module

5.4 FLOW DIAGRAM

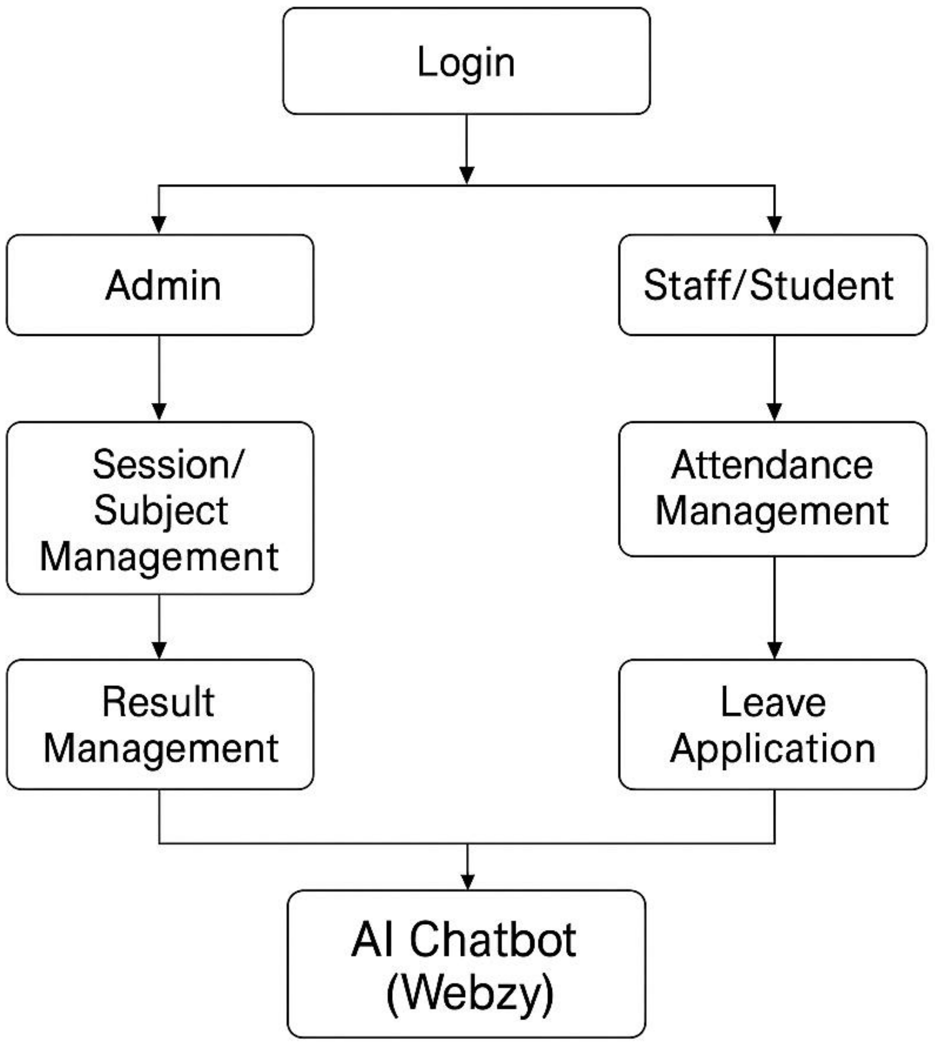


FIGURE 5.4.1 Flow diagram of SMS With Chatbot AI – Webzy

5.5 DATA FLOW DIAGRAM

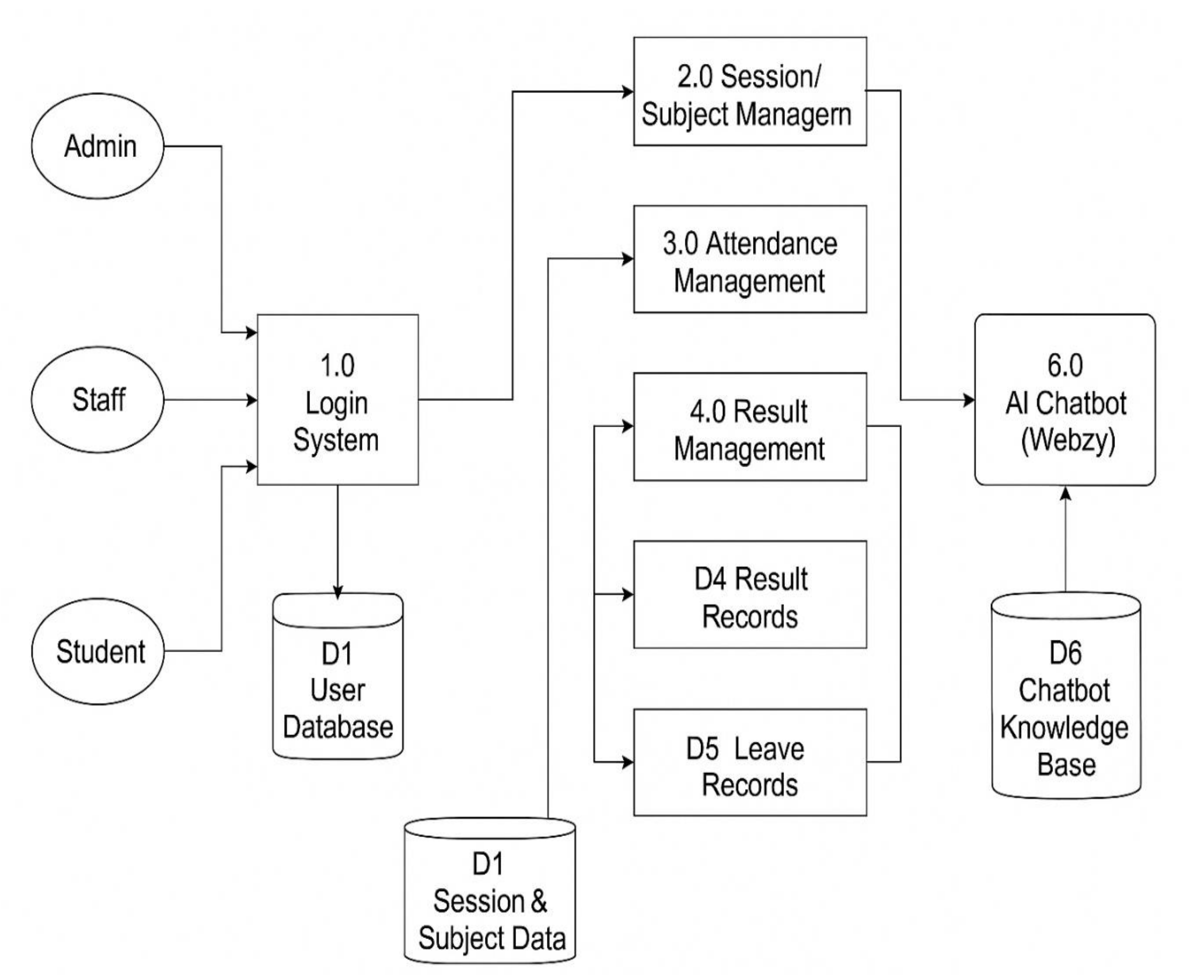


FIGURE 5.5.1 Data Flow Diagram of SMS with Chatbot AI – Webzy

CHAPTER – 6

RESULTS & DISCUSSION

Admin Page →

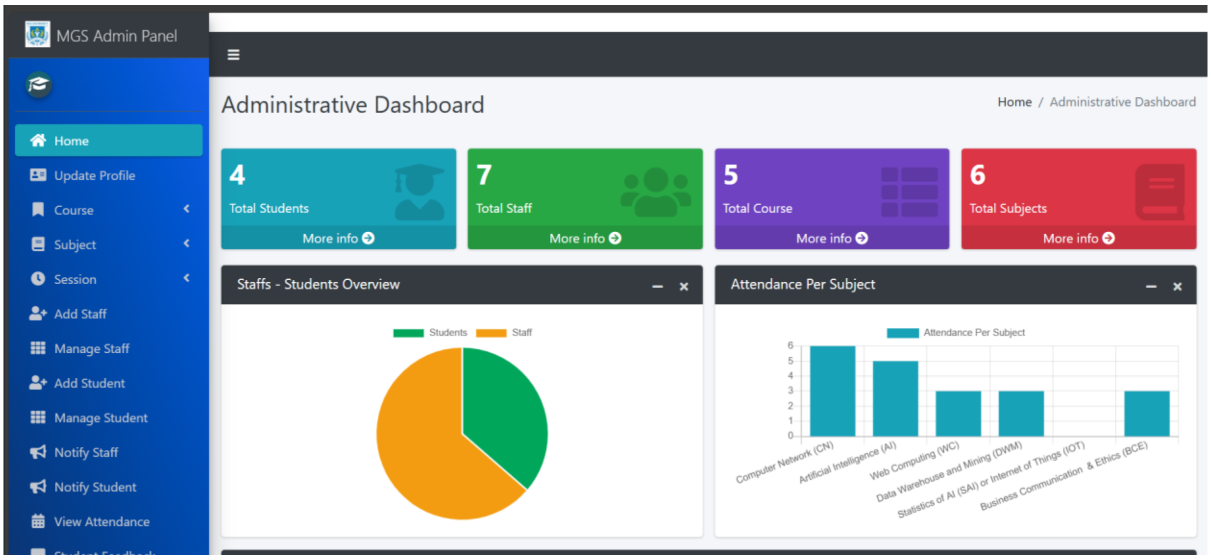
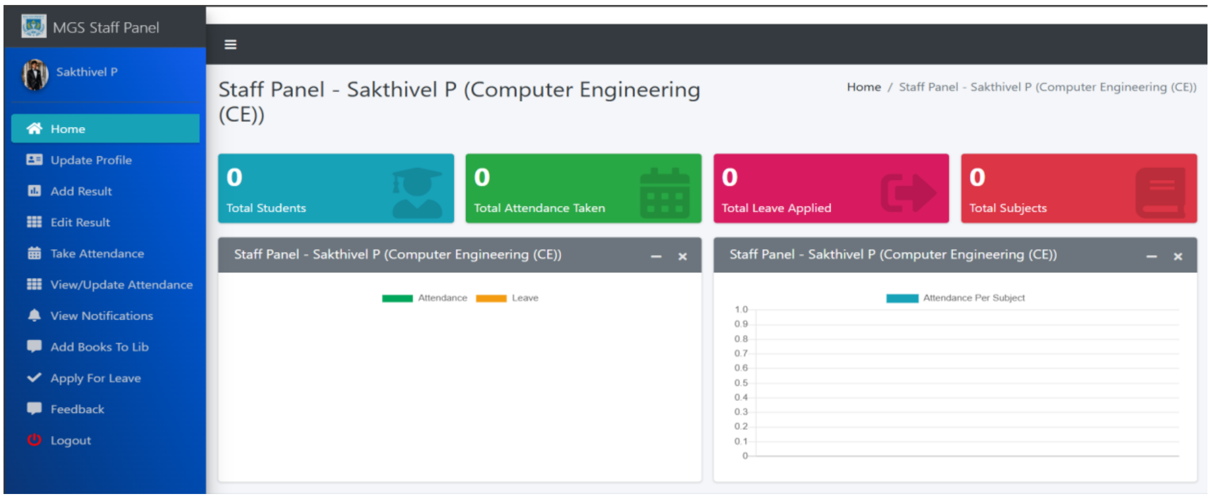


FIGURE 6.1.1 Admin Module Page

The Admin Dashboard provides a comprehensive overview of the system’s core data, such as total students, staff, courses, and subjects. Visual elements like pie charts and bar graphs enhance data understanding—for example, a pie chart shows the ratio of students to staff, while a bar chart presents attendance per subject. This allows administrators to monitor system activity at a glance and make quick decisions. Sidebar navigation enables easy access to key features including staff/student management, attendance, and notifications. The interface is clean, user-friendly, and designed for efficient academic administration.

Staff Module →



Staff Panel - Sakthivel P (Computer Engineering (CE))

0 Total Students

0 Total Attendance Taken

0 Total Leave Applied

0 Total Subjects

Staff Panel - Sakthivel P (Computer Engineering (CE))

Staff Panel - Sakthivel P (Computer Engineering (CE))

FIGURE 6.1.2 Staff Module Page

The Staff Panel provides faculty members with a personalized interface to manage academic activities efficiently. It displays key information such as the total number of students assigned, attendance taken, leaves applied, and total subjects. The sidebar menu includes essential functions like adding/editing results, taking attendance, applying for leave, and sending feedback. Graphical elements like bar charts and pie charts give a quick visual overview of attendance and leave metrics. Though currently the data is zero, the design supports dynamic updates as records are added. This structured layout helps staff streamline their academic responsibilities with ease.

Student Module →

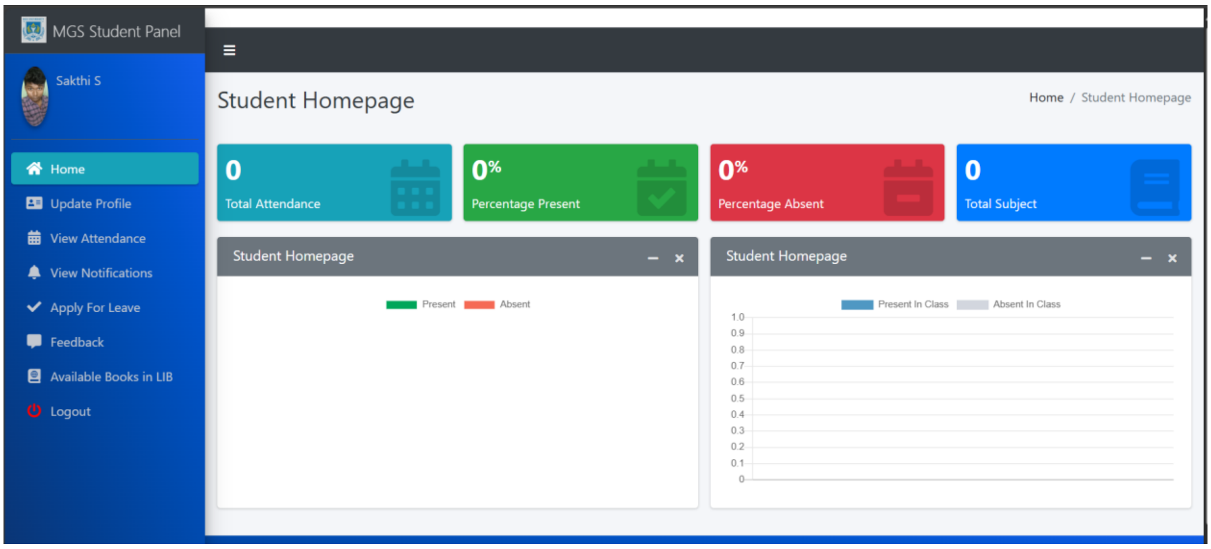


FIGURE 6.1.3 Student module Page

The Student Panel serves as a personalized dashboard for students to monitor their academic status and activities. It presents an overview of key statistics such as total attendance, attendance percentage, absence percentage, and total subjects enrolled. The sidebar offers essential features including profile updates, viewing attendance records, applying for leave, accessing library books, and submitting feedback. The interface includes visual graphs that will dynamically reflect attendance data once available. This intuitive layout empowers students to take ownership of their academic progress with ease and transparency.

Chatbot AI – Webzy →

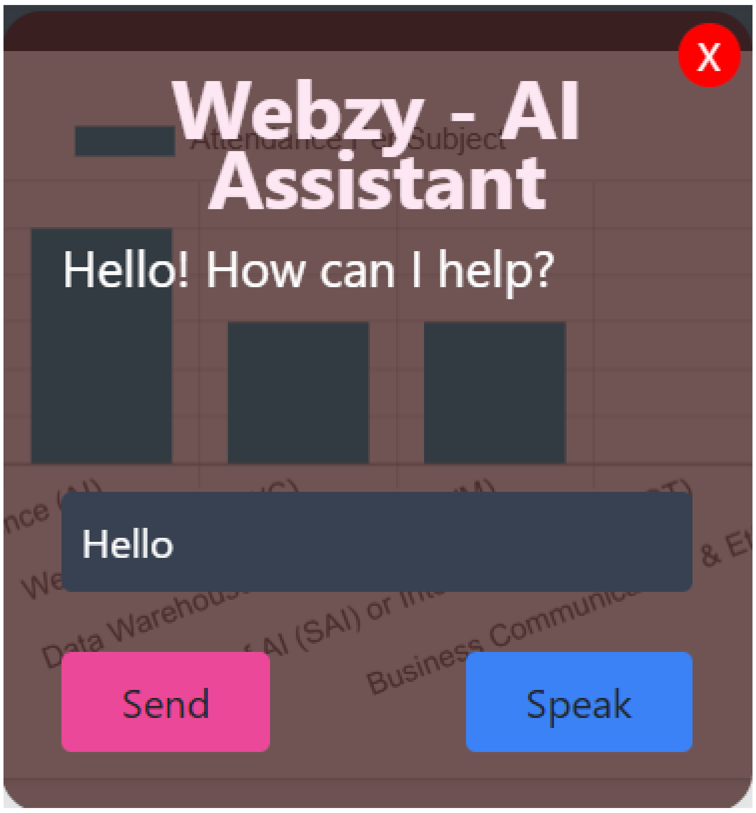


FIGURE 6.1.4 Chatbot AI – Webzy

The Webzy Chatbot Interface provides an interactive platform for users to receive instant assistance regarding academic and administrative queries within the student management system. Built with a user-friendly design, the chatbot allows students and staff to ask questions related to attendance, results, leave status, library books, and profile updates. It utilizes natural language processing to interpret user input and respond appropriately, enhancing the overall user experience. The chatbot operates 24/7, ensuring timely support and reducing manual workload for administrative staff. As data is fetched from the backend dynamically, it ensures responses remain accurate and up-to-date.

CHAPTER – 7

CONCLUSION & FUTURE WORK

Conclusion

The Student Management System (SMS) with AI Chatbot – Webzy is a comprehensive web-based platform developed to streamline the academic and administrative operations of an educational institution. This system efficiently bridges the communication and management gap between students, staff, and administrators by integrating key functionalities like student registration, attendance tracking, result management, leave applications, and library book tracking.

A standout feature of this project is the AI-powered chatbot "Webzy", which enhances user interaction and support. Webzy allows users to query information such as attendance status, exam results, leave requests, and available books in the library—all in natural language. This intelligent assistant helps reduce manual effort, accelerates access to information, and ensures 24/7 availability of basic support functions, thereby improving the user experience significantly.

With clean and intuitive interfaces for Admin, Staff, and Student modules, the system ensures role-based access and functionality, reducing complexity and enhancing usability. The dashboard gives quick visual insights into attendance, results, and subjects, while secure login and update features maintain data integrity and privacy.

In summary, this project demonstrates how modern web technologies and AI can be integrated to create a robust and intelligent academic management tool, increasing efficiency and satisfaction for all stakeholders.

Future Scope & Works

The current version of the Student Management System with Webzy Chatbot offers solid foundational features, but there are numerous opportunities to expand and refine the system in future iterations:

1. **Advanced AI Chatbot Integration:**
    - Upgrade Webzy to handle voice commands and multi-language support.
    - Implement machine learning to personalize responses based on user behavior and history.
2. **Mobile Application Support:**
    - Develop Android and iOS apps to ensure access on the go.1
    - Integrate push notifications for attendance, results, or announcements.
3. **Real-Time Notifications and Alerts:**
    - Add real-time messaging for staff and students regarding timetable changes, exam schedules, and urgent announcements.
4. **Enhanced Analytics:**
    - Include detailed analytics and reporting tools for performance tracking, subject-wise analysis, and staff evaluation.
5. **Biometric and RFID Integration:**
    - Automate attendance using fingerprint or RFID scanners to ensure precision and eliminate proxy entries.
6. **Online Exam & Result Evaluation:**
    - Incorporate online test modules with auto-grading features.
    - Allow instant result publication and performance graphs.
7. **Parent Access Portal:**
    - Provide login access to parents to monitor their ward’s performance, attendance, and fee status.
8. **Cloud Integration:**
    - Move data to a cloud platform for better scalability, security, and remote access capabilities.
    - With these advancements, the Student Management System can evolve into a fully digital campus solution, offering a smarter and more connected academic environment.

APPENDIX 1.1

Main Python File Source Code →

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'college_management_system.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
if __name__ == '__main__':
    main()
```

Login html Source Code →

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <title>SVCET College Portal</title>
   <!-- Tell the browser to be responsive to screen width -->
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <!-- favicon -->
   <link rel="shortcut icon" href="{%  static '/image/logo.png' %}">
   <!-- Font Awesome -->
   <link rel="stylesheet" href="{% static 'plugins/fontawesome-free/css/all.min.css'
%}">
   <!-- Ionicons -->
   <link rel="stylesheet" href="{% static
'https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css' %}">
   <!-- icheck bootstrap -->
   <link rel="stylesheet" href="{% static 'plugins/icheck-bootstrap/icheck-
bootstrap.min.css' %}">
   <!-- Theme style -->
   <link rel="stylesheet" href="{% static 'dist/css/adminlte.min.css' %}">
   <!-- Google Font: Source Sans Pro -->
```

```
    <link
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700"
rel="stylesheet">
    <!-- Google Recaptcha     -->
    <!-- <script src="https://www.google.com/recaptcha/api.js" async defer></script> -
->
</head>
<body class="hold-transition login-page" style="background-image: linear-gradient(to
right top, #051937, #044269, #00719a, #00a4c6, #12d9eb);">
    <div class="login-box">
        <img style="margin-left: 100px; height: 150px; width: 160px;" src="{% static
'dist/img/logo.jpg' %}" rel="icon" alt="logo" />
        <div class="login-logo">
            <a href="" style="color: white;"><b>MGS University</b> Portal</a>
        </div>
        <!-- /.login-logo -->
        <div class="card">
            <div class="card-body login-card-body">
                <!-- <p class="login-box-msg">Sign in to start your session</p> -->
           {% if messages %}
                <div class="col-12">
                    {% for message in messages %}
                    {% if message.tags == 'error' %}
                    <div class="alert alert-danger text-center ">
                        {{message }}
                    </div>
                    {% endif %}
                    {% endfor %}
                </div>
                {% endif %}
                <form action="doLogin/" method="post">
                    {% csrf_token %}
                    <div class="input-group mb-3">
                        <input required type="email"  name='email' class="form-control"
placeholder="Email" >
                        <div class="input-group-append">
                            <div class="input-group-text">
                                <span class="fas fa-envelope"></span>
                            </div>
                        </div>
                    </div>
                    <div class="input-group mb-3">
                        <input required type="password" name='password' class="form-
control" placeholder="Password">
                        <div class="input-group-append">
                            <div class="input-group-text">
                                <span class="fas fa-lock"></span>
                            </div>
                        </div>
                    </div>
                    <!-- <div class="input-group mb-3">
                        <div class="g-recaptcha" data-sitekey="your_site_key"></div>
                    </div> -->
                    <div class="row">
                        <div class="col-8">
                            <div class="icheck-primary">
                                <input type="checkbox" id="remember">
                                <label for="remember">
                                    Remember Me
                                </label>
                            </div>
                        </div>
                        <!-- /.col -->
                        <div class="col-4">
                            <button type="submit" class="btn btn-success btn-block">Log
In</button>
                        </div>
                        <!-- /.col -->
                    </div>
                </form>
```

```
            <p class="mb-1">
                <a href="{% url 'password_reset' %}">Forgot Password?</a>
            </p>
          </div>
          <!-- /.login-card-body -->
        </div>
      </div>
      <!-- /.login-box -->
      <!-- jQuery -->
      <script src="{% static 'plugins/jquery/jquery.min.js' %}"></script>
      <!-- Bootstrap 4 -->
      <script src="{% static 'plugins/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
      <!-- AdminLTE App -->
      <script src="{% static 'dist/js/adminlte.min.js' %}"></script>
    </body></html>
```

Admin Module HTML Source Code →

```
{% load static %}

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <title>MGS University Portal</title>

    <!-- Tell the browser to be responsive to screen width -->

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Font Awesome -->

    <link rel="stylesheet" href="{% static 'plugins/fontawesome-
free/css/all.min.css'%} ">

    <!-- Ionicons -->

    <link rel="stylesheet"
href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">

    <!-- favicon -->

    <link rel="shortcut icon" href="{%  static '/image/logo.png' %}">

    <!-- Tempusdominus Bbootstrap 4 -->

    <link rel="stylesheet"

        href="{% static 'plugins/tempusdominus-bootstrap-4/css/tempusdominus-
bootstrap-4.min.css'%} ">
```

```
    <!-- iCheck -->
    <link rel="stylesheet" href="{% static 'plugins/icheck-bootstrap/icheck-
bootstrap.min.css'%} ">
    <!-- Theme style -->ś
    <link rel="stylesheet" href="{% static 'dist/css/adminlte.min.css'%} ">
    <!-- overlayScrollbars -->
    <link rel="stylesheet" href="{% static
'plugins/overlayScrollbars/css/OverlayScrollbars.min.css'%} ">
    <!-- Daterange picker -->
    <link rel="stylesheet" href="{% static
'plugins/daterangepicker/daterangepicker.css'%} ">
    <!-- Google Font: Source Sans Pro -->
    <link
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,
700" rel="stylesheet">
    {% block custom_css %}
    {% endblock custom_css %}
</head>
<body class="hold-transition sidebar-mini layout-fixed" >
    <div class="wrapper" >
        <!-- Navbar -->
        <nav class="main-header navbar navbar-expand navbar-light "
style="background-color: #343a40;">
```

```
            <!-- Left navbar links -->
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" data-widget="pushmenu" href="#"><i
style="color: white;" class="fas fa-bars"></i></a>
                </li> </ul>
        </nav>
        <!-- /.navbar -->
        <!-- Main Sidebar Container -->
        <div style=" position: fixed; bottom: 10px; right: 10px;">
        </div>
        <div class="content-wrapper">
            <!-- Content Wrapper. Contains page content -->
            <!-- Content Header (Page header) -->
            <div class="content-header">
                <div class="container-fluid">
                    <div class="row mb-2">
                        <div class="col-sm-6">
                            <h1 class="m-0 text-dark">{% block page_title %}{%
endblock page_title %}</h1>
                        </div><!-- /.col -->
                        <div class="col-sm-6">
                <ol class="breadcrumb float-sm-right">
                  <li class="breadcrumb-item"><a href="#">Home</a></li>
                  <li class="breadcrumb-item active">{{ page_title }}</li>
                </ol>
              </div><!-- /.col -->
            </div><!-- /.row -->
          </div><!-- /.container-fluid -->
        </div>
        <!-- /.content-header -->
        <!-- Main content -->
<section class="content">
  <div class="container-fluid">
    <div class="row">
        <div class="col-md-12">
        <div class="form-group">
          {% if messages %}
          {% for message in messages  %}
          {% if message.tags == 'success' %}
          <div class="alert alert-success">
            {{message}}
          </div>
          {% else %}
```

```
            <div class="alert alert-danger">
                {{message}}
            </div>
            {% endif %}
            {% endfor %}
            {% endif %}
        </div>
        </div></div></div></section>
        {% block content %}
        {% endblock content %}
        <!-- /.content -->
    </div>
    <div id="chatbot-container">
        <button id="close-chatbot" style="position: absolute; top: 5px; right:
5px; background: red; color: white; border: none; border-radius: 50%; width:
25px; height: 25px; cursor: pointer;">X</button>
        {% include "main_app/chatbotAI.html" %}
    </div>
    <style>
    #chatbot-container {
        position: fixed;
        bottom: 20px;
```

```
            right: 20px;

            z-index: 1000; /* Ensure it appears above other elements */

            width: 300px; /* Adjust width as needed */

            height: auto; /* Adjust height as needed */

            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /* Optional: Add shadow for
better visibility */

        }

        </style>

        <script>

            // Make the chatbot draggable

            const chatbotContainer = document.getElementById("chatbot-
container");

            chatbotContainer.style.position = "absolute"; // Ensure it's positioned
absolutely for dragging

            chatbotContainer.style.cursor = "move"; // Change cursor to indicate
draggable

            let offsetX = 0, offsetY = 0, isDragging = false;

            chatbotContainer.addEventListener("mousedown", (e) => {

                isDragging = true;

                offsetX = e.clientX - chatbotContainer.offsetLeft;

                offsetY = e.clientY - chatbotContainer.offsetTop;

            });

            document.addEventListener("mousemove", (e) => {
```

```
            if (isDragging) {
                chatbotContainer.style.left = `${e.clientX - offsetX}px`;
                chatbotContainer.style.top = `${e.clientY - offsetY}px`;
            }
        });
        document.addEventListener("mouseup", () => {
            isDragging = false;
        });
    </script>
    <script>
        // Close button functionality
        const closeButton = document.getElementById("close-chatbot");
        closeButton.addEventListener("click", () => {
            chatbotContainer.style.display = "none"; // Hide the chatbot
        });
    </script>
    <button id="reopen-chatbot" style="position: fixed; bottom: 10px; left:
10px; background: blue; color: white; border: none; padding: 10px; border-
radius: 5px; cursor: pointer; display: none;">Reopen Chatbot</button>
    <script>
        const reopenButton = document.getElementById("reopen-chatbot");
        closeButton.addEventListener("click", () => {
            chatbotContainer.style.display = "none";
            reopenButton.style.display = "block"; // Show the reopen button
        });
        reopenButton.addEventListener("click", () => {
            chatbotContainer.style.display = "block"; // Show the chatbot
            reopenButton.style.display = "none"; // Hide the reopen button
        });
    </script>
    <!-- /.content-wrapper -->
    {% include "main_app/footer.html" %}
    {% include "main_app/sidebar_template.html" %}
</div>
<!-- ./wrapper -->
<!-- jQuery -->
<script src="{% static 'plugins/jquery/jquery.min.js'%} "></script>
<!-- jQuery UI 1.11.4 -->
<script src="{% static 'plugins/jquery-ui/jquery-ui.min.js'%} "></script>
<!-- Resolve conflict in jQuery UI tooltip with Bootstrap tooltip -->
<script>
    $.widget.bridge('uibutton', $.ui.button)
</script>
<!-- Bootstrap 4 -->
```

```
    <script src="{% static 'plugins/bootstrap/js/bootstrap.bundle.min.js'%}
"></script>
    <!-- ChartJS -->
    <script src="{% static 'plugins/chart.js/Chart.min.js'%} "></script>
    <!-- Sparkline -->
    <script src="{% static 'plugins/sparklines/sparkline.js'%} "></script>
    <!-- jQuery Knob Chart -->
    <script src="{% static 'plugins/jquery-knob/jquery.knob.min.js'%}
"></script>
    <!-- daterangepicker -->
    <script src="{% static 'plugins/moment/moment.min.js'%} "></script>
    <script src="{% static 'plugins/daterangepicker/daterangepicker.js'%}
"></script>
    <!-- Tempusdominus Bootstrap 4 -->
    <script src="{% static 'plugins/tempusdominus-bootstrap-
4/js/tempusdominus-bootstrap-4.min.js'%} "></script>
    <!-- overlayScrollbars -->
    <script src="{% static
'plugins/overlayScrollbars/js/jquery.overlayScrollbars.min.js'%} "></script>
    <!-- AdminLTE App -->
    <script src="{% static 'dist/js/adminlte.js'%} "></script>
    <!-- AdminLTE dashboard demo (This is only for demo purposes) -->
    <script src="{% static 'dist/js/pages/dashboard.js'%} "></script>
    <!-- AdminLTE for demo purposes -->
    <script src="{% static 'dist/js/demo.js'%} "></script>
    {% block custom_js %}
    {% endblock custom_js %}
</body></html>
```

Admin Module python file →

```
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from .models import *.
class UserModel(UserAdmin):
    ordering = ('email',)
admin.site.register(CustomUser, UserModel)
admin.site.register(Staff)
admin.site.register(Student)
admin.site.register(Course)
admin.site.register(Book)
admin.site.register(IssuedBook)
admin.site.register(Library)
admin.site.register(Subject)
admin.site.register(Session)
```

Satff Module Source Code →

```
{% extends 'main_app/base.html' %}

{% load static %}

{% block page_title %}{{page_title}}{% endblock page_title %}

{% block content %}

<section class="content">

    <div class="container-fluid">

        <!-- Small boxes (Stat box) -->

        <div class="row">

            <div class="col-lg-3 col-6">

                <!-- small box -->

                <div class="small-box bg-info">

                    <div class="inner">

                        <h3>{{total_students}}</h3>

                        <p>Total Students</p>

                    </div>

                    <div class="icon">

                      <i class="nav-icon fas fa-user-graduate"></i>
                    </div>
                </div>
            </div>
            <!-- ./col -->
            <div class="col-lg-3 col-6">
                <!-- small box -->
                <div class="small-box bg-success">
                    <div class="inner">
                        <h3>{{total_attendance}}</h3>
                        <p>Total Attendance Taken</p>
                    </div>
                    <div class="icon">
                      <i class="nav-icon fas fa-calendar-alt"></i>
                    </div>
                </div>
            </div>
            <!-- ./col -->
            <div class="col-lg-3 col-6">
                <!-- small box -->
                <div class="small-box bg-maroon">
                    <div class="inner">
                        <h3>{{total_leave}}</h3>
                        <p>Total Leave Applied</p>
```

```
      </div>
      <div class="icon">
       <i class="nav-icon fas fa-sign-out-alt"></i>
      </div>
    </div>
  </div>
  <!-- ./col -->
  <div class="col-lg-3 col-6">
    <!-- small box -->
    <div class="small-box bg-danger">
      <div class="inner">
        <h3>{{total_subject}}</h3>
        <p>Total Subjects</p>
      </div>
      <div class="icon">
       <i class="nav-icon fas fa-book"></i>
      </div>

    </div>
  </div>
  <!-- ./col -->
</div>
<!-- /.row -->
```

```
        <!-- Main row -->
        <div class="row">
          <div class="col-md-6">
            <!-- LINE CHART -->
            <div class="card card-secondary">
              <div class="card-header">
                <h3 class="card-title">{{page_title}}</h3>
                <div class="card-tools">
                  <button type="button" class="btn btn-tool" data-card-
widget="collapse"><i class="fas fa-minus"></i>
                  </button>
                  <button type="button" class="btn btn-tool" data-card-
widget="remove"><i class="fas fa-times"></i></button>
                </div>
              </div>
              <div class="card-body">
                <div class="chart">
                  <canvas id="pieChart" style="min-height: 250px; height: 250px; max-
height: 250px; max-width: 100%;"></canvas>
                </div>
              </div></div>
              <!-- /.card-body -->
            </div>
            <div class="col-md-6">
```

```
                    <div class="card card-secondary">
              <div class="card-header">
                <h3 class="card-title">{{page_title}}</h3>
                <div class="card-tools">
                  <button type="button" class="btn btn-tool" data-card-
widget="collapse"><i class="fas fa-minus"></i>
                  </button>
                  <button type="button" class="btn btn-tool" data-card-widget="remove"><i
class="fas fa-times"></i></button>
                </div>
              </div>
              <div class="card-body">
                <div class="chart">
                  <canvas id="barChart" style="min-height: 250px; height: 250px; max-
height: 250px; max-width: 100%;"></canvas>
                </div>
              </div>
              <!-- /.card-body -->
            </div>
                </div>
            <!-- right col -->
        </div>
        <!-- /.row (main row) -->
    </div><!-- /.container-fluid -->
```

```
</section>

{% endblock content %}

{% block custom_js %}

  <script>

      $(document).ready(function(){

        var donutData        = {

            labels: ['Attendance', 'Leave'],

            datasets: [

              {

                data:[{{total_attendance}}, {{total_leave}}],

                backgroundColor : ['#00a65a', '#f39c12',],

              }

            ]

          }

        var pieChartCanvas = $('#pieChart').get(0).getContext('2d')

        var pieData        = donutData;

        var pieOptions     = {

          maintainAspectRatio : false,

          responsive : true,

        }

        //Create pie or douhnut chart

        // You can switch between pie and douhnut using the method below.

        var pieChart = new Chart(pieChartCanvas, {
```

```
      type: 'pie',
      data: pieData,
      options: pieOptions
    });
    var subject_list = {{ subject_list|safe|escape }};
    var attendance_list = {{ attendance_list }};
        var barChartData = {
  labels  : subject_list,
  datasets: [
  {
    label               : 'Attendance Per Subject',
    backgroundColor     : '#17A2B8',
    borderColor         : 'rgba(60,141,188,0.8)',
    pointRadius          : false,
    pointColor          : '#3b8bba',
    pointStrokeColor    : 'rgba(60,141,188,1)',
    pointHighlightFill  : '#fff',
    pointHighlightStroke: 'rgba(60,141,188,1)',
    data                : attendance_list
  },
  ]
}
    var barChartCanvas = $('#barChart').get(0).getContext('2d')
    var temp0 = barChartData.datasets[0]
    //var temp1 = areaChartData.datasets[1]
    barChartData.datasets[0] = temp0
   // barChartData.datasets[1] = temp0

var stackedBarChartOptions = {
  responsive              : true,
  maintainAspectRatio     : false,
  scales: {
    xAxes: [{
      stacked: true,
    }],
    yAxes: [{
      stacked: true
    }]
  }
}

    var barChart = new Chart(barChartCanvas, {
      type: 'bar',
      data: barChartData,
      options: stackedBarChartOptions
    })
```

```
      })
      </script>
      <!-- The core Firebase JS SDK is always required and must be listed first -->
      <script src="https://www.gstatic.com/firebasejs/7.23.0/firebase-app.js"></script>

      <!-- TODO: Add SDKs for Firebase products that you want to use
       https://firebase.google.com/docs/web/setup#available-libraries -->
      <script src="https://www.gstatic.com/firebasejs/7.23.0/firebase-
analytics.js"></script>
      <script src="https://www.gstatic.com/firebasejs/7.22.1/firebase-
messaging.js"></script>
      <script>
         // Your web app's Firebase configuration
         // For Firebase JS SDK v7.20.0 and later, measurementId is optional
         var firebaseConfig = {
            apiKey: "AIzaSyBarDWWHTfTMSrtc5Lj3Cdw5dEvjAkFwtM",
            authDomain: "sms-with-django.firebaseapp.com",
            databaseURL: "https://sms-with-django.firebaseio.com",
            projectId: "sms-with-django",
            storageBucket: "sms-with-django.appspot.com",
            messagingSenderId: "945324593139",
            appId: "1:945324593139:web:03fa99a8854bbd38420c86",
            measurementId: "G-2F2RXTL9GT"
```

```
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig)
const messaging = firebase.messaging();
function InitializeFireBaseMessaging() {
    messaging
        .requestPermission()
        .then(function () {
            console.log("Notification Permission");
            return messaging.getToken();
        })
        .then(function (token) {
            console.log("Token : " + token);
            sendToServer(token);
        })
        .catch(function (reason) {
            console.log(reason)
        })
}
messaging.onMessage(function (payload) {
    const notificationOption = {
        body: payload.notification.body,
        icon: payload.notification.icon,
        }
        if (Notification.permission == 'granted') {
            var notification = new Notification(payload.notification.title,
notificationOption);
            notification.onclick = function (event) {
                event.preventDefault();
                window.open(payload.notification.click_action, "_blank");
                notification.close();
            }
        }
        console.log(payload);
    });
    messaging.onTokenRefresh(function () {
        messaging.getToken()
            .then(function (newToken) {
                console.log("New Token : " + newToken);
                sendToServer(newToken);
            })
            .catch(function (reason) {
                console.log(reason)
            })
    })
```

```
function sendToServer(token){
  $.ajax({
    url: "{% url 'staff_fcmtoken' %}",
    type: 'POST',
    data: {
        token: token,
    }
}).done(function (response) {
}).fail(function (response) {
})
}
InitializeFireBaseMessaging(); </script>{% endblock custom_js %}
```

Staff Module Python File →

```
import json
from django.contrib import messages
from django.core.files.storage import FileSystemStorage
from django.http import HttpResponse, JsonResponse
from django.shortcuts import (HttpResponseRedirect, get_object_or_404,redirect,
render)
from django.urls import reverse
from django.views.decorators.csrf import csrf_exempt
from .forms import *
from .models import *
from . import forms, models

from datetime import date

def staff_home(request):

    staff = get_object_or_404(Staff, admin=request.user)

    total_students = Student.objects.filter(course=staff.course).count()

    total_leave = LeaveReportStaff.objects.filter(staff=staff).count()

    subjects = Subject.objects.filter(staff=staff)

    total_subject = subjects.count()

    attendance_list = Attendance.objects.filter(subject__in=subjects)

    total_attendance = attendance_list.count()

    attendance_list = []

    subject_list = []

    for subject in subjects:

        attendance_count = Attendance.objects.filter(subject=subject).count()

        subject_list.append(subject.name)

        attendance_list.append(attendance_count)

    context = {

        'page_title': 'Staff Panel - ' + str(staff.admin.first_name) + ' ' +
str(staff.admin.last_name[0]) + '' + ' (' + str(staff.course) + ')',

        'total_students': total_students,

        'total_attendance': total_attendance,

        'total_leave': total_leave,

        'total_subject': total_subject,
        'subject_list': subject_list,
        'attendance_list': attendance_list
    }
    return render(request, 'staff_template/home_content.html', context)
def staff_take_attendance(request):
    staff = get_object_or_404(Staff, admin=request.user)
    subjects = Subject.objects.filter(staff_id=staff)
    sessions = Session.objects.all()
    context = {
        'subjects': subjects,
        'sessions': sessions,
        'page_title': 'Take Attendance'
    }
    return render(request, 'staff_template/staff_take_attendance.html', context)
@csrf_exempt
def get_students(request):
    subject_id = request.POST.get('subject')
    session_id = request.POST.get('session')
    try:
        subject = get_object_or_404(Subject, id=subject_id)
        session = get_object_or_404(Session, id=session_id)
        students = Student.objects.filter(
            course_id=subject.course.id, session=session)
        student_data = []
        for student in students:
            data = {
                    "id": student.id,
                    "name": student.admin.last_name + " " + student.admin.first_name
                    }
            student_data.append(data)
        return JsonResponse(json.dumps(student_data), content_type='application/json',
safe=False)
    except Exception as e:
        return e
@csrf_exempt
def save_attendance(request):
    student_data = request.POST.get('student_ids')
    date = request.POST.get('date')
    subject_id = request.POST.get('subject')
    session_id = request.POST.get('session')
    students = json.loads(student_data)
    try:
        session = get_object_or_404(Session, id=session_id)
        subject = get_object_or_404(Subject, id=subject_id)
        attendance = Attendance(session=session, subject=subject, date=date)
        attendance.save()
        for student_dict in students:
            student = get_object_or_404(Student, id=student_dict.get('id'))
            attendance_report = AttendanceReport(student=student,
attendance=attendance, status=student_dict.get('status'))
            attendance_report.save()
    except Exception as e:
        return None
    return HttpResponse("OK")
def staff_update_attendance(request):
    staff = get_object_or_404(Staff, admin=request.user)
    subjects = Subject.objects.filter(staff_id=staff)
    sessions = Session.objects.all()
    context = {
        'subjects': subjects,
        'sessions': sessions,
        'page_title': 'Update Attendance'
    }
    return render(request, 'staff_template/staff_update_attendance.html', context)
@csrf_exempt
def get_student_attendance(request):
    attendance_date_id = request.POST.get('attendance_date_id')
    try:
        date = get_object_or_404(Attendance, id=attendance_date_id)
        attendance_data = AttendanceReport.objects.filter(attendance=date)
        student_data = []
        for attendance in attendance_data:
            data = {"id": attendance.student.admin.id,
                    "name": attendance.student.admin.last_name + " " +
attendance.student.admin.first_name,
                    "status": attendance.status}
            student_data.append(data)
        return JsonResponse(json.dumps(student_data), content_type='application/json',
safe=False)
    except Exception as e:
        return e
@csrf_exempt
def update_attendance(request):
    student_data = request.POST.get('student_ids')
    date = request.POST.get('date')
    students = json.loads(student_data)
    try:
        attendance = get_object_or_404(Attendance, id=date)
        for student_dict in students:
            student = get_object_or_404(
                Student, admin_id=student_dict.get('id'))
            attendance_report = get_object_or_404(AttendanceReport, student=student,
attendance=attendance)
```

```
            attendance_report.status = student_dict.get('status')

            attendance_report.save()

    except Exception as e:

        return None

    return HttpResponse("OK")

def staff_apply_leave(request):

    form = LeaveReportStaffForm(request.POST or None)

    staff = get_object_or_404(Staff, admin_id=request.user.id)

    context = {

        'form': form,

        'leave_history': LeaveReportStaff.objects.filter(staff=staff),

        'page_title': 'Apply for Leave'

    }

    if request.method == 'POST':

        if form.is_valid():

            try:

                obj = form.save(commit=False)

                obj.staff = staff

                obj.save()

                messages.success(

                    request, "Application for leave has been submitted for review")

                return redirect(reverse('staff_apply_leave'))

            except Exception:
                messages.error(request, "Could not apply!")
        else:
            messages.error(request, "Form has errors!")
    return render(request, "staff_template/staff_apply_leave.html", context)
def staff_feedback(request):
    form = FeedbackStaffForm(request.POST or None)
    staff = get_object_or_404(Staff, admin_id=request.user.id)
    context = {
        'form': form,
        'feedbacks': FeedbackStaff.objects.filter(staff=staff),
        'page_title': 'Add Feedback'
    }
    if request.method == 'POST':
        if form.is_valid():
            try:
                obj = form.save(commit=False)
                obj.staff = staff
                obj.save()
                messages.success(request, "Feedback submitted for review")
                return redirect(reverse('staff_feedback'))
            except Exception:
                messages.error(request, "Could not Submit!")
        else:
            messages.error(request, "Form has errors!")
    return render(request, "staff_template/staff_feedback.html", context)
def staff_view_profile(request):
    staff = get_object_or_404(Staff, admin=request.user)
    form = StaffEditForm(request.POST or None, request.FILES or
None,instance=staff)
    context = {'form': form, 'page_title': 'View/Update Profile'}
    if request.method == 'POST':
        try:
            if form.is_valid():
                first_name = form.cleaned_data.get('first_name')
                last_name = form.cleaned_data.get('last_name')
                password = form.cleaned_data.get('password') or None
                address = form.cleaned_data.get('address')
                gender = form.cleaned_data.get('gender')
                passport = request.FILES.get('profile_pic') or None
                admin = staff.admin
                if password != None:
                    admin.set_password(password)
                if passport != None:
                    fs = FileSystemStorage()
                    filename = fs.save(passport.name, passport)
                    passport_url = fs.url(filename)
                    admin.profile_pic = passport_url
                admin.first_name = first_name
                admin.last_name = last_name
                admin.address = address
                admin.gender = gender
                admin.save()
                staff.save()
                messages.success(request, "Profile Updated!")
                return redirect(reverse('staff_view_profile'))
            else:
                messages.error(request, "Invalid Data Provided")
                return render(request, "staff_template/staff_view_profile.html", context)
        except Exception as e:
            messages.error(
                request, "Error Occured While Updating Profile " + str(e))
            return render(request, "staff_template/staff_view_profile.html", context)
    return render(request, "staff_template/staff_view_profile.html", context)
@csrf_exempt
def staff_fcmtoken(request):
    token = request.POST.get('token')
    try:
        staff_user = get_object_or_404(CustomUser, id=request.user.id)
        staff_user.fcm_token = token
```

```
            staff_user.save()

            return HttpResponse("True")

        except Exception as e:

            return HttpResponse("False")

def staff_view_notification(request):

    staff = get_object_or_404(Staff, admin=request.user)

    notifications = NotificationStaff.objects.filter(staff=staff)

    context = {

        'notifications': notifications,

        'page_title': "View Notifications"

    }

    return render(request, "staff_template/staff_view_notification.html", context)

def staff_add_result(request):

    staff = get_object_or_404(Staff, admin=request.user)

    subjects = Subject.objects.filter(staff=staff)

    sessions = Session.objects.all()

    context = {

        'page_title': 'Result Upload',

        'subjects': subjects,

        'sessions': sessions

    }

    if request.method == 'POST':

        try:
            student_id = request.POST.get('student_list')
            subject_id = request.POST.get('subject')
            test = request.POST.get('test')
            exam = request.POST.get('exam')
            student = get_object_or_404(Student, id=student_id)
            subject = get_object_or_404(Subject, id=subject_id)
            try:
                data = StudentResult.objects.get(
                    student=student, subject=subject)
                data.exam = exam
                data.test = test
                data.save()
                messages.success(request, "Scores Updated")
            except:
                result = StudentResult(student=student, subject=subject, test=test,
exam=exam)
                result.save()
                messages.success(request, "Scores Saved")
        except Exception as e:
            messages.warning(request, "Error Occured While Processing Form")
    return render(request, "staff_template/staff_add_result.html", context)
@csrf_exempt
def fetch_student_result(request):
    try:
        subject_id = request.POST.get('subject')
        student_id = request.POST.get('student')
        student = get_object_or_404(Student, id=student_id)
        subject = get_object_or_404(Subject, id=subject_id)
        result = StudentResult.objects.get(student=student, subject=subject)
        result_data = {
            'exam': result.exam,
            'test': result.test
        }
        return HttpResponse(json.dumps(result_data))
    except Exception as e:
        return HttpResponse('False')
#library
def add_book(request):
    if request.method == "POST":
        name = request.POST['name']
        author = request.POST['author']
        isbn = request.POST['isbn']
        category = request.POST['category']
        books = Book.objects.create(name=name, author=author, isbn=isbn,
category=category )
        books.save()
        alert = True
        return render(request, "staff_template/add_book.html", {'alert':alert})
    context = {
        'page_title': "Add Book"
    }
    return render(request, "staff_template/add_book.html",context)
#issue book
def issue_book(request):
    form = forms.IssueBookForm()
    if request.method == "POST":
        form = forms.IssueBookForm(request.POST)
        if form.is_valid():
            obj = models.IssuedBook()
            obj.student_id = request.POST['name2']
            obj.isbn = request.POST['isbn2']
            obj.save()
            alert = True
            return render(request, "staff_template/issue_book.html", {'obj':obj,
'alert':alert})
    return render(request, "staff_template/issue_book.html", {'form':form})
def view_issued_book(request):
    issuedBooks = IssuedBook.objects.all()
    details = []
    for i in issuedBooks:
        days = (date.today()-i.issued_date)
        d=days.days
        fine=0
        if d>14:
            day=d-14
            fine=day*5
        books = list(models.Book.objects.filter(isbn=i.isbn))
        # students = list(models.Student.objects.filter(admin=i.admin))
        i=0
        for l in books:
t=(books[i].name,books[i].isbn,issuedBooks[0].issued_date,issuedBooks[0].expiry_da
te,fine)
            i=i+1
            details.append(t)
    return render(request, "staff_template/view_issued_book.html",
{'issuedBooks':issuedBooks, 'details':details})
```

Student Module Sourec Code →

```
{% extends 'main_app/base.html' %}
{% load static %}
{% block page_title %}{{page_title}}{% endblock page_title %}
{% block content %}
<section class="content">
    <div class="container-fluid">
```

```
        <!-- Small boxes (Stat box) -->
        <div class="row">
           <div class="col-lg-3 col-6">
              <!-- small box -->
              <div class="small-box bg-info">
                 <div class="inner">
                    <h3>{{total_attendance}}</h3>
                    <p>Total Attendance</p>
                 </div>
                 <div class="icon">
                   <i class="nav-icon fas fa-calendar-alt"></i>
                 </div>
                 {# <a href="#" class="small-box-footer">More info <i class="fas fa-
arrow-circle-right"></i></a> #}
              </div>
           </div>
           <!-- ./col -->
           <div class="col-lg-3 col-6">
              <!-- small box -->
              <div class="small-box bg-success">
                 <div class="inner">
                    <h3>{{ percent_present }}<sup style="font-size:
20px">%</sup></h3>
```

```
            <p>Percentage Present</p>
          </div>
          <div class="icon">
            <i class="nav-icon fas fa-calendar-check"></i>
          </div>
          {# <a href="#" class="small-box-footer">More info <i class="fas fa-
arrow-circle-right"></i></a> #}
        </div>
      </div>
      <!-- ./col -->
      <div class="col-lg-3 col-6">
        <!-- small box -->
        <div class="small-box bg-danger">
          <div class="inner">
            <h3>{{percent_absent}}<sup style="font-size: 20px">%</sup></h3>
            <p>Percentage Absent</p>
          </div>
          <div class="icon">
            <i class="nav-icon fas fa-calendar-minus"></i>
          </div>
          {# <a href="#" class="small-box-footer">More info <i class="fas fa-
arrow-circle-right"></i></a> #}
        </div>
```

```
            </div>

            <!-- ./col -->

            <div class="col-lg-3 col-6">

                <!-- small box -->

                <div class="small-box bg-primary">

                    <div class="inner">

                        <h3>{{total_subject}}</h3>

                        <p>Total Subject</p>

                    </div>

                    <div class="icon">

                      <i class="nav-icon fas fa-book"></i>

                    </div>

                    {# <a href="#" class="small-box-footer">More info <i class="fas fa-
arrow-circle-right"></i></a> #}

                </div>

            </div>

            <!-- ./col -->

        </div>

        <div class="row">

            <div class="col-lg-6">

                        <!-- DONUT CHART -->

                        <div class="card card-secondary">

                          <div class="card-header">
```

```
              <h3 class="card-title">{{page_title}}</h3>

              <div class="card-tools">

                <button type="button" class="btn btn-tool" data-card-
widget="collapse"><i class="fas fa-minus"></i>

                </button>

                <button type="button" class="btn btn-tool" data-card-
widget="remove"><i class="fas fa-times"></i></button>

              </div>

            </div>

            <div class="card-body">

              <canvas id="attendanceData" style="min-height: 250px; height:
250px; max-height: 250px; max-width: 100%;"></canvas>

            </div>

            <!-- /.card-body -->

          </div>

          <!-- /.card -->

    </div>

    <div class="col-lg-6">


          <div class="card card-secondary">

            <div class="card-header">

              <h3 class="card-title">{{page_title}}</h3>

              <div class="card-tools">
```

```
                              <button type="button" class="btn btn-tool" data-card-
widget="collapse"><i class="fas fa-minus"></i>
                              </button>
                              <button type="button" class="btn btn-tool" data-card-
widget="remove"><i class="fas fa-times"></i></button>
                            </div>
                          </div>
                          <div class="card-body">
                            <div class="chart">
                              <canvas id="attendanceStatistics" style="min-height: 250px;
height: 250px; max-height: 250px; max-width: 100%;"></canvas>
                          <!-- /.card-body -->
                        <!-- /.card -->
                        </div>
                        </div>
                        </div>
                        </div>
                        </div>
                        </div>
                        </section>
{% endblock content %}
{% block custom_js %}
  <script>
$(document).ready(function(){
```

```
//Dataset
var subjects = {{data_name|safe}}
var data_present = {{data_present}}
var data_absent = {{data_absent}}
//-------------
//- DONUT CHART -
var attendanceDataCanvas = $('#attendanceData').get(0).getContext('2d')
var donutData        = {
  labels: [
     'Present', 'Absent'
  ],
  datasets: [
    {
      data: [{{percent_present}}, {{percent_absent}}],
      backgroundColor : ['#00a65a','#f56954'],
    }
  ]
}
var donutOptions     = {
  maintainAspectRatio : false,
  responsive : true,
}
var attendanceData = new Chart(attendanceDataCanvas, {
      type: 'doughnut',
      data: donutData,
      options: donutOptions
    });
//attendanceStatistics
//Bar Chart
var areaChartData = {
    labels  : subjects,
    datasets: [
    {
        label               : 'Present In Class',
        backgroundColor     : 'rgba(60,141,188,0.9)',
        borderColor         : 'rgba(60,141,188,0.8)',
        pointRadius          : false,
        pointColor          : '#3b8bba',
        pointStrokeColor    : 'rgba(60,141,188,1)',
        pointHighlightFill  : '#fff',
        pointHighlightStroke: 'rgba(60,141,188,1)',
        data                : data_present
      },
      {
        label               : 'Absent In Class',
        backgroundColor     : 'rgba(210, 214, 222, 1)',
        borderColor         : 'rgba(210, 214, 222, 1)',
        pointRadius         : false,
        pointColor          : 'rgba(210, 214, 222, 1)',
        pointStrokeColor    : '#c1c7d1',
        pointHighlightFill  : '#fff',
        pointHighlightStroke: 'rgba(220,220,220,1)',
        data                : data_absent
      },
    ]
  }
var barChartCanvas = $('#attendanceStatistics').get(0).getContext('2d')
var barChartData = jQuery.extend(true, {}, areaChartData)
var temp = areaChartData.datasets[0]
barChartData.datasets[0] = temp
var barChartOptions = {
  responsive              : true,
  maintainAspectRatio     : false,
  datasetFill             : false
}

var barChart = new Chart(barChartCanvas, {
  type: 'bar',
  data: barChartData,
  options: barChartOptions
})
})
  </script>
        <!-- The core Firebase JS SDK is always required and must be listed first -->
        <script src="https://www.gstatic.com/firebasejs/7.23.0/firebase-app.js"></script>

        <!-- TODO: Add SDKs for Firebase products that you want to use
         https://firebase.google.com/docs/web/setup#available-libraries -->
        <script src="https://www.gstatic.com/firebasejs/7.23.0/firebase-
analytics.js"></script>
        <script src="https://www.gstatic.com/firebasejs/7.22.1/firebase-
messaging.js"></script>
        <script>
            // Your web app's Firebase configuration
            // For Firebase JS SDK v7.20.0 and later, measurementId is optional
            var firebaseConfig = {
                apiKey: "AIzaSyBarDWWHTfTMSrtc5Lj3Cdw5dEvjAkFwtM",
                authDomain: "sms-with-django.firebaseapp.com",
                databaseURL: "https://sms-with-django.firebaseio.com",
                projectId: "sms-with-django",
                storageBucket: "sms-with-django.appspot.com",
                messagingSenderId: "945324593139",
  appId: "1:945324593139:web:03fa99a8854bbd38420c86",
  measurementId: "G-2F2RXTL9GT"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig)
const messaging = firebase.messaging();
function InitializeFireBaseMessaging() {
  messaging
    .requestPermission()
    .then(function () {
      console.log("Notification Permission");
      return messaging.getToken();
    })
    .then(function (token) {
      console.log("Token : " + token);
      sendToServer(token);
    })
    .catch(function (reason) {
      console.log(reason)
    })
}
messaging.onMessage(function (payload) {
  const notificationOption = {
```

```
                body: payload.notification.body,
                icon: payload.notification.icon,
            }
            if (Notification.permission == 'granted') {
                var notification = new Notification(payload.notification.title,
notificationOption);
                notification.onclick = function (event) {
                    event.preventDefault();
                    window.open(payload.notification.click_action, "_blank");
                    notification.close();
                }
            }
            console.log(payload);
        });
        messaging.onTokenRefresh(function () {
            messaging.getToken()
                .then(function (newToken) {
                    console.log("New Token : " + newToken);
                    sendToServer(newToken);
                })
                .catch(function (reason) {
                    console.log(reason)
                })
        })
        function sendToServer(token){
          $.ajax({
            url: "{% url 'student_fcmtoken' %}",
            type: 'POST',
            data: {
                token: token,
            }
        }).done(function (response) {
        }).fail(function (response) {
        })
        }
        InitializeFireBaseMessaging();
    </script>
{% endblock custom_js %}
```

Student Module Python File →

```
import json
import math
from datetime import datetime
from django.contrib import messages
from django.core.files.storage import FileSystemStorage
from django.http import HttpResponse, JsonResponse
from django.shortcuts import (HttpResponseRedirect, get_object_or_404,
                        redirect, render)
from django.urls import reverse
from django.views.decorators.csrf import csrf_exempt
from .forms import *
from .models import *
def student_home(request):
    student = get_object_or_404(Student, admin=request.user)
    total_subject = Subject.objects.filter(course=student.course).count()
    total_attendance = AttendanceReport.objects.filter(student=student).count()
    total_present = AttendanceReport.objects.filter(student=student,
status=True).count()
    if total_attendance == 0:  # Don't divide. DivisionByZero
        percent_absent = percent_present = 0
    else:
        percent_present = math.floor((total_present/total_attendance) * 100)
        percent_absent = math.ceil(100 - percent_present)
    subject_name = []
    data_present = []
    data_absent = []
    subjects = Subject.objects.filter(course=student.course)
    for subject in subjects:
        attendance = Attendance.objects.filter(subject=subject)
        present_count = AttendanceReport.objects.filter(
            attendance__in=attendance, status=True, student=student).count()
        absent_count = AttendanceReport.objects.filter(
            attendance__in=attendance, status=False, student=student).count()
        subject_name.append(subject.name)
        data_present.append(present_count)
        data_absent.append(absent_count)
    context = {
        'total_attendance': total_attendance,
        'percent_present': percent_present,
        'percent_absent': percent_absent,
        'total_subject': total_subject,
        'subjects': subjects,
        'data_present': data_present,
        'data_absent': data_absent,
        'data_name': subject_name,
        'page_title': 'Student Homepage'
    }
    return render(request, 'student_template/home_content.html', context)
@ csrf_exempt
def student_view_attendance(request):
    student = get_object_or_404(Student, admin=request.user)
    if request.method != 'POST':
        course = get_object_or_404(Course, id=student.course.id)
        context = {
            'subjects': Subject.objects.filter(course=course),
            'page_title': 'View Attendance'
        }
        return render(request, 'student_template/student_view_attendance.html', context)
    else:
        subject_id = request.POST.get('subject')
        start = request.POST.get('start_date')
        end = request.POST.get('end_date')
        try:
            subject = get_object_or_404(Subject, id=subject_id)
            start_date = datetime.strptime(start, "%Y-%m-%d")
            end_date = datetime.strptime(end, "%Y-%m-%d")
            attendance = Attendance.objects.filter(
                date__range=(start_date, end_date), subject=subject)
            attendance_reports = AttendanceReport.objects.filter(
                attendance__in=attendance, student=student)
            json_data = []
            for report in attendance_reports:
                data = {
                    "date":  str(report.attendance.date),
                    "status": report.status
                }
            json_data.append(data)
        return JsonResponse(json.dumps(json_data), safe=False)
    except Exception as e:
        return None
def student_apply_leave(request):
    form = LeaveReportStudentForm(request.POST or None)
    student = get_object_or_404(Student, admin_id=request.user.id)
    context = {
        'form': form,
        'leave_history': LeaveReportStudent.objects.filter(student=student),
        'page_title': 'Apply for leave'
    }
    if request.method == 'POST':
        if form.is_valid():
            try:
                obj = form.save(commit=False)
                obj.student = student
                obj.save()
                messages.success(
                    request, "Application for leave has been submitted for review")
                return redirect(reverse('student_apply_leave'))
            except Exception:
                messages.error(request, "Could not submit")
        else:
            messages.error(request, "Form has errors!")
    return render(request, "student_template/student_apply_leave.html", context)
def student_feedback(request):
    form = FeedbackStudentForm(request.POST or None)
    student = get_object_or_404(Student, admin_id=request.user.id)
    context = {
        'form': form,
        'feedbacks': FeedbackStudent.objects.filter(student=student),
        'page_title': 'Student Feedback'
    }
    if request.method == 'POST':
        if form.is_valid():
            try:
                obj = form.save(commit=False)
                obj.student = student
                obj.save()
                messages.success(
                    request, "Feedback submitted for review")
                return redirect(reverse('student_feedback'))
            except Exception:
                messages.error(request, "Could not Submit!")
        else:
            messages.error(request, "Form has errors!")
    return render(request, "student_template/student_feedback.html", context)
def student_view_profile(request):
    student = get_object_or_404(Student, admin=request.user)
    form = StudentEditForm(request.POST or None, request.FILES or None,
                           instance=student)
    context = {'form': form,
               'page_title': 'View/Edit Profile'
               }
    if request.method == 'POST':
        try:
            if form.is_valid():
                first_name = form.cleaned_data.get('first_name')
                last_name = form.cleaned_data.get('last_name')
                password = form.cleaned_data.get('password') or None
                address = form.cleaned_data.get('address')
                gender = form.cleaned_data.get('gender')
                passport = request.FILES.get('profile_pic') or None
                admin = student.admin
                if password != None:
                    admin.set_password(password)
                if passport != None:
                    fs = FileSystemStorage()
                    filename = fs.save(passport.name, passport)
                    passport_url = fs.url(filename)
                    admin.profile_pic = passport_url
                admin.first_name = first_name
                admin.last_name = last_name
                admin.address = address
                admin.gender = gender
                admin.save()
                student.save()
                messages.success(request, "Profile Updated!")
                return redirect(reverse('student_view_profile'))
            else:
                messages.error(request, "Invalid Data Provided")
        except Exception as e:
            messages.error(request, "Error Occured While Updating Profile " + str(e))

    return render(request, "student_template/student_view_profile.html", context)
@csrf_exempt
def student_fcmtoken(request):
    token = request.POST.get('token')
    student_user = get_object_or_404(CustomUser, id=request.user.id)
    try:
        student_user.fcm_token = token
        student_user.save()
        return HttpResponse("True")
    except Exception as e:
        return HttpResponse("False")



def student_view_notification(request):
    student = get_object_or_404(Student, admin=request.user)
    notifications = NotificationStudent.objects.filter(student=student)
    context = {
        'notifications': notifications,
        'page_title': "View Notifications"
    }
    return render(request, "student_template/student_view_notification.html", context)
def student_view_result(request):
    student = get_object_or_404(Student, admin=request.user)
    results = StudentResult.objects.filter(student=student)
    context = {
        'results': results,
        'page_title': "View Results"
    }
    return render(request, "student_template/student_view_result.html", context)
#library
```

```
def view_books(request):
    books = Book.objects.all()
    context = {
        'books': books,
        'page_title': "Library"
    }
    return render(request, "student_template/view_books.html", context)
```

Chatbot AI – Webzy Source code →

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Webzy Chatbot</title>
<link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"
rel="stylesheet">
</head>
<body class="bg-gray-900 text-white font-sans h-screen flex flex-col justify-center
items-center">
  <div style="background-color: rgba(60, 22, 22, 0.735);" class="w-full max-w-md p-
6 rounded-2xl shadow-2xl">
    <h2 class="text-xl font-bold mb-2 text-center text-pink-100">Webzy - AI
Assistant</h2>
```

```
    <div id="response" style="color: white;" class="text-lg mb-4 h-20 overflow-y-
auto"></div>
    <input type="text" id="userInput" placeholder="Ask something..." class="w-full p-
2 rounded bg-gray-700 text-white mb-4 focus:outline-none"/>
    <div class="flex justify-between items-center">
      <button onclick="sendMessage()" class="bg-pink-500 px-4 py-2 rounded
hover:bg-pink-600">Send</button>
      <button onclick="startVoiceRecognition()" class="bg-blue-500 px-4 py-2 rounded
hover:bg-blue-600">Speak</button>
    </div>
  </div>
  <script>
window.speechSynthesis.cancel();
 let isChatbotShutdown = false;
    async function sendMessage() {
      const inputField = document.getElementById("userInput");
      const responseDiv = document.getElementById("response");
      const userMessage = inputField.value.trim().toLowerCase();
 if (!userMessage) {
        responseDiv.innerText = "Please enter a message.";
        return;
      }
if (userMessage === "stop" || userMessage === "exit") {
    window.speechSynthesis.cancel();
  responseDiv.innerText = "Speech stopped. You can type a new command.";
  return;
}
responseDiv.innerText = "Processing...";
inputField.value = "";
try {
  const res = await fetch("/chatbot/reply/", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-CSRFToken": getCSRFToken()  // Ensure CSRF token is included
    },
    body: JSON.stringify({ message: userMessage })
      });
      if (!res.ok) {
        throw new Error(`HTTP error! status: ${res.status}`);
      }
      const data = await res.json();
      const reply = data.reply;
  if (reply.toLowerCase().includes("program has stopped")) {
    isChatbotShutdown = true;
    responseDiv.innerText = "Webzy is shutting down.";
    speakText("Webzy is shutting down.");
    inputField.disabled = true;
    return;
  }
      responseDiv.innerText = reply;
      speakText(reply);
    } catch (err) {
      console.error("Error fetching response:", err);
      responseDiv.innerText = "Error getting response. Please try again.";
    }
  }
  function getCSRFToken() {
    const cookies = document.cookie.split(';');
    for (let cookie of cookies) {
      const [name, value] = cookie.trim().split('=');
      if (name === 'csrftoken') {
        return value;
      }
    }
    console.warn("CSRF token not found in cookies.");
    return '';
  }
  async function startVoiceRecognition() {
    const responseDiv = document.getElementById("response");
```

```
      responseDiv.innerText = "Listening... Please speak.";
      try {
        const recognition = new (window.SpeechRecognition ||
window.webkitSpeechRecognition)();
        recognition.lang = "en-US";
        recognition.onresult = async (event) => {
          const userMessage = event.results[0][0].transcript;
          if (isChatbotShutdown || ["exit", "stop"].includes(userMessage.toLowerCase()))
{
  responseDiv.innerText = "Webzy is shutting down.";
  speakText("Webzy is shutting down.");
  isChatbotShutdown = true;
  document.getElementById("userInput").disabled = true;
  return;
}
          console.log("Recognized speech:", userMessage);
          document.getElementById("userInput").value = userMessage;
          responseDiv.innerText = "Processing...";
          const res = await fetch("/chatbot/reply/", {
            method: "POST",
            headers: {
              "Content-Type": "application/json",
              "X-CSRFToken": getCSRFToken()
        },
        body: JSON.stringify({ message: userMessage })
      });
      if (!res.ok) {
        throw new Error(`HTTP error! status: ${res.status}`);
      }
      const data = await res.json();
      const reply = data.reply;
      responseDiv.innerText = reply;
      speakText(reply);
    };
    recognition.onerror = (event) => {
      console.error("Speech recognition error:", event.error);
      responseDiv.innerText = "Error recognizing speech. Please try again.";
    };
    recognition.start();
  } catch (err) {
    console.error("Speech recognition not supported:", err);
    responseDiv.innerText = "Speech recognition is not supported in this browser.";
  }
}
function speakText(text) {
  const synth = window.speechSynthesis;
```

```
      synth.cancel();
      const utterance = new SpeechSynthesisUtterance(text);
      utterance.lang = "en-US";
      synth.speak(utterance);
    }
  document.getElementById('userInput').addEventListener('keypress', function (e) {
    if (e.key === 'Enter') {
      e.preventDefault();  // Prevent form submission
      sendMessage();  // Trigger sendMessage() when Enter is pressed
    }
  });
  window.onload = function () {
    speakText("Hello, I am Webzy, your AI assistant. How can I assist you today?");
  };
  </script>
</body>
</html>
```

Chatbot Module Python file →

```
'''import speech_recognition as sr
import pyttsx3
import time
from datetime import datetime
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import openai  # Gemini API library
# Configure the Gemini API with your API key
openai.api_key = "AIzaSyDliTjiJL0o3Gm0wQHerzy1Q5S4vvOG-oY"
# Predefined training data and responses
training_sentences = [
    "hello", "how are you", "who are you", "time", "what is the time",
    "flip a coin", "roll a dice", "attendance tracking", "fee management", "exam performance",
    "AI chatbot", "parent communication", "library management", "cloud system", "role-based access",
    "next cm","AI in education",  "how to update profile", "add course", "manage course", "add subject",
    "add session", "add staff", "view attendance", "date", "whats the date",
    "today's date", "your name", "what is your name", "what date is it", "what day is it"
]
```

```
responses = [
    "Hello! How can I help?",
    "I'm doing great! Thanks for asking.",
    "I'm Webzy, your AI assistant.",
    f"The current time is {time.strftime('%I:%M %p')}",
    "It's " + ("Heads!" if time.time() % 2 > 1 else "Tails!"),
    f"You rolled a {str(int(time.time() % 6) + 1)}",
    "Marks attendance automatically and generates reports.",
    "Allows online payments and sends reminders.",
    "Analyzes student results to improve learning.",
    "Helps students with quick answers and guidance.",
    "Keeps parents updated on student progress.",
    "Helps students find books and study materials online.",
    "Stores data securely and makes access easy anywhere.",
    "Gives different permissions to students, teachers, and admins.",
    "Uses AI to personalize learning and predict student needs.",
    "Thalapathy Vijay Sir.",
    "In the left sidebar, at the top option, click to update your profile.",
    "In the sidebar, select the second option. If you see any sub-options inside, it will
show the 'add subject' option.",
    "It is shown inside the course options.",
    "It is shown on the left side in the second option. Select it to add the subject.",
    "It is shown as the third option in the same list. Click to add or manage the session.",
    "It is shown as the fourth option. A person symbol with a plus sign is visible—click it to add staff.",
    "Below the staff and student notify options, a calendar symbol appears. Click it to view attendance.",
    f"Today's date is {datetime.now().strftime('%B %d, %Y')}.",
    "My name is Webzy, your website assistant.",
    "I am Webzy, your website assistant.",
]
```

REFERENCES

[1] A. Kumar and R. Sharma, "Design and Implementation of Student Information System," International Journal of Computer Applications, vol. 141, no. 5, pp. 15–19, May 2016.

[2] M. R. J. Qureshi, S. Ullah, and M. A. Khan, "Development of a Web Based Student Information System," International Journal of Scientific & Engineering Research, vol. 6, no. 2, pp. 154–158, Feb. 2015.

[3] D. S. Rani and K. S. Rao, "An Effective Chatbot for College Management System Using AI," International Journal of Advanced Research in Computer and Communication Engineering, vol. 9, no. 5, pp. 1–6, May 2020.

[4] B. Shawar and E. Atwell, "Chatbots: Are They Really Useful?," LDV Forum – GLDV Journal for Computational Linguistics and Language Technology, vol. 22, no. 1, pp. 29–49, 2007.

[5] R. Jain and P. Kumar, "Integration of Artificial Intelligence Based Chatbots in Education," International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 8, no. 6S, pp. 289–293, April 2019.

[6] "Dialogflow Documentation," Google Cloud, [Online]. Available: https://cloud.google.com/dialogflow/docs. [Accessed: 05-May-2025].

[7] Bootstrap, "Bootstrap · The most popular HTML, CSS, and JS library," [Online]. Available: https://getbootstrap.com. [Accessed: 05-May-2025].

[8] Django Project, "The Web framework for perfectionists with deadlines," [Online]. Available: https://www.djangoproject.com. [Accessed: 05-May-2025].

[9] OpenAI, "ChatGPT: Optimizing Language Models for Dialogue," [Online]. Available: https://openai.com/chatgpt. [Accessed: 05-May-2025].

https://youtu.be/FoAXWjkWvrM?si=x-ABmk7oErTzEQ_W

https://youtu.be/RemVc1u-jVU?si=4QC4OkhlwCK5GgSa

https://youtu.be/ohb-oknZQa8?si=0sX2XSMQOhTmIZkV

https://chatgpt.com/c/67d82ecd-0808-8003-84c3-c38a2faf15eb

https://chatgpt.com/

https://youtu.be/15CUwvA9lck?si=3ovO6yN45lIiUyEp

https://www.geeksforgeeks.org/student-management-system-in-python/

https://github.com/PrathameshDhande22/Student-Management-System-in-Python

https://codewithcurious.com/projects/student-management-system-using-python/

https://pythongeeks.org/python-student-management-system/

https://chatgpt.com/c/68171ade-a038-8013-8d2e-d300963e95e9