AcadME (e-portal)

A Project Report

Submitted for the partial fulfillment for the award of the degree of

B.Tech - CS

Submitted by

Group : CSD 45

Ananya Garg	2216212
Ardhisha Jain	2216235
Asmita Kamal	2216241
Dakshita Arora	2216258
Gyanvi Gupta	2216274

Under the supervision of

Dr. Ajay Kumar Yadav

Department of Computer Science

Banasthali Vidyapith

Session: 2024-25

Certificate

Certified that the following students have carried out the project work titled "**AcadME**" from 1-August-2024 to 31-March-2025 for the award of the **Bachelor of Technology** from **Banasthali Vidyapith** under my supervision. The report embodies results of original work and studies carried out by students themselves and the contents of the project report do not form the basis for the award of any other degree to the candidates or to anybody else.

Member 1: 2216212 Ananya Garg CSE

Member 2: 2216235 Ardhisha Jain CSE

Member 3: 2216241 Asmita Kamal CSE

Member 4: 2216258 Dakshita Arora CSE

Member 5: 2216274 Gyanvi Gupta CSE

Dr. Ajay Kumar Yadav 04/04/2025

Designation: Assistant Professor

Place: Department of Computer Science

Date: 04-04-2025

Abstract

AcadME e-Portal is a web-based academic management system designed to streamline student, teacher, and admin activities at Banasthali Vidyapeeth. It replaces manual processes with an automated, role-based platform for managing attendance, timetables, assignments, results, and study materials. The system provides admins with tools to manage users, create timetables, and track attendance. Teachers can take attendance, upload assignments, and provide feedback, while students can view timetables, track attendance, and submit assignments. Additional features include customized FAQs, an achievements gallery, and notifications for real-time updates. AcadMe e-Portal enhances academic management by reducing workload, and making information easily accessible.

Acknowledgement

We would like to express our profound gratitude to our HOD, Dr. Rajiv Singh, of the Computer Science department, and Dean, Dr. C K Jha of Banasthali Vidyapith for their contributions to the completion of our project titled AcadME.

We are thankful to Banasthali Vidyapith for providing us with the opportunity to pursue a project in such a dynamic and promising field. This project allowed us to gain valuable practical experience and apply our theoretical knowledge.

We are incredibly grateful to Dr. Ajay Kumar Yadav for being our mentor throughout this project. His guidance and support in every step of the way were instrumental in our success. His expertise and willingness to answer our questions helped us navigate challenges and ensure the project's direction.

We extend our gratitude to Dr. Neelam Sharma and Dr. Deepak Kumar for their roles as project coordinators. Their guidance on writing the report and keeping us updated on project deadlines was crucial. Their feedback on our work helped us improve the clarity and quality of our report.

Table of Contents

INTRODUCTION

AcadME e-Portal is a comprehensive web-based platform designed to streamline academic and administrative tasks at Banasthali Vidyapith. The current system relies on manual processes or email-based communication, which can be inefficient. AcadME aims to centralize and automate these operations, providing students, teachers, and admins with an integrated, user-friendly solution. The platform is built using Django for the backend, ensuring security, scalability, and ease of use.

The primary objective of AcadME e-Portal is to digitize and streamline academic management processes, replacing traditional methods with an efficient, digital solution. The platform aims to reduce manual workload, ensuring seamless coordination of academic activities.

By implementing this system, students can conveniently track their academic progress, teachers can manage their responsibilities more effectively, and administrators can oversee operations efficiently.

Key Features of AcadME e-Portal

Admin Features

- Manage staff, students, courses, subjects, and session years.
- The admin can create and manage timetables for various course types, ensuring organized schedules for students and teacher
- Add branches and view student attendance and results.
- View and reply to student and staff feedback.
- Send notifications.

Teacher Features

- Take and update attendance records.

- Upload assignments and view submitted assignments.
- Add and edit student results.
- Provide feedback and view feedback reply.
- Upload study materials.
- View the timetable assigned by the admin.
- View notifications.

Student Features

- View attendance records.
- View result grade cards.
- Submit assignments.
- Access uploaded materials.
- Receive and send feedback messages.
- View notifications.
- View the timetable assigned by the admin.

Common Features Across Dashboards

- **Timetable Management:** Admin creates timetables for students and teachers, which are accessible in respective dashboards.
- **Customized FAQs:** Each dashboard has its own FAQ section tailored to user-specific queries.
- **Achievements & Gallery:** A dedicated section displaying college achievements and event highlights.
- **About Us Section:** Provides an overview of the college and its objectives.
- **Notifications:** Real-time announcements.

By integrating these features into a single platform, AcadME e-Portal ensures efficient academic management, reduces manual work, and enhances communication between students, teachers, and administrators.

REQUIREMENT ANALYSIS

(SRS)

REQUIREMENT SPECIFICATION

1. User Authentication and Role-Based Dashboard Access

1.1 Description and Priority

This feature ensures secure user authentication and provides role-based access to dashboards for admins, teachers, and students. Users log in using their email and password, and if credentials are invalid, an error message is displayed. Upon successful login, users are directed to their respective dashboards with appropriate functionalities.

1.2 Stimulus/Response Sequence

- **Stimulus**: User enters email and password.
  Response: The system validates the credentials. If valid, the system directs the user to the appropriate dashboard; if invalid, it displays an error message.
- **Stimulus**: User interacts with dashboard features.
  Response: The system provides access to role-specific functionalities based on the user's permissions.
- **Stimulus**: User logs out.
  Response: The system logs out the user and redirects them to the login page.

1.3 Functional Requirements

- REQ-1: Validate user credentials (email and password).
- REQ-2: Display error messages for invalid login attempts.
- REQ-3: Redirect users to their respective dashboards upon successful login.
- REQ-4: Provide role-specific features in each dashboard.
- REQ-5: Include a logout option that returns users to the login page.

2. Admin Management of Users and Academic Resources

2.1 Description and Priority

Admins can manage students, staff, courses, subjects, branches, and session years.

2.2 Stimulus/Response Sequences

- **Stimulus**: Admin opens the dashboard and adds details of a new student along with username and password.
  Response: The student is added into the system.
- **Stimulus**: Admin views or edits student details.
  Response: The system updates the student's information and reflects the changes in the student module.
- **Stimulus**: Admin deletes a student.
  Response: The system removes the student from the database and this change gets reflected in the student module.
- **Stimulus**: Admin manages courses, subjects, branches and teachers information.
  Response: The system allows the admin to add, update, or delete this information.
- **Stimulus**: Admin manages session years.
  Response: The system allows the admin to add this information.

2.3 Functional Requirements

- REQ-1: Add, update, and delete student records, including usernames and passwords.
- REQ-2: Manage teacher details and assign subjects.
- REQ-3: Handle course, subject, and session year data.

3. Assignment Management

3.1 Description and Priority

Teachers can create assignments, and students can submit them.

3.2 Stimulus/Response Sequences

- **Stimulus**: The teacher navigates to the "Upload Assignments" section. **Response**: The system prompts the teacher to enter details such as the

assignment title, due date, description, and any attached files. Upon submission, the assignment is saved.

- **Stimulus**: The student selects an assignment to view its details. **Response**: The system displays the assignment details, including the submission deadline and any associated files.
- **Stimulus**: The student uploads a completed assignment and submits it. **Response**: The system accepts the submission, confirms the upload, and records the submission date and time.
- **Stimulus**: The teacher views student submissions for an assignment. **Response**: The system displays a list of submissions and teachers can download it as well.

3.3 Functional Requirements

- REQ-1: Teachers can create new assignments.
- REQ-2: Students can view and submit assignments.
- REQ-3: The system records submission timestamps.
- REQ-4: Teachers can download and grade assignments.

4. Attendance Management

4.1 Description and Priority

Teachers can log in, mark attendance, and view & update attendance records. Students can check their attendance history, and admins can access all attendance data.

4.2 Stimulus/Response Sequences

a) Taking Attendance (Teacher)

- **Stimulus**: The teacher navigates to the Take Attendance section, selects the course, branch, subject, class type, session year, and date then marks attendance.
  Response: The system records the attendance and confirms the successful entry.

b) Viewing and Updating Attendance (Teacher)

- **Stimulus**: The teacher navigates to the View Update Attendance section, selects the course, branch, subject, class type, session year, and date to view and update the attendance.
  Response: The system displays the attendance records for the selected criteria.

c) Viewing Attendance (Student)

- **Stimulus**: The student selects the subject, class type ,month to view attendance.
  Response: The system displays the attendance records for the selected subject.

d) Viewing Attendance(Admin)

- **Stimulus**: The admin selects the course, branch, subject, class type, session year, and month.
  Response: The system displays the attendance records for the selected criteria.

4.3 Functional Requirements

- REQ-1: Teachers can mark attendance by course, branch, subject, class type, session year, and date.
- REQ-2: Teachers can update and view attendance records.
- REQ-3: Students can log in and view attendance.
- REQ-4: Admins can access attendance data.

5. Upload Materials

5.1 Description and Priority

Teachers can upload handouts and study materials by selecting the relevant course, branch, and subject. Students can access these study materials.

5.2 Stimulus/Response Sequences

a) Uploading Study Materials (Teacher)

- **Stimulus**: The teacher navigates to the "Upload Materials" section, selects the course, branch, subject, and uploads the material.
  Response: The system saves the material and makes it accessible to the respective students.

b) Viewing Study Materials (Student)

- **Stimulus**: The student navigates to the "ViewMaterials" section and selects the subject.
  Response: The system displays the available study materials for the selected subject, and students can download these materials.

5.3 Functional Requirements

- REQ-1: Teachers can upload study materials by selecting the course, branch, subject.
- REQ-2: Students can access & download uploaded materials based on their enrolled subjects.

6. Grading System

6.1 Description and Priority

Teachers can upload & edit student marks for assignments & periodicals, and students can view their results.

6.2 Stimulus/Response Sequences

- **Stimulus**: Teacher navigates to the Add Result section, selects a course, branch, subject, session year, and then uploads marks.
  Response: The system sends the marks to the student module.
- **Stimulus**: Teacher navigates to the Edit Result section, selects a course, branch, subject, session year and then edit marks.
  Response: The system sends the marks to the student module.
- **Stimulus**: Students navigate to the Grade Card section, to view their results.
  Response: The system displays the marks.

6.3 Functional Requirements

- REQ-1: Teachers can enter student marks.
- REQ-2: The system updates the student module with marks.
- REQ-3: Students can view their marks.

7. Feedback Management System

7.1 Description and Priority

Teachers and students can send feedback to the admins and receive responses.

7.2 Stimulus/Response Sequences

- **Stimulus**: Teacher clicks on the "Feedback" in their dashboard. **Response**: The system opens a form for the teacher to write and submit feedback to the admin.
- **Stimulus**: Admin receives and replies to the teacher's feedback. **Response**: The system displays the reply status in the teacher's dashboard.
- **Stimulus**: Student sends feedback to the Admin. **Response**: The system delivers the feedback to the admin and shows the reply status in the student panel.

7.3 Functional Requirements

- REQ-1: The system provides a form for teachers to submit feedback to the admin.
- REQ-2: The system allows the admin to receive and reply to feedback from teachers.
- REQ-3: The system displays the reply status in the teacher's dashboard.
- REQ-4: The system allows students to send feedback to the Admin.
- REQ-5: The system displays the admin's reply status in the student panel.

8. Timetable Management

8.1 Description and Priority

The admin can create and manage timetables for various course types, ensuring organized schedules for students and teachers.

8.2 Stimulus/Response Sequences

- **Stimulus**: Admin navigates to the "Timetable" section and creates a new timetable.

  Response: The system saves the timetable and makes it available for students and teachers.
- **Stimulus**: Teacher or student logs into their dashboard and selects the "View Timetable" option.

  Response: The system displays the assigned timetable for the respective user.

8.3 Functional Requirements

- REQ-1: Admins create timetables.
- REQ-2: Students and teachers can view timetables.

9. Notifications System

9.1 Description and Priority

Admins can send notifications regarding exams, events, and important announcements to both students & teachers.

9.2 Stimulus/Response Sequences

- **Stimulus**: Admin creates and sends a notification regarding an exam, event, or announcement.

  Response: The system delivers the notification to all relevant students and staff members.
- **Stimulus**: Student or teacher logs into their dashboard and checks the "Notifications" section.

  Response: The system displays the latest notifications in real-time.

9.3 Functional Requirements

- REQ-1: Admins can send notifications.
- REQ-2: Students and staff receive notifications.

10. Common Features Across Dashboards

a) **Customized FAQs**: Each dashboard has a tailored FAQ section.
   - **Stimulus**: User selects the "FAQ" icon from the dashboard. **Response**: The system displays frequently asked questions specific to the user's role (Admin, Teacher, or Student).

b) **Achievements & Gallery**: Highlights college achievements and events.
   - **Stimulus**: User navigates to the "Achievements & Gallery" section. **Response**: The system displays the latest achievements and event highlights.

c) **About Us Section**: Provides an overview of the college.
   - **Stimulus**: User clicks on the "About Us" section. **Response**: The system displays details about the college, its history, mission, and vision.

11. Other Nonfunctional Requirements

Non-functional requirements are the aspects of a software system that describe how the system should perform rather than what it should do.

12. Performance Requirements

The website is interactive. So, in every action-response of the system, there are no immediate delays. The website performs efficiently and responds quickly to user interactions.

13. Security Requirements

The website prioritizes security and protects user data from unauthorized access or breaches. Different user roles (e.g., admin, students, teachers) shall have distinct levels of authorization.

HARDWARE AND SOFTWARE REQUIREMENTS

1. Operating Environment

Compatible:

- **Browsers:** The website will be compatible with common web browsers like Google Chrome, Microsoft Edge, etc.
- **Operating Systems:** Compatible with any OS.

Hardware Limitations:

- The platform will work efficiently on devices running Windows and Android, utilizing any common web browser such as Google Chrome and Mozilla Firefox.

2. Hardware Interfaces

Server Side:

- **RAM:** 32 GB (recommended)
- **Hard Disk:** 2 TB HDD or SSD (recommended for fast data storage).
- **Processor:** 2.4 GHz 64-bit processor (Intel Core i5 or above).

Client Side:

- **RAM:** 8 GB (recommended) for smooth browsing and access to the portal.
- **Hard Disk:** 128 GB or more.
- **Processor:** 2.4 GHz (Intel Core i3 or above).

Developer Side:

- **RAM:** 8 GB (recommended for development tasks).
- **Hard Disk:** 512 GB SSD or more (for storing project files and software).
- **Processor:** Intel® Core i3 8th Generation or onwards.

3. Software Interfaces

Server Side:

- **Operating System:** Windows Server 2016 or onwards.
- **Web Server:** Django's Built-in Development Server.
- **Database:** MySQL for storing and managing all user data, attendance records, assignments, grades, etc.

Client Side:

- **Operating System:** Any OS (Windows).
- **Browser:** Any modern browser (Google Chrome ,Microsoft Edge)

Developer Side:

- **Operating System:** Windows 8 and onwards
- **IDE:** PyCharm for Django(Python) development.
- **Web Development Tools:**
    - **Frontend:** HTML, CSS, JavaScript.
    - **Backend:** Django
    - **Database Management:** MySQL Workbench for managing MySQL databases.

FEASIBILITY STUDY

1. Economic Feasibility

- Utilizes free and open-source tools like Django, MySQL, and PyCharm, reducing costs.
- Minimal development and maintenance expenses with readily available resources.

2. Technical Feasibility

- Built on widely supported technologies: **Django, MySQL, HTML, CSS, JavaScript**.
- Compatible with modern web browsers and devices.
- Secure, scalable, and easy to deploy on cloud or local servers.

3. Operational Feasibility

- User-friendly design with clear navigation and self-explanatory interfaces.
- Supports multiple roles (Students, Teachers, Admins) with tailored functionalities.
- Minimal training is required for effective use.

4. Behavioral Feasibility

- Intuitive interface with guided workflows and error handling.
- Real-time feedback and validation for smooth user experience.
- Accessible and compliant with data protection standards.

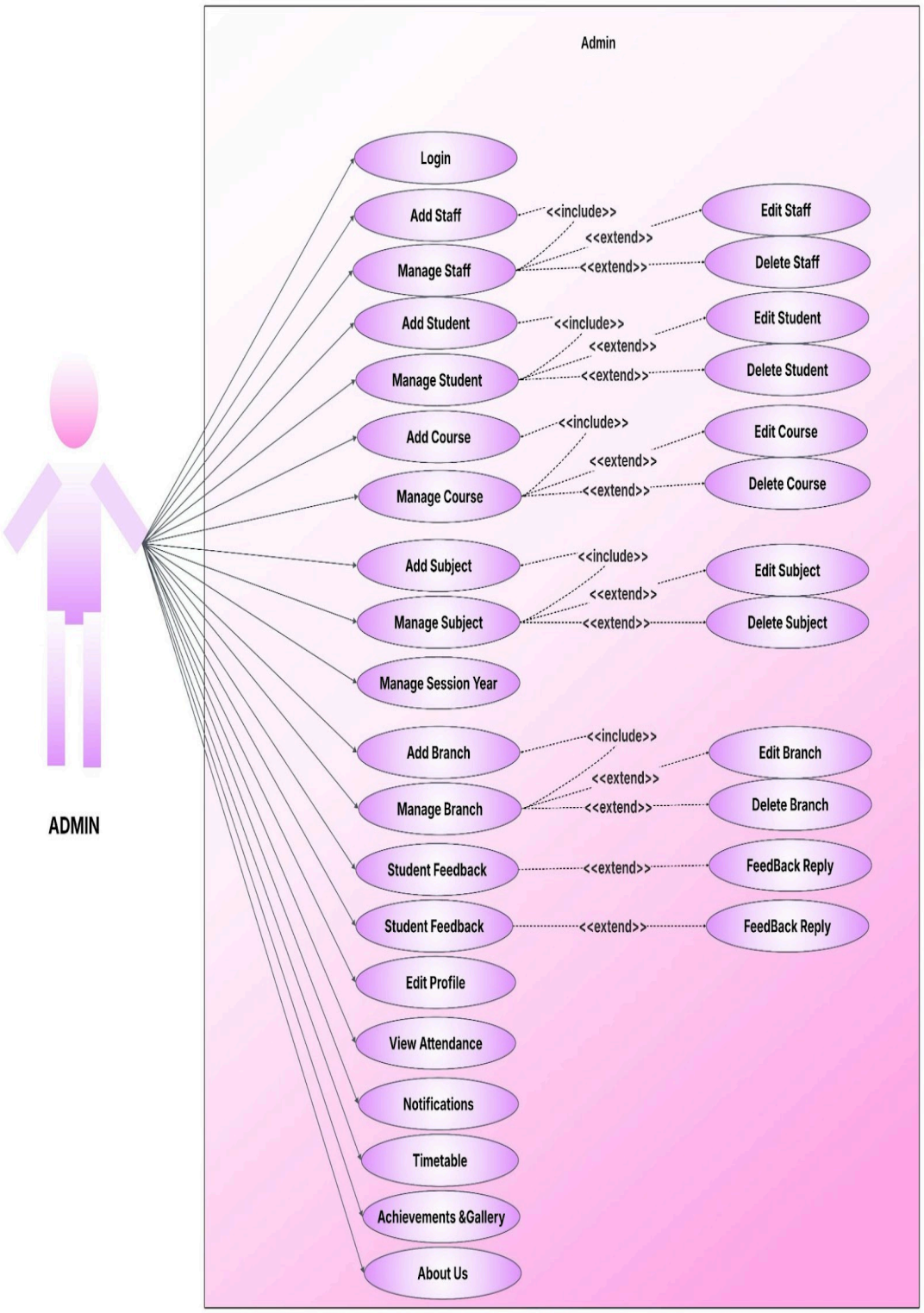
PRODUCT FUNCTIONS

The following are the main functions included in the AcadMe:

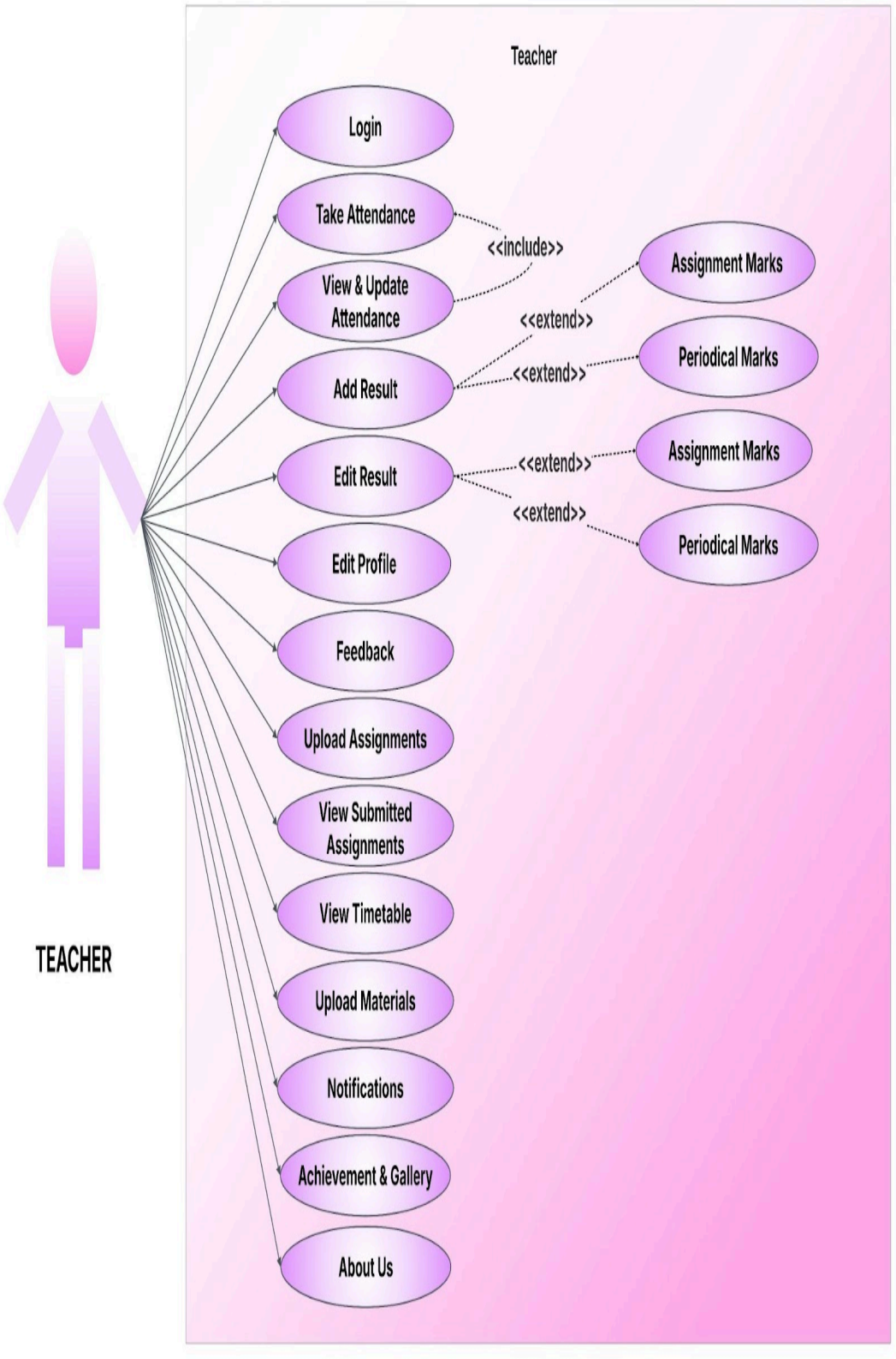
- **User Registration and Authentication**: It securely verifies user credentials (username and password) during login and manages user sessions.
- **Attendance Tracking**: The system allows users to track and manage attendance records, ensuring that attendance data is stored and accessible as needed.
- **Assignment Submission and Management**: Users can view and upload assignments.
- **Feedback System:** Users can provide and manage feedback on various aspects of the academic experience. The system includes for viewing and responding to feedback by admin.
- **Student and Staff Information Management:** The portal facilitates the management of student and staff information, including adding, updating, and deleting personal information.
- **Course and Subject Management:** The system allows for the management of courses and subjects, including the addition, updation, and deletion of a course.
- **Grading System:** Users can manage and view marks for Periodicals, assignments.
- **Timetable Management:** Users can view timetables and admins can create & organize class schedules.
- **Upload Materials:** Teachers can upload handouts, books, and other study resources for easy access which can be accessed by students.
- **Notifications:** The portal provides features for real-time notification and updates on academic activities and events.
- **FAQ:** Access to FAQ feature that answers frequently asked questions for quick support.
- **Achievements & About Us:** Explore a curated gallery of academic and extracurricular successes alongside detailed information about the institution's history and values.

USE-CASE DIAGRAMS

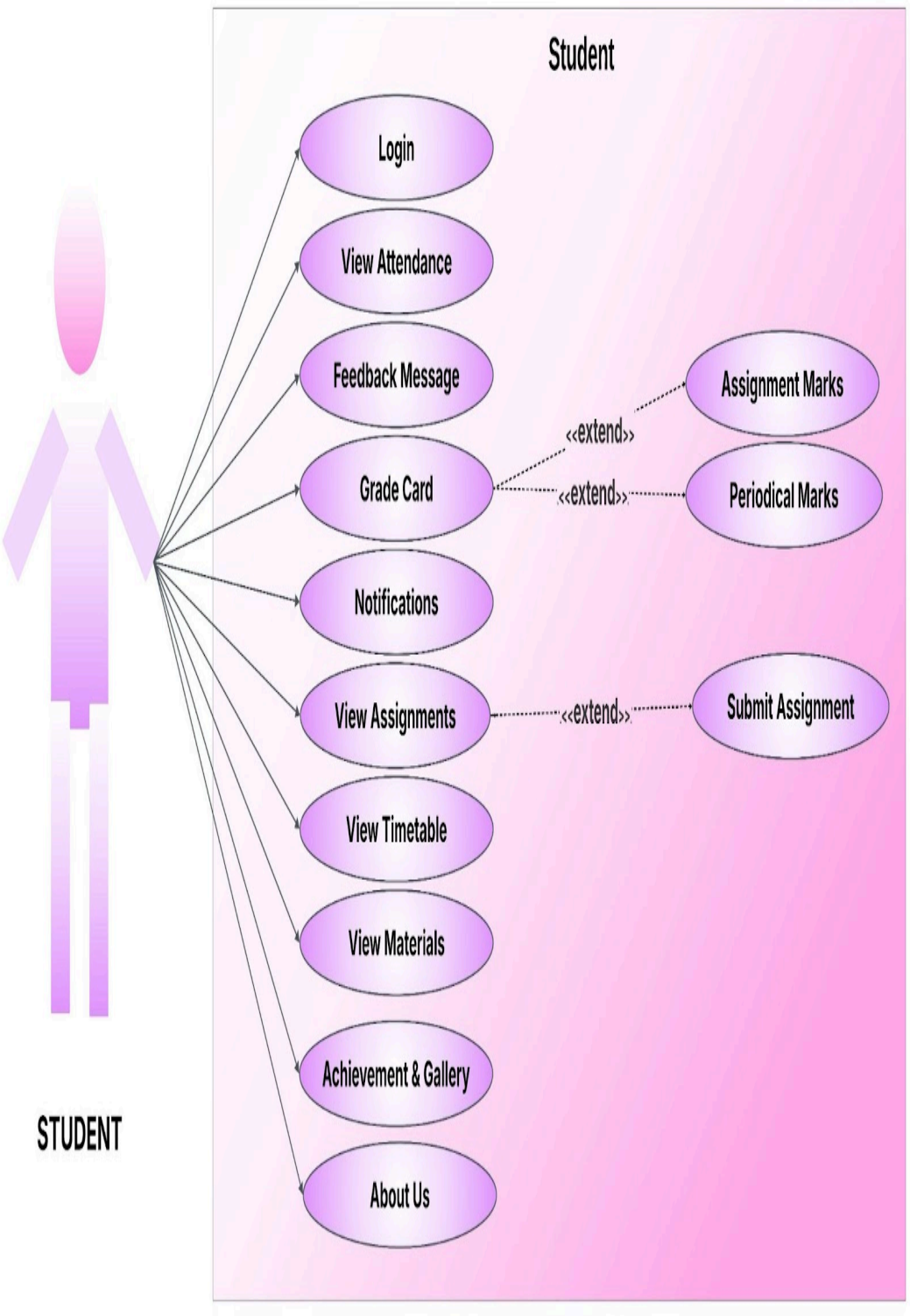
- **Use Case Admin**



- **Use Case Teacher**



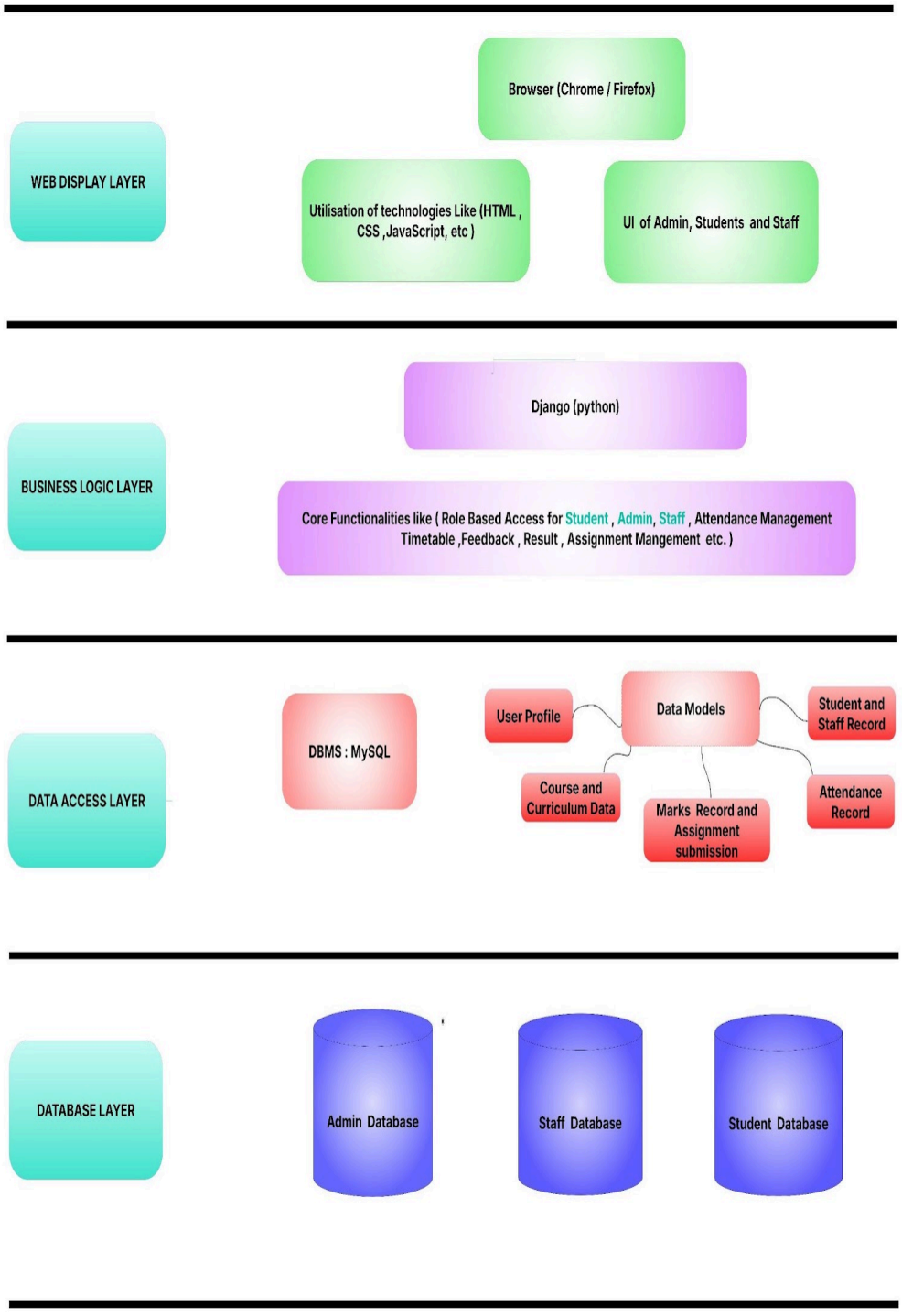
- **Use Case Student**



SYSTEM DESIGN

(SDS)

HIGH LEVEL DESIGN



ER DIAGRAM


customers
id BIGINT
password VARCHAR(128)
last_login DATETIME(6)
username VARCHAR(150)
first_name VARCHAR(150)
last_name VARCHAR(150)
email VARCHAR(254)
is_staff TINYINT(1)
is_active TINYINT(1)
date_joined DATETIME(6)
user_type VARCHAR(10)
students
id INT
admin_id BIGINT
full_name VARCHAR(200)
profile_pic VARCHAR(100)
address LONGTEXT
created_at DATETIME(6)
updated_at DATETIME(6)
session_year_id INT
course_id INT
branch_id INT
staffs
id INT
admin_id BIGINT
gender VARCHAR(15)
teacher_id VARCHAR(20)
address LONGTEXT
created_at DATETIME(6)
updated_at DATETIME(6)
profile_pic VARCHAR(100)
courses
id INT
course_name VARCHAR(255)
created_at DATETIME(6)
updated_at DATETIME(6)
attendance
id INT
subject_id INT
attendance_date DATE
created_at DATETIME(6)
session_year_id INT
class_type VARCHAR(10)
updated_at DATETIME(6)
labs
id INT
lab_id VARCHAR(50)
lab_name VARCHAR(255)
subject_id INT
created_at DATETIME(6)
branch
id INT
name VARCHAR(255)
course_id INT
created_at DATETIME(6)
subjects
id INT
subject_name VARCHAR(255)
subject_code VARCHAR(100)
created_at DATETIME(6)
updated_at DATETIME(6)
staff_id BIGINT
course_id INT
branch_id INT
has_lab TINYINT(1)
notification_student
id INT
student_id INT
message LONGTEXT
created_at DATETIME(6)
updated_at DATETIME(6)
feedback_staffs
id INT
staff_id INT
feedback LONGTEXT
feedback_reply LONGTEXT
created_at DATETIME(6)
updated_at DATETIME(6)
attendance_report
id INT
student_id INT
attendance_id INT
status TINYINT(1)
class_type VARCHAR(10)
created_at DATETIME(6)
updated_at DATETIME(6)
feedback_student
id INT
student_id INT
feedback LONGTEXT
feedback_reply LONGTEXT
created_at DATETIME(6)
updated_at DATETIME(6)
admin
id INT
admin_id BIGINT
created_at DATETIME(6)
updated_at DATETIME(6)
session_years
id INT
session_start_year DATE
session_end_year DATE
notification_staffs
id INT
staff_id INT
message LONGTEXT
created_at DATETIME(6)
updated_at DATETIME(6)
syllabus
id INT
title VARCHAR(255)
subject_id INT
uploaded_by INT
file VARCHAR(255)
uploaded_at DATETIME(6)
timetable
id INT
subject_id INT
teacher_id INT
day_of_week VARCHAR(10)
start_time TIME
end_time TIME
student_result
id INT
student_id INT
subject_id INT
assignment1_marks FLOAT
assignment2_marks FLOAT
periodical1_marks FLOAT
periodical2_marks FLOAT
created_at DATETIME(6)
updated_at DATETIME(6)
assignment
id INT
title VARCHAR(255)
description LONGTEXT
subject_id INT
staff_id INT
due_date DATETIME(6)
assignment_file VARCHAR(255)
created_at DATETIME(6)
updated_at DATETIME(6)
assignment_submission
id INT
assignment_id INT
student_id INT
submission_file VARCHAR(255)
submitted_at DATETIME(6)
status VARCHAR(20)
feedback LONGTEXT
marks_obtained FLOAT
faq
id INT
question VARCHAR(255)
answer LONGTEXT
user_type VARCHAR(20)

DATABASE DESIGN

Admin

Field	Type	Description	Constraints
id	Int	Unique ID for each admin record	Primary Key
admin_id	BigInt	Reference to Custom user	Foreign Key
created_at	DateTime(6)	Timestamp when admin was created	Not Null
updated_at	DateTime(6)	Timestamp when admin was updated	Not Null

Staff

Field	Type	Description	Constraints
id	Int	Unique ID for each staff record	Primary Key
admin_id	BigInt	Reference to Custom user	Foreign Key
teacher_id	Varchar(20)	Unique teacher ID	Default: "Not Assigned"
gender	Varchar(15)	Gender of the staff	Choices: Male, Female, Not Assigned
address	LongText	Address of the staff	Not Null
profile_pic	Varchar(100)	Profile picture of the staff	Nullable, Optional
created_at	DateTime(6)	Timestamp when admin was created	Not Null
updated_at	DateTime(6)	Timestamp when admin was updated	Not Null

Session Year

Field	Type	Description	Constraints
id	Int	Unique ID for each session year	Primary Key
session_start_year	Date	Start date of the session year	Not Null
session_end_year	Date	End date of the session year	Not Null

Custom User

Field	Type	Description	Constraints
id	BigInt	Unique ID for each user	Primary Key
username	Varchar(150)	User's username	Unique, Not Null
email	Varchar(254)	User's email	Unique, Not Null
password	Varchar(128)	Hashed password	Not Null
first_name	Varchar(150)	User's first name	Not Null
last_name	Varchar(150)	User's last name	Not Null
is_staff	TinyInt(1)	Whether user is staff	Default: False
is_active	TinyInt(1)	Whether user is active	Default: True
is_superuser	TinyInt(1)	Superuser status	Default: False
date_joined	DateTime	Timestamp when account was created	Not Null
user_type	Varchar(10)	Defines user type (Admin, Staff, Student)	Choices: (1: Admin, 2: Staff, 3: Student), Default: 1

Courses

Field	Type	Description	Constraints
id	Int	Unique ID for each course	Primary Key
course_name	Varchar(255)	Name of the course	Not Null
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null

Branch

Field	Type	Description	Constraints
id	Int	Unique ID for each branch	Primary Key
name	Varchar(255)	Name of the branch	Not Null
course_id	Int	Reference to Courses	Foreign Key , Default: 1
created_at	DateTime(6)	DateTimeField	Not Null

Labs

Field	Type	Description	Constraints
id	Int	Unique ID for each lab	Primary Key
lab_id	Varchar(50)	Unique Lab ID	Unique, Not Null
lab_name	Varchar(255)	Name of the lab	Not Null
subject_id	Int	Reference to Subjects	Foreign Key
created_at	DateTime(6)	Timestamp when created	Not Null

Subjects

Field	Type	Description	Constraints
id	Int	Unique ID for each subject	Primary Key
subject_name	Varchar(255)	Name of the subject	Not Null
subject_code	Varchar(50)	Unique subject code	Unique, Default: ""
course_id	Int	Reference to Courses	Foreign Key
branch_id	Int	Reference to Branch	Foreign Key
staff_id	BigInt	Reference to Custom user	Foreign Key
has_lab	TinyInt(1)	Whether the subject has a lab	Default: False, Not Null
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null

FAQ

Field	Type	Description	Constraints
id	Int	Unique ID for FAQ	Primary Key
question	Varchar(255)	Frequently asked question	Unique, Not Null
answer	LongText	Answer to the question	Not Null
user_type	Varchar(20)	User type (Admin, Staff, Student)	Choices: Admin, Staff, Student

Students

Field	Type	Description	Constraints
id	Int	Unique ID for each student	Primary Key
admin_id	BigInt	Reference to Custom User	Foreign Key
roll_no	Varchar(20)	Unique roll number	Unique, Default: "Not Assigned"
btbt_id	Varchar(30)	Unique BTBT ID	Unique, Default: "Not Assigned"
profile_pic	Varchar(100)	Profile picture	Nullable, Optional
address	LongText	Address of student	Not Null
course_id	Int	Reference to Courses	Foreign Key
session_year_id	Int	Reference to SessionYearModel	Foreign Key
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null
branch_id	Int	Reference to Branch	Foreign Key,Nullable, Optional

Attendance

Field	Type	Description	Constraints
id	Int	Unique ID for each record	Primary Key
subject_id	Int	Reference to Subjects	Foreign Key
attendance_date	Date	Date of attendance	Not Null
session_year_id	Int	Reference to SessionYearModel	Foreign Key
class_type	Varchar(10)	Type of class (Theory/Lab)	Choices: Theory, Lab, Default: Theory
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null

Attendance Report

Field	Type	Description	Constraints
id	Int	Unique ID for each report	Primary Key
student_id	Int	Reference to Students	Foreign Key
attendance_id	Int	Reference to Attendance	Foreign Key
status	TinyInt	Attendance status	Default: False
class_type	Varchar(10)	Type of class (Theory/Lab)	Choices: Theory, Lab, Default: Theory
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null

Student Feedback

Field	Type	Description	Constraints
id	Int	Unique ID for each feedback	Primary Key
student_id	Int	Reference to Students	Foreign Key
feedback	LongText	Student feedback message	Not Null
feedback_reply	LongText	Reply to feedback	Not Null
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null

Staff Feedback

Field	Type	Description	Constraints
id	Int	Unique ID for each feedback	Primary Key
staff_id	Int	Reference to Staffs	Foreign Key
feedback	LongText	Staff feedback message	Not Null
feedback_reply	LongText	Reply to feedback	Not Null
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null

Student Notification

Field	Type	Description	Constraints
id	Int	Unique ID for each notification	Primary Key
student_id	Int	Reference to Students	Foreign Key
message	LongText	Notification message	Not Null
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null

Staff Notification

Field	Type	Description	Constraints
id	Int	Unique ID for each notification	Primary Key
staff_id	Int	Reference to Staffs	Foreign Key
message	LongText	Notification message	Not Null
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null

Student Results

Field	Type	Description	Constraints
id	Int	Unique ID for each record	Primary Key
student_id	Int	Reference to Student	Foreign Key
subject_id	Int	Reference to Subjects	Foreign Key
assignment1_marks	Double	Marks for Assignment 1	Range: 0-20, Nullable, Default: None
assignment2_marks	Double	Marks for Assignment 2	Range: 0-20, Nullable, Default: None
periodical1_marks	Double	Marks for Periodical 1	Range: 0-20, Nullable, Default: None
periodical2_marks	Double	Marks for Periodical 2	Range: 0-20, Nullable, Default: None
created_at	DateTime(6)	Timestamp when created	Not Null
updated_at	DateTime(6)	Timestamp when updated	Not Null

Assignment

Field	Type	Description	Constraints
id	AutoField	Unique ID for each assignment	Primary Key
title	Varchar(255)	Assignment title	Not Null
description	TextField	Assignment details	Not Null
subject	ForeignKey	Reference to Subjects	Foreign Key
staff	ForeignKey	Reference to Staffs	Foreign Key
due_date	DateTimeField	Assignment due date	Not Null
assignment_file	FileField	Assignment file (PDF)	Nullable, Optional, PDF Validator
created_at	DateTimeField	Timestamp when created	Not Null
updated_at	DateTimeField	Timestamp when updated	Not Null

Assignment Submission

Field	Type	Description	Constraints
id	Int	Unique ID for each submission	Primary Key
assignment_id	Int	Reference to Assignment	Foreign Key
student_id	Int	Reference to Students	Foreign Key
submission_file	Varchar(100)	Submitted file (PDF)	Not Null, PDF Validator
submitted_at	DateTime(6)	Submission timestamp	Not Null
status	Varchar(20)	Submission status	Choices: Submitted, Graded, Late

Timetable

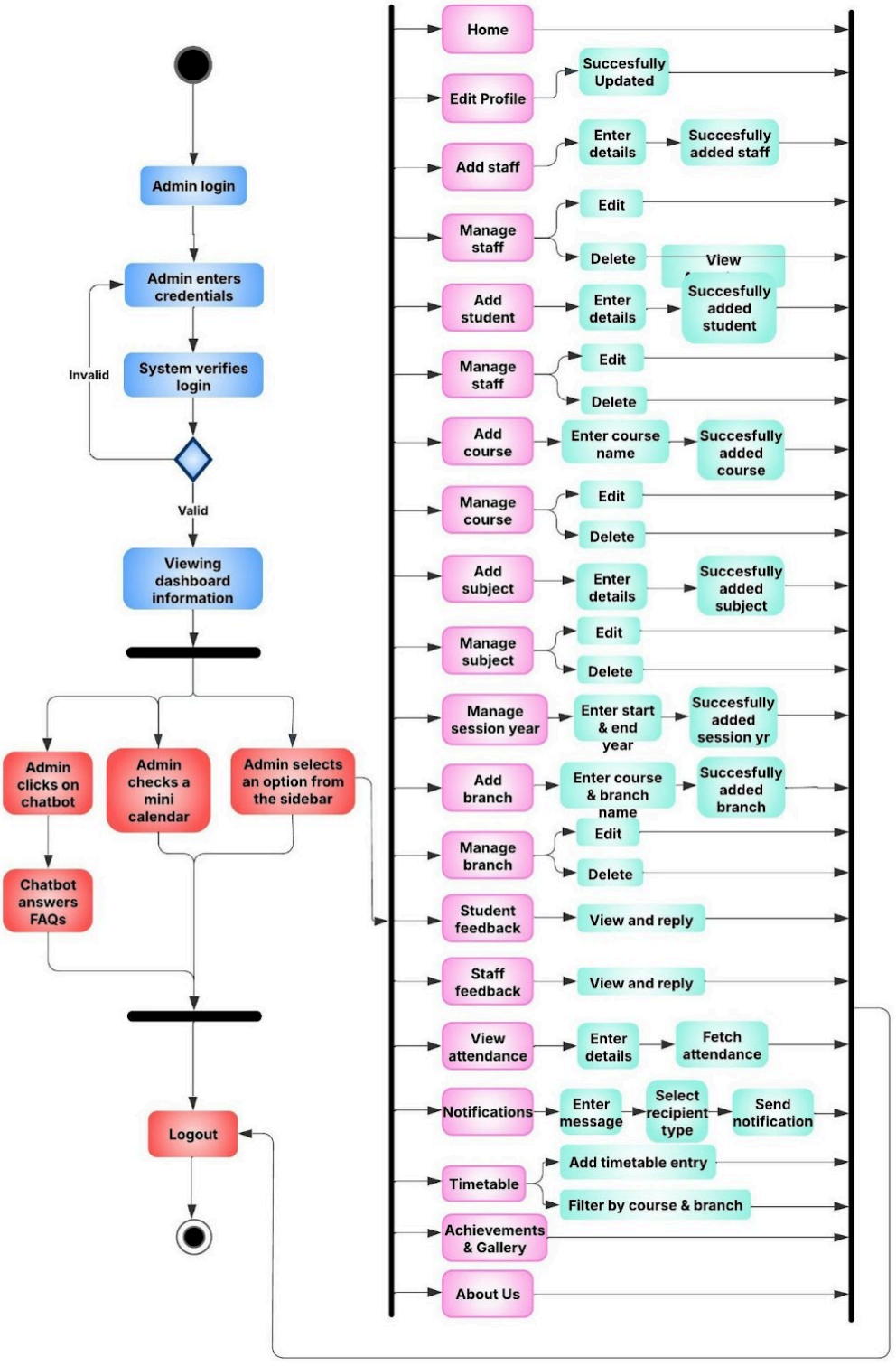
Field	Type	Description	Constraints
id	Int	Unique ID for timetable	Primary Key
subject_id	Int	Reference to Subjects	Foreign Key , Nullable
course_id	Int	Reference to Courses	Foreign Key
teacher_id	Int	Reference to Staffs	Foreign Key
day_of_week	Varchar(10)	Day of the class	Choices: Monday-Sunday
start_time	TimeField	Class start time	Not Null
start_time	TimeField	Class end time	Not Null

Syllabus

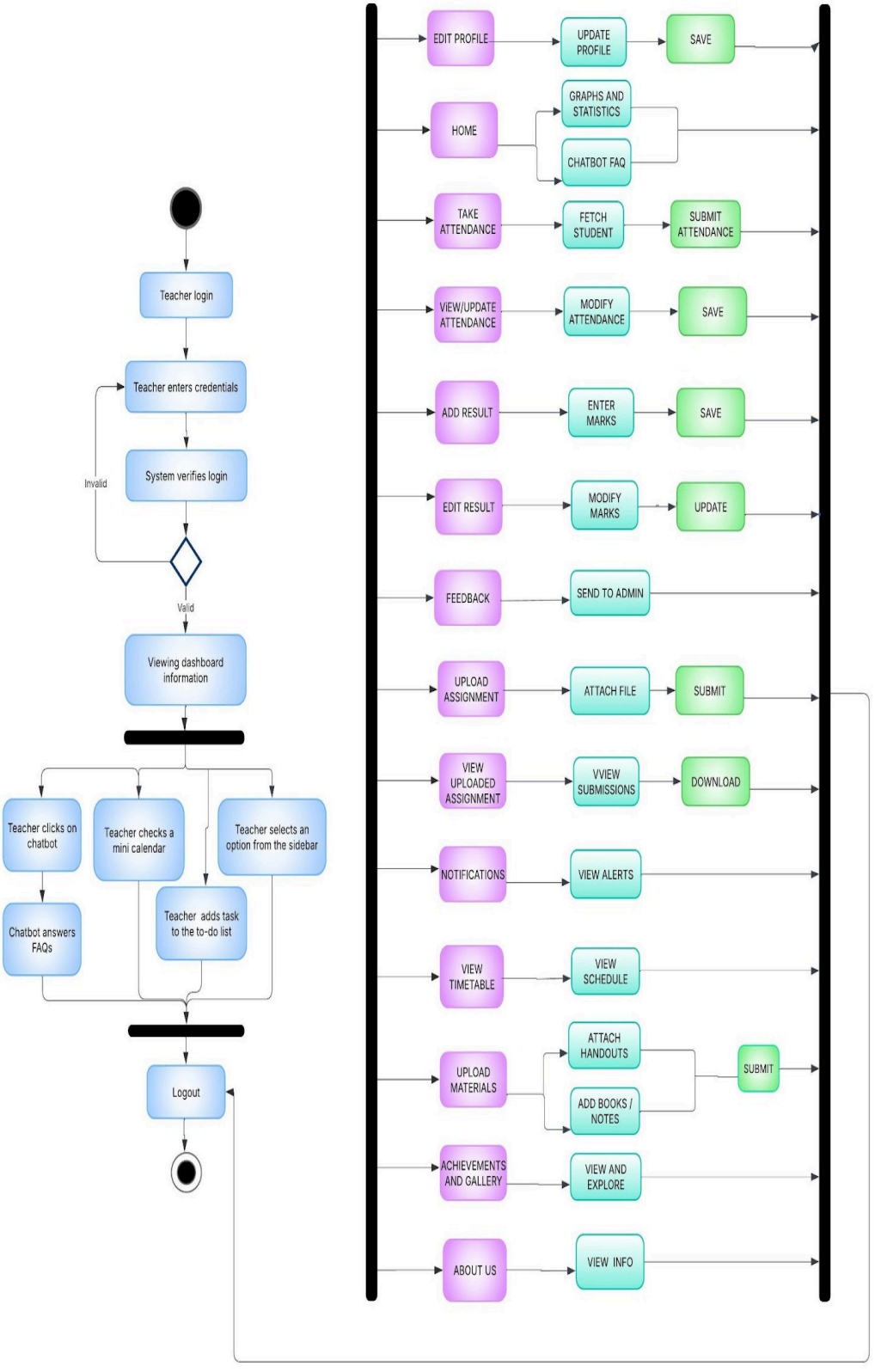
Field	Type	Description	Constraints
id	Int	Unique ID for syllabus	Primary Key
title	Varchar(255)	Syllabus title	Not Null
subject_id	Int	Reference to Subjects	Foreign Key
uploaded_by_id	Int	Reference to Staffs	Foreign Key
file	Varchar(100)	Syllabus file (PDF)	Not Null, PDF Validator
uploaded_at	DateTime(6)	Upload timestamp	Not Null

ACTIVITY DIAGRAMS

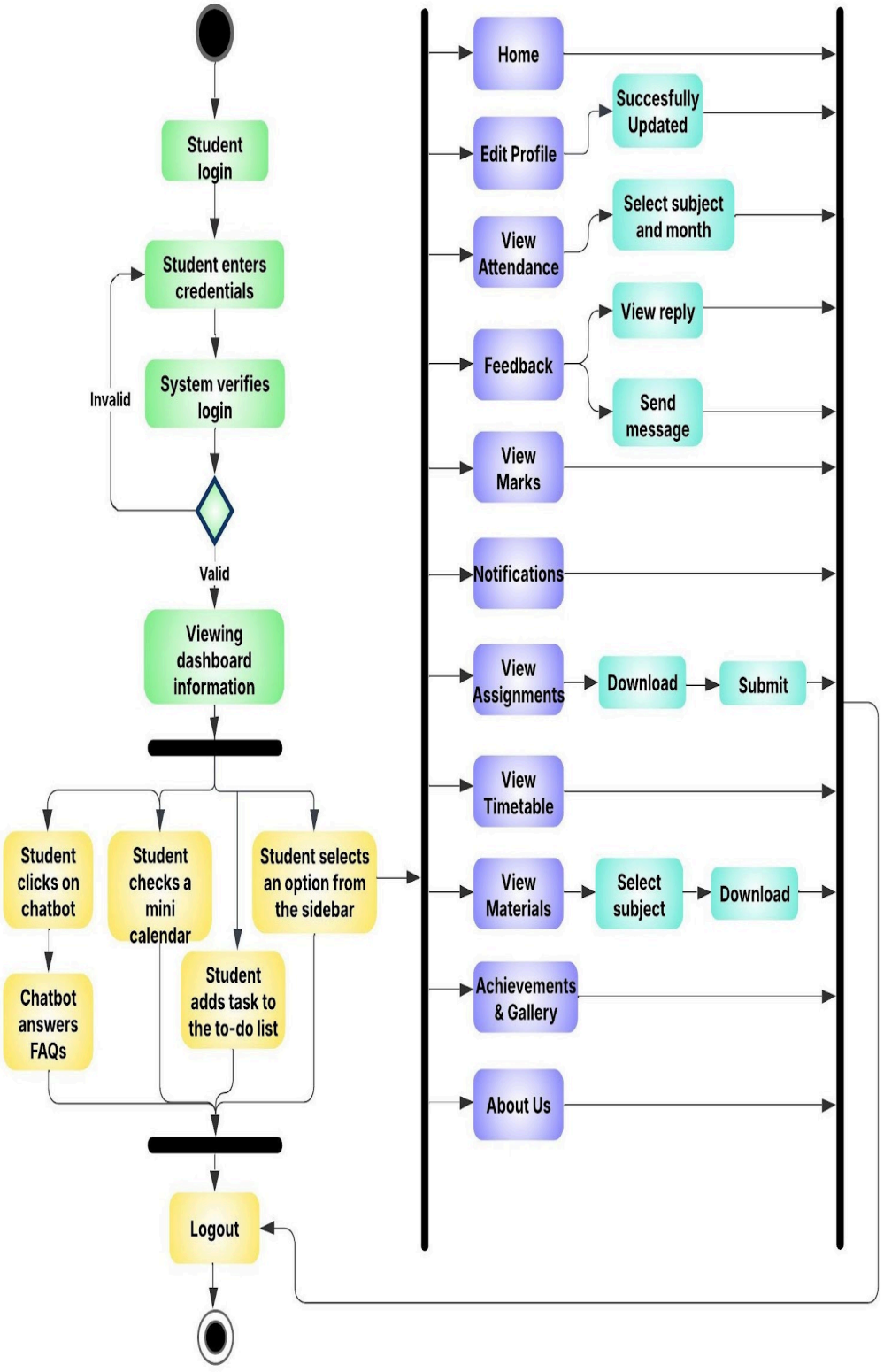
- **Admin Activity Diagram**



- **Teacher Activity Diagram**

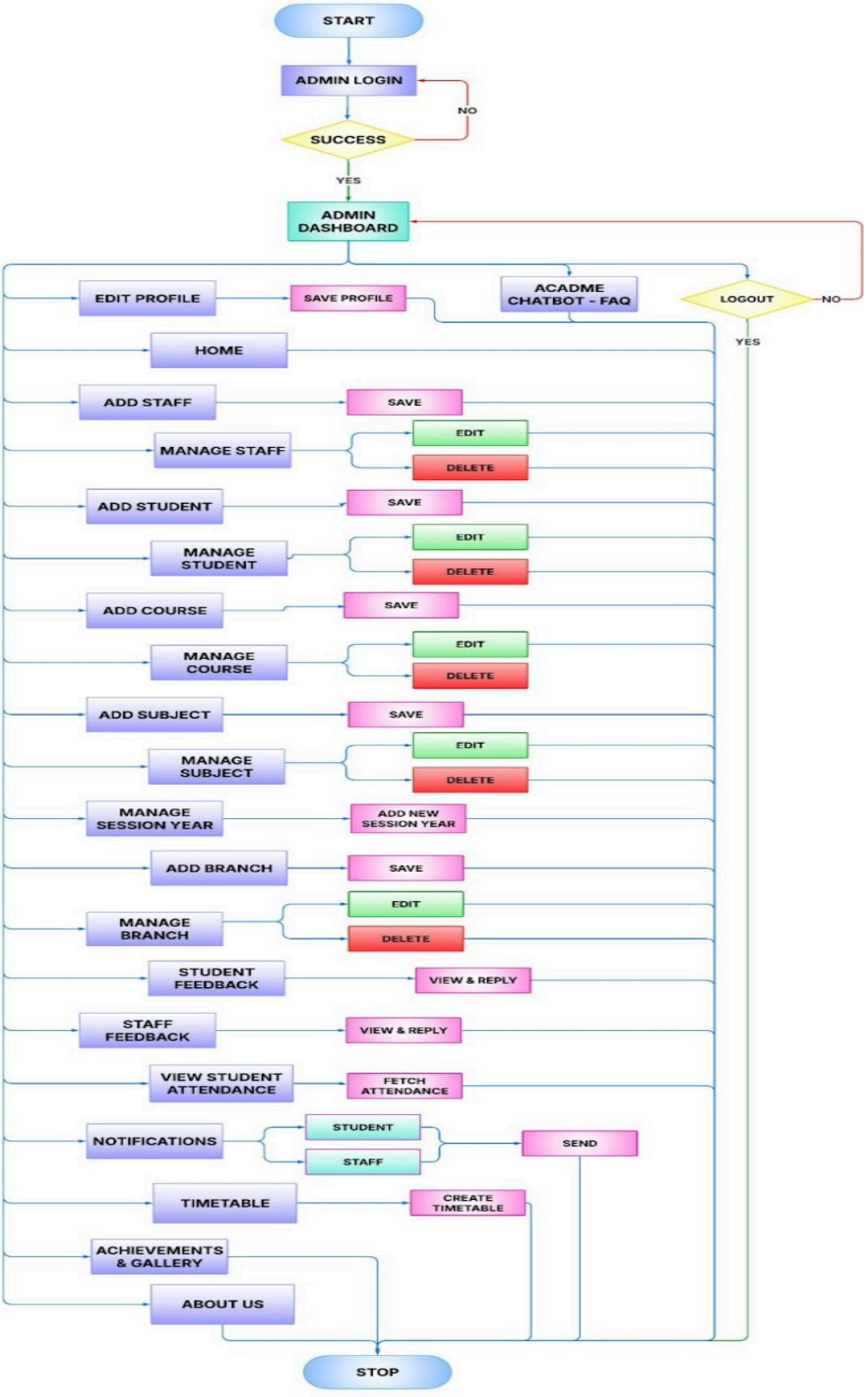


- **Student Activity Diagram**

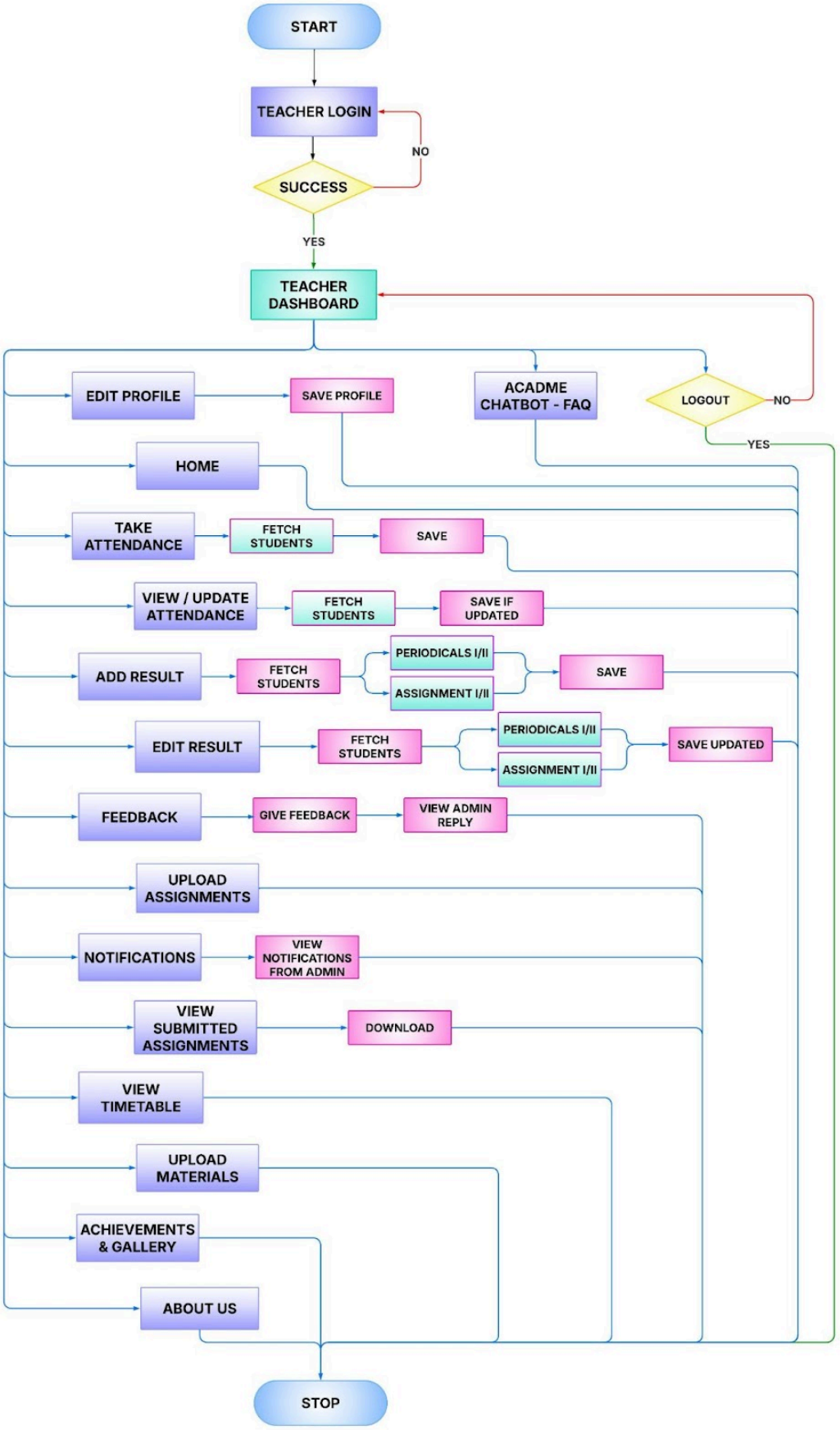


FLOWCHARTS

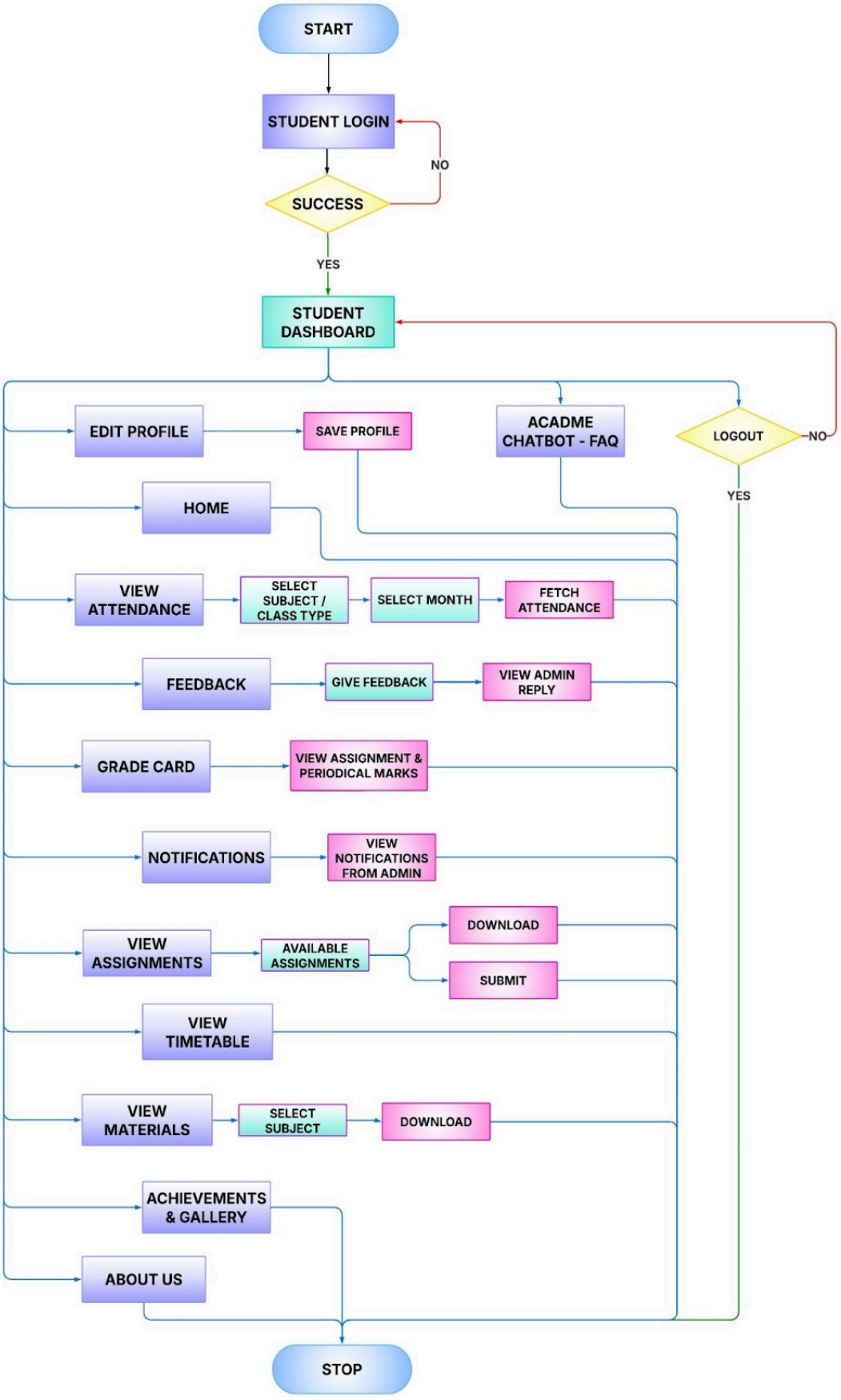
- **Admin Flowchart**



- **Teacher Flowchart**



- **Student Flowchart**



CODING

settings.py

```
DATABASES = {
   'default': {
       'ENGINE': 'django.db.backends.mysql',
       'NAME': 'acadme3',
       'USER': 'acadme4',
       'HOST': '172.19.131.135',
       'PORT': '3306'
   } }
```

Login.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<body>
   {% if messages %}
       <div class="alert alert-danger" id="error-message">
           {% for message in messages %}
               {{ message }}
           {% endfor %}
       </div>
   {% endif %}
   <div class="tagline">
       <span id="word1">Welcome</span>
       <span id="word2">to the</span>
       <span id="word3">AcadME Portal</span>
   </div>

   <div class="login-container">
       <img src="{% static 'assests/img/BV_logo.png' %}"
class="logo" alt="Logo">
       <h3 class="login-title">Sign in with email</h3>
       <form action="/doLogin" method="post">
           {% csrf_token %}
           <div class="form-group">
               <i class="fas fa-envelope"></i>
               <input class="form-control" type="text"
placeholder="Email" name="email">
           </div>
```

```
            <div class="form-group">
                <i class="fas fa-lock"></i>
                <input class="form-control" type="password"
id="password" placeholder="Password" name="password">
                <i class="fas fa-eye-slash toggle-password"
id="toggle-password"></i>
            </div>
            <button class="login-btn" type="submit">Login</button>
        </form>
    </div>
<script>
particlesJS("particles-js", {
    particles: {
        number: { value: 50 },
        shape: { type: "circle" },
        opacity: { value: 0.3 },
        size: { value: 3 },
        move: { enable: true, speed: 2 }
    }
});


document.getElementById('toggle-password').addEventListener('clic
k', function(event) {
        event.stopPropagation();
        let passwordField = document.getElementById('password');
        let type = passwordField.getAttribute('type') ===
'password' ? 'text' : 'password';
        passwordField.setAttribute('type', type);
        this.classList.toggle('fa-eye-slash');
        this.classList.toggle('fa-eye');
});
</script>
</body>
</html>
```

AdminViews.py

```
def add_staff_save(request):
    if request.method!="POST":
        return HttpResponse("Method Not Allowed")
    else:
        first_name=request.POST.get("first_name")
```

```
        last_name=request.POST.get("last_name")
        username=request.POST.get("username")
        teacher_id=request.POST.get("teacher_id")
        email=request.POST.get("email")
        password=request.POST.get("password")
        address=request.POST.get("address")
        gender = request.POST.get("gender")
        profile_pic = request.FILES.get('profile_pic', None)
        if profile_pic:
            fs = FileSystemStorage(location=settings.MEDIA_ROOT)
            filename = fs.save(profile_pic.name, profile_pic)
            profile_pic_url = fs.url(filename)
        else:
            profile_pic_url = None
        try:
user=CustomUser.objects.create_user(username=username,password=pa
ssword,email=email,last_name=last_name,first_name=first_name,user
_type=2)
            user.staffs.address=address
            user.staffs.teacher_id=teacher_id
            user.staffs.gender = gender
            user.staffs.profile_pic = profile_pic_url
            user.save()
            messages.success(request,"Successfully Added Staff")
            return HttpResponseRedirect(reverse("add_staff"))
        except:
            messages.error(request,"Failed to Add Staff")
            return HttpResponseRedirect(reverse("add_staff"))
```

add_staff_template.html

```
{% extends 'admin_template/base_template.html' %}
{% block page_title %}
Add Staff
{% endblock page_title %}
{% block main_content %}
<section class="content">
     <div class="container-fluid">
          <div class="row">
            <div class="col-md-12">
           <div class="card card-primary">
            <div class="card-header">
```

```
              <h3 class="card-title">Add Staff</h3>
            </div>

            <form role="form" action="/add_staff_save"
method="post" enctype="multipart/form-data">
                {% csrf_token %}
              <div class="card-body">
                <div class="form-group">
                  <label>Email address</label>
                  <input type="email" class="form-control"
name="email
" placeholder="Enter email" id="id_email" autocomplete="off">
                </div>
                <div class="form-group">
                  <label>Password</label>
                  <input type="password" class="form-control"
placeholder="Password" name="password">
                </div>
                  <div class="form-group">
                   <label for="gender">Gender:</label>
       <select id="gender" name="gender">
           <option value="Female" selected>Female</option>
           <option value="Male">Male</option>
           <option value="others">others</option>
       </select>
                </div>
                 <div class="form-group">
                  <label>First Name</label>
                  <input type="text" class="form-control"
placeholder="First Name" name="first_name">
                </div>
                   <div class="form-group">
                  <label>Last Name</label>
                  <input type="text" class="form-control"
placeholder="Last Name" name="last_name">
                </div>
                  <div class="form-group">
                  <label>Teacher id</label>
                  <input type="text" class="form-control"
placeholder="enter the teacher id:" name="teacher_id">
                </div>
                 <div class="form-group">
```

```
                    <label>Username</label>
                    <input type="text" class="form-control"
placeholder="Username" name="username" id="id_username"
autocomplete="off">
                </div>
                <div class="form-group">
                  <label>Address</label>
                  <input type="text" class="form-control"
placeholder="Address" name="address">
                </div>
                  <div class="form-group">
                      <label>Profile Picture</label>
                      <input type="file" class="form-control"
name="profile_pic" accept="image/*">
                  </div>
                    <div class="form-group">
                    {% if messages %}
                      {% for message in messages %}
                      {% if message.tags == 'error' %}
                      <div class="alert alert-danger"
style="margin-top:10px">{{ message }}</div>
                      {% endif %}
                      {% if message.tags == 'success' %}
                      <div class="alert alert-success"
style="margin-top:10px">{{ message }}</div>
                      {% endif %}
                      {% endfor %}
                    {% endif %}
                  </div>
              </div>
              <div class="card-footer">
                <button type="submit" class="btn btn-primary
btn-block">Add Staff</button>
              </div>
            </form>
          </div>
        </div>
        </div>
    </div>
  </section>
{% endblock main_content %}
{% block custom_js %}
```

Take Attendance template

```
{% extends 'staff_template/base_template.html' %}
{% block page_title %}
Take Attendance
{% endblock page_title %}
{% block main_content %}
<section class="content">
 <div class="container-fluid">
     <div class="row">
         <div class="col-md-12">
           <div class="card card-primary">
             <div class="card-header">
               <h3 class="card-title">Take Attendance</h3>
             </div>
             <div class="card-body">

                 <div class="form-group">
                       <label>Course</label>
                       <select class="form-control" name="course"
id="course">
                           <option value="">-- Select Course
--</option>
                       </select>
                 </div>
                 <div class="form-group">
                       <label>Branch</label>
                      <select class="form-control" name="branch"
id="branch">
                           <option value="">-- Select Branch
--</option>
                       </select>
                 </div>

                 <div class="form-group">
                       <label>Subject</label>
                       <select class="form-control"
name="subject" id="subject">
                           <option value="">-- Select Subject
--</option>
                       </select>
                 </div>
```

```
                <div class="form-group" id="class_type_block"
style="display: none;">
                    <label>Class Type</label>
                    <select class="form-control"
name="class_type" id="class_type">
                        <option value="theory">Theory</option>
                        <option value="lab">Lab</option>
                    </select>
                </div>

                <div class="form-group">
                  <label>Session Year</label>
                  <select class="form-control"
name="session_year" id="session_year">
                    <option value="">-- Select Session Year
--</option>
              {% for session_year in session_years %}
                     <option value="{{ session_year.id }}">{{
session_year.session_start_year }} to {{
session_year.session_end_year }}</option>
                   {% endfor %}
                  </select>
                </div>
            </div>

            <div class="card-footer">
              <button type="button" class="btn btn-primary
btn-block" id="fetch_student">Fetch Students</button>
            </div>

            <div id="student_data" class="card-footer"></div>
          </div>
        </div>
    </div>
 </div>
</section>
{% endblock main_content %}
{% block custom_js %}
<script>
   $(document).ready(function () {
       $.ajax({
```

```
            url: "/get-teacher-courses/",
            type: "GET",
            dataType: "json",
            success: function (data) {
                let courseDropdown = $("#course");
                courseDropdown.html('<option value="">-- Select
Course --</option>');
                $.each(data, function (key, value) {
                    courseDropdown.append(`<option
value="${value.id}">${value.course_name}</option>`);
                });
            },
            error: function (xhr) {
                console.error("Error fetching courses:",
xhr.responseText);
            }
        });

        // Fetch branches when a course is selected
        $("#course").change(function () {
            let courseId = $(this).val();
            let branchDropdown = $("#branch");
            let subjectDropdown = $("#subject");
            branchDropdown.html('<option value="">-- Select Branch
--</option>');
            subjectDropdown.html('<option value="">-- Select
Subject --</option>');

            if (courseId) {
                $.ajax({
                    url: `/get-branches/${courseId}/`,
                    type: "GET",
                    dataType: "json",
                    success: function (data) {
                        $.each(data, function (key, value) {
                            branchDropdown.append(`<option
value="${value.id}">${value.name}</option>`);
                        });
                    },
                    error: function (xhr) {
                        console.error("Error fetching branches:",
xhr.responseText);
```

```
                    }
                });
            }
        });

        // Fetch subjects when a branch is selected
        $("#branch").change(function () {
            let branchId = $(this).val();
            let subjectDropdown = $("#subject");
            subjectDropdown.html('<option value="">-- Select
Subject --</option>');

            if (branchId) {
                $.ajax({
                    url: `/get-subjects/${branchId}/`,
                    type: "GET",
                    dataType: "json",
                    success: function (data) {
                        $.each(data, function (key, value) {
                            subjectDropdown.append(`<option
value="${value.id}"
data-has-lab="${value.has_lab}">${value.subject_name}</option>`);
                        });
                    },
                    error: function (xhr) {
                        console.error("Error fetching subjects:",
xhr.responseText);
                    }
                });
            }
        });

        $("#subject").change(function () {
            let hasLab = $("#subject
option:selected").data("has-lab");
            if (hasLab) {
                $("#class_type_block").show();
            } else {
                $("#class_type_block").hide();
            }
        });
```

```
        // Fetch Students (including branch in the request)
        $("#fetch_student").click(function () {
            let subject = $("#subject").val();
            let session_year = $("#session_year").val();
            let class_type = $("#class_type").val();
            let branch = $("#branch").val();  // Added branch
value


            $.ajax({
                url: '{% url "get_students" %}',
                type: "POST",
                dataType: "json", // jQuery will parse JSON
automatically.
                data: {subject: subject, session_year:
session_year, class_type: class_type, branch: branch},
                headers: { "X-CSRFToken": "{{ csrf_token }}" }
            })
            .done(function (response) {
                let json_data = response;
                let div_data = "<div
class='form-group'><label>Attendance Date:</label><input
type='date' name='attendance_date' id='attendance_date'
class='form-control'></div>";
                div_data += "<div class='table-responsive'><table
class='table table-bordered'>";
                div_data += "<thead
class='thead-dark'><tr><th>Roll No</th><th>Name</th><th>BTBT
ID</th><th>Present</th></tr></thead><tbody>";


                $.each(json_data, function (key, student) {
                    div_data += `<tr>
                        <td>${student.roll_no}</td>
                        <td>${student.name}</td>
                        <td>${student.btbt_id}</td>
                        <td class='text-center'><input
type='checkbox' checked='checked' name='student_data[]'
value='${student.id}'></td>
                    </tr>`;
                });


                div_data += "</tbody></table></div>";
                div_data += "<div class='form-group
text-center'>";
```

```
                div_data += "<button id='save_attendance'
class='btn btn-success' type='button'><i class='fas fa-save'></i>
Save Attendance</button>";
                div_data += "</div>";

                $("#student_data").html(div_data);
            })
            .fail(function () {
                alert("Error in Fetching Students");
            });
        });

        // Save Attendance
        $(document).on("click", "#save_attendance", function () {
            $(this).attr("disabled", "disabled").text("Saving
Attendance...");
            let student_data =
$("input[name='student_data[]']").map(function () {
                return {id: $(this).val(), status:
$(this).is(":checked") ? 1 : 0};
            }).get();

            let attendance_date = $("#attendance_date").val();
            let subject_id = $("#subject").val();
            let session_year_id = $("#session_year").val();
            let class_type = $("#class_type").val();

            $.ajax({
                url: '{% url "save_attendance_data" %}',
                type: "POST",
                data: {student_ids: JSON.stringify(student_data),
attendance_date: attendance_date, subject_id: subject_id,
session_year_id: session_year_id, class_type: class_type},
                headers: { "X-CSRFToken": "{{ csrf_token }}" }
            })
            .done(function (response) {
                if (response.includes("ERROR")) {
            alert("" + response);
        } else {
            alert(" Attendance Saved Successfully!");
            location.reload();
        }
```

```
    })
    .fail(function (xhr) {
        alert("" + xhr.responseText);
    });
        });
    });
</script>
{% endblock custom_js %}
```

```
@csrf_exempt
def staff_take_attendance(request):
    subjects =
Subjects.objects.filter(staff_id=request.user.id).values("id",
"subject_name", "has_lab")
    session_years=SessionYearModel.object.all()
    return
render(request,"staff_template/staff_take_attendance.html",{"subj
ects":subjects,"session_years":session_years})
```

```
@csrf_exempt
def save_attendance_data(request):
    student_ids=request.POST.get("student_ids")
    subject_id=request.POST.get("subject_id")
    attendance_date= request.POST.get("attendance_date")
    session_year_id= request.POST.get("session_year_id")
    class_type = request.POST.get("class_type")

    try:
        attendance_date_obj = datetime.strptime(attendance_date,
"%Y-%m-%d").date()
        today_date = datetime.today().date()
        two_days_before = today_date - timedelta(days=2)
        session_model =
SessionYearModel.object.get(id=session_year_id)
        session_start_date = session_model.session_start_year
        session_end_date = session_model.session_end_year

        if not (session_start_date <= attendance_date_obj <=
session_end_date):
            return HttpResponse("ERROR: Attendance date must be
within the session year range.", status=400)
        if not (two_days_before <= attendance_date_obj <=
today_date):
```

```
            return HttpResponse("ERROR: Attendance can only be
taken for today or the past two days.", status=400)
        subject_model = Subjects.objects.get(id=subject_id)

        existing_attendance = Attendance.objects.filter(
            subject_id=subject_model,
            attendance_date=attendance_date,
            session_year_id=session_model,
            class_type=class_type
        ).exists()

        if existing_attendance:
            return HttpResponse("ERROR: Attendance already marked
for this date.", status=400)
        json_sstudent = json.loads(student_ids)
        if class_type not in ["theory", "lab"]:
            return HttpResponse("ERROR: Invalid class_type")
attendance=Attendance(subject_id=subject_model,attendance_date=at
tendance_date,session_year_id=session_model,
class_type=class_type)
        attendance.save()
        for stud in json_sstudent:
            student=Students.objects.get(admin=stud['id'])

attendance_report=AttendanceReport(student_id=student,attendance_
id=attendance,status=stud['status'],class_type=class_type)
            attendance_report.save()
        return HttpResponse("OKAY")
    except SessionYearModel.DoesNotExist:
        return HttpResponse("ERROR: Invalid Session Year",
status=400)
    except Subjects.DoesNotExist:
        return HttpResponse("ERROR: Invalid Subject", status=400)
    except Exception as e:
        print(f"Error while saving attendance: {str(e)}")
        return HttpResponse(f"ERROR: {str(e)}")
```

Student(Result)

```
{% extends 'student_template/base_template.html' %}
{% block page_title %}
Result
{% endblock page_title %}
```

```
{% block main_content %}
<section class="content">
     <div class="container-fluid">
         <div class="row">
             <div class="col-md-12">
                <div class="card card-primary">
                 <div class="card-header">
                     <h3 class="card-title">Marks Obtained</h3>
                 </div>
                    <div class="table-responsive">
                       <table class="table table-bordered">
                           <thead class="thead-dark">
                               <tr>
                                    <th>ID</th>
                                    <th>Subject</th>
                                       <th>Assignment 1</th>
                                       <th>Assignment 2</th>
                                       <th>Periodical 1</th>
                                       <th>Periodical 2</th>
                               </tr>
                           </thead>
                       <tbody>
                           {% for row in studentresult %}
                            <tr>
                                <td>{{ row.id }}</td>
                                <td>{{
row.subject_id.subject_name }}</td>
                                <td>{{
row.assignment1_marks|default:"-" }}</td>
                                <td>{{
row.assignment2_marks|default:"-" }}</td>
                                <td>{{
row.periodical1_marks|default:"-" }}</td>
                                <td>{{
row.periodical2_marks|default:"-" }}</td>
                           </tr>
                            {% empty %}
                              <tr><td colspan="9"
class="text-center">No Results Available</td></tr>
                            {% endfor %}
                         </tbody>
                       </table>
```

```
                    </div>
                </div>
            </div>
        </div>
    </div>
    </section>
{% endblock main_content %}
```

StudentViews.py

```
def student_view_result(request):
    student=Students.objects.get(admin=request.user.id)

studentresult=StudentResult.objects.filter(student_id=student.id)
    return
render(request,"student_template/student_result.html",{"studentre
sult":studentresult})
```

TESTING

Testing the AcadMe platform involves multiple types of tests to ensure functionality, performance, security, and usability for students, teachers, and admins.

1. Functional Testing

- **Unit Testing:**
  - Each module (e.g., attendance tracking, timetable management, study material upload, etc) will be tested individually.
  - Example: Testing the function that calculates student attendance percentage.

- **Integration Testing:**
  - Ensuring that different modules interact correctly (e.g., attendance updates reflecting in student dashboards).
  - Example: Verifying that when a teacher marks attendance, the database updates correctly, and students can view it.

- **System Testing:**
  - The entire system will be tested to verify that all functionalities work together.
  - Example: Checking whether the notification system works when admins send updates about exams.

2. Usability Testing

- **UI Testing:**
  - Ensuring the user interface is consistent and user-friendly.
  - Example: Checking if navigation menus work properly across different user roles (Admin, Teacher, Student).

- **UX Testing:**
    - Evaluating the overall user experience, ensuring smooth workflow and accessibility.
    - Example: Checking if students can easily locate and download study materials uploaded by teachers.

3. Compatibility Testing

- **Browser Compatibility:**
    - Ensuring the platform works smoothly on different web browsers (Google Chrome, Edge , Mozilla Firefox).

4. Security Testing

- **Authentication and Authorization Testing:**
    - Ensuring user roles have appropriate access controls (e.g., only admins can create timetables).

- **Data Security Testing:**
    - Checking if user credentials and sensitive data are securely stored.

5. Database Testing

- **Data Integrity Testing:**
    - Ensuring the correctness and consistency of data stored in the MySQL database.
    - Example: Verifying that attendance records remain intact even after multiple updates.

6. Regression Testing

- Re-testing previously working features after updates to ensure no unintended issues arise.

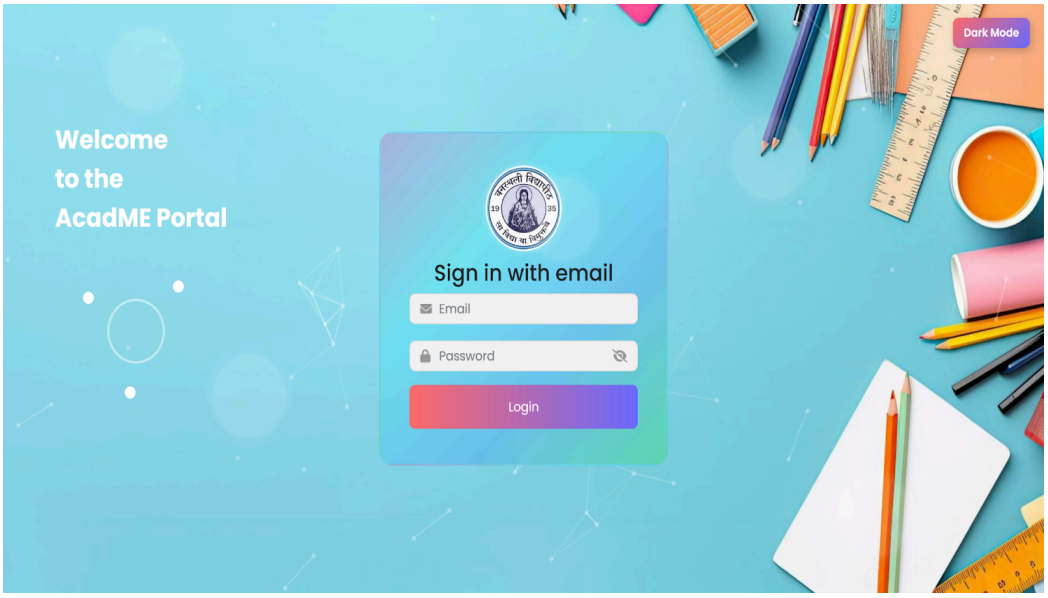
- Example: After updating the results management module, checking that Teacher marks entry and student score displays remain accurate.

7. Scalability Testing

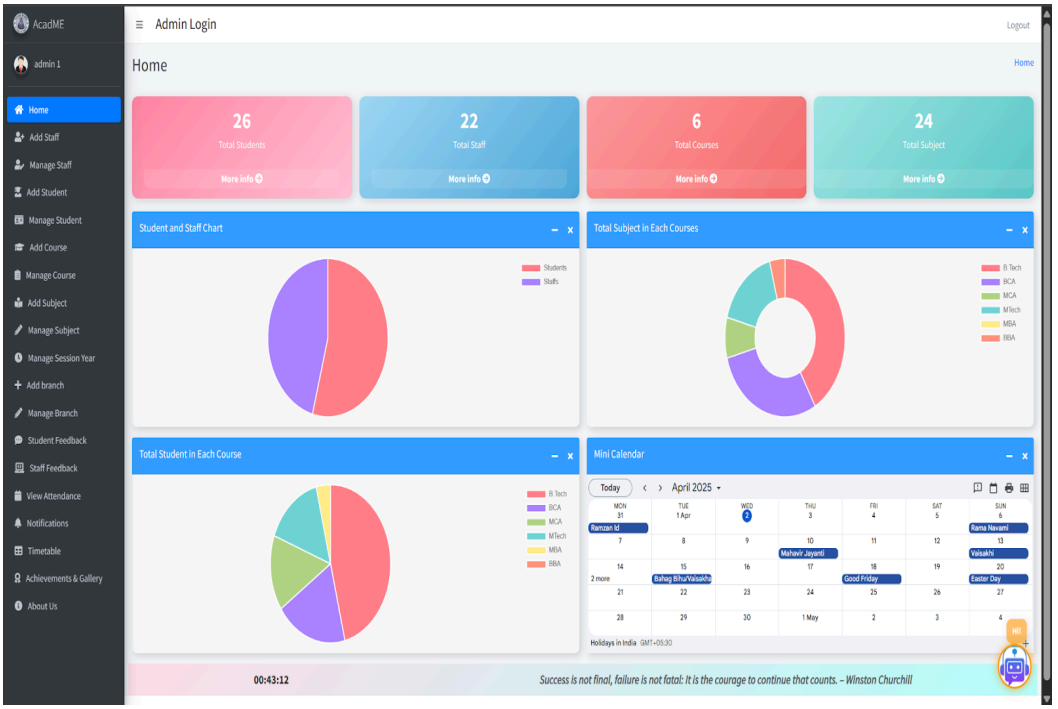
- Ensuring that the platform can handle increasing numbers of users and data over time.
- Example: Simulating system usage during peak periods like admissions or exam result announcements.

USER INTERFACES

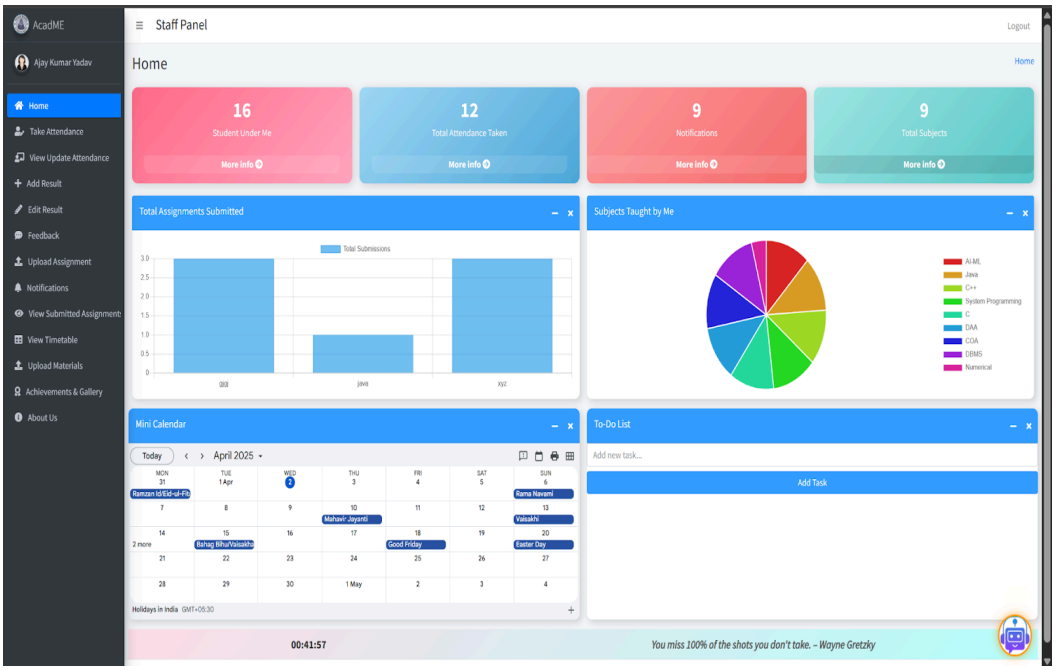
Login Page



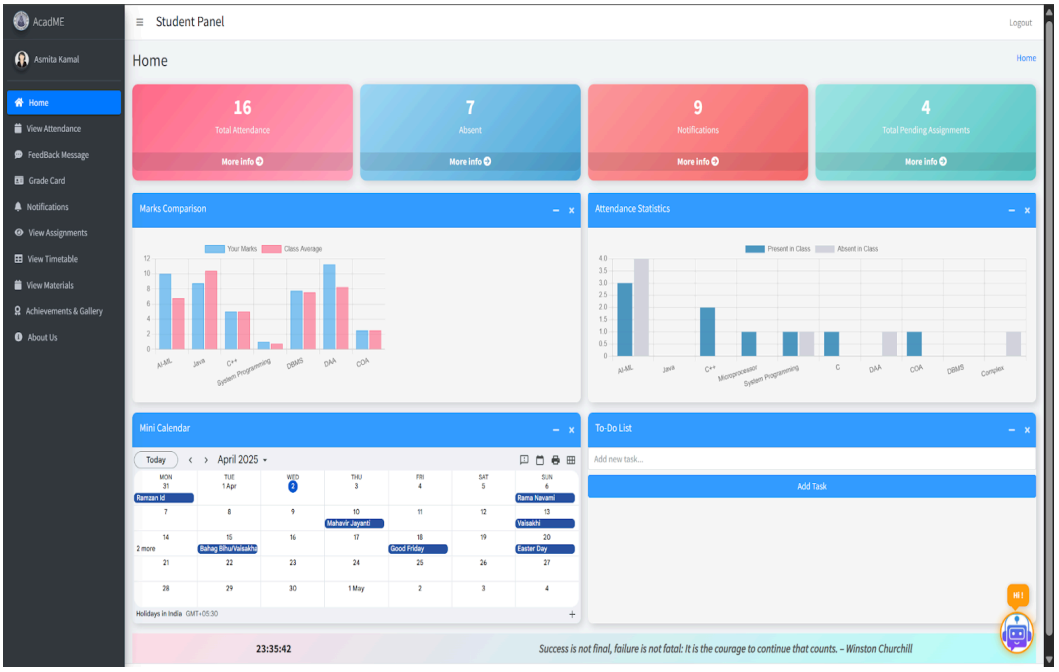
Admin Dashboard



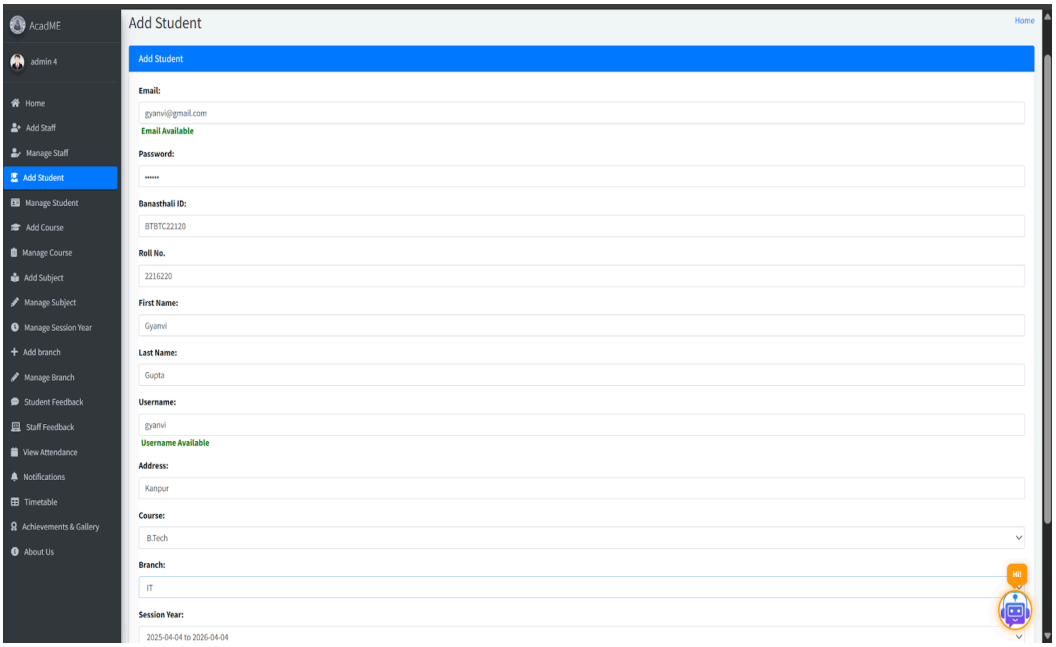
Teacher Dashboard



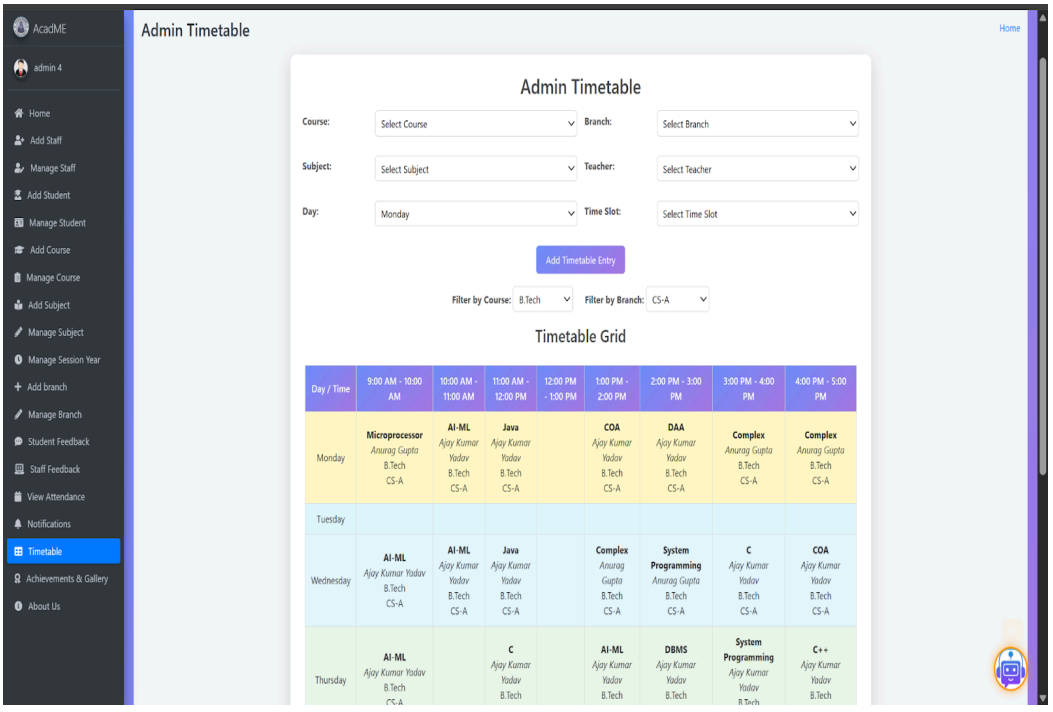
Student Dashboard



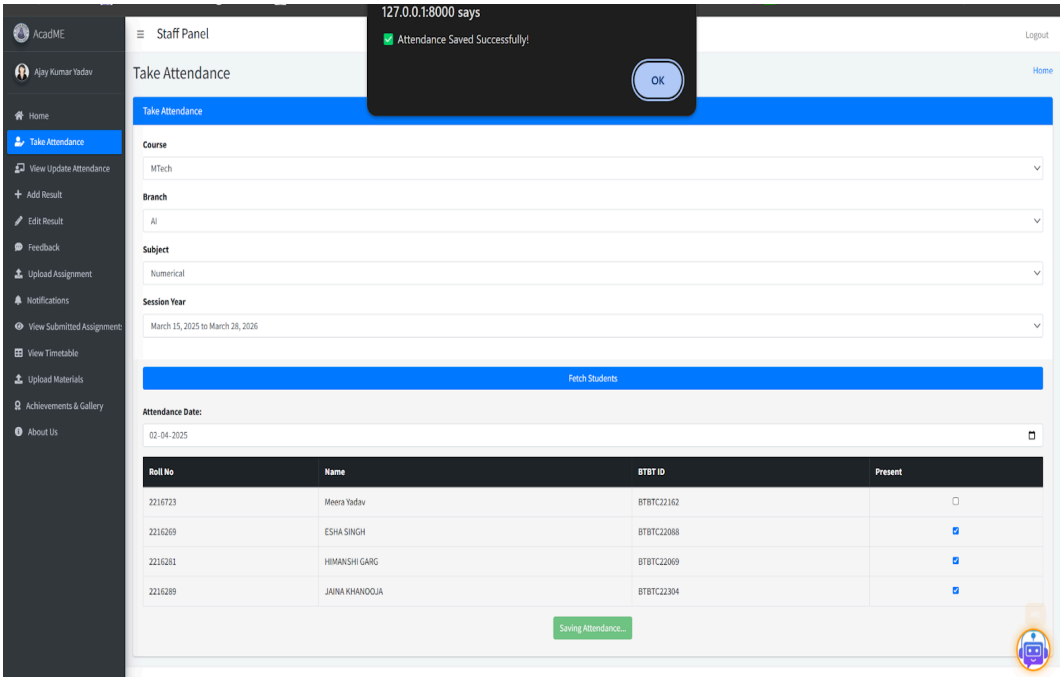
Add student



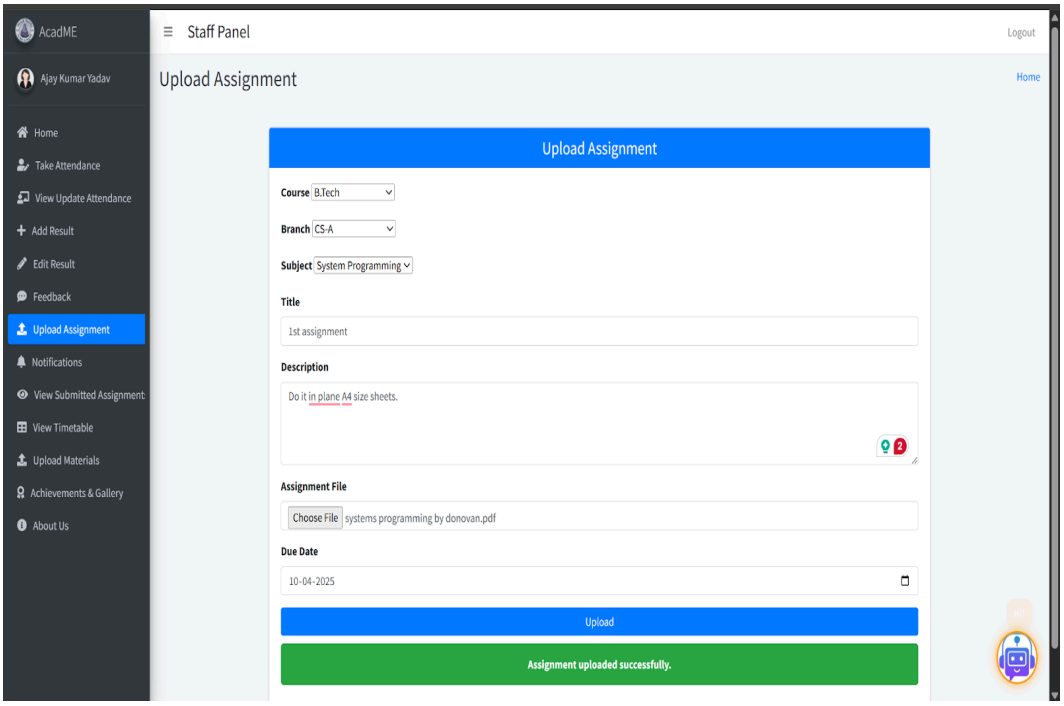
Admin Timetable



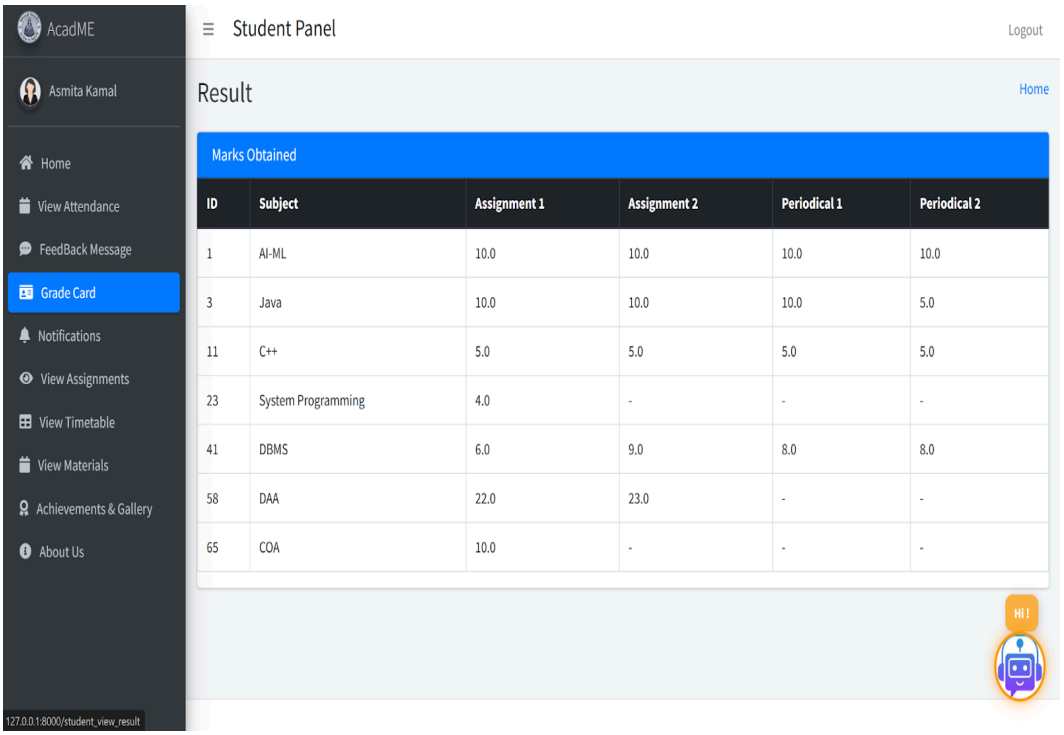
Staff take attendance



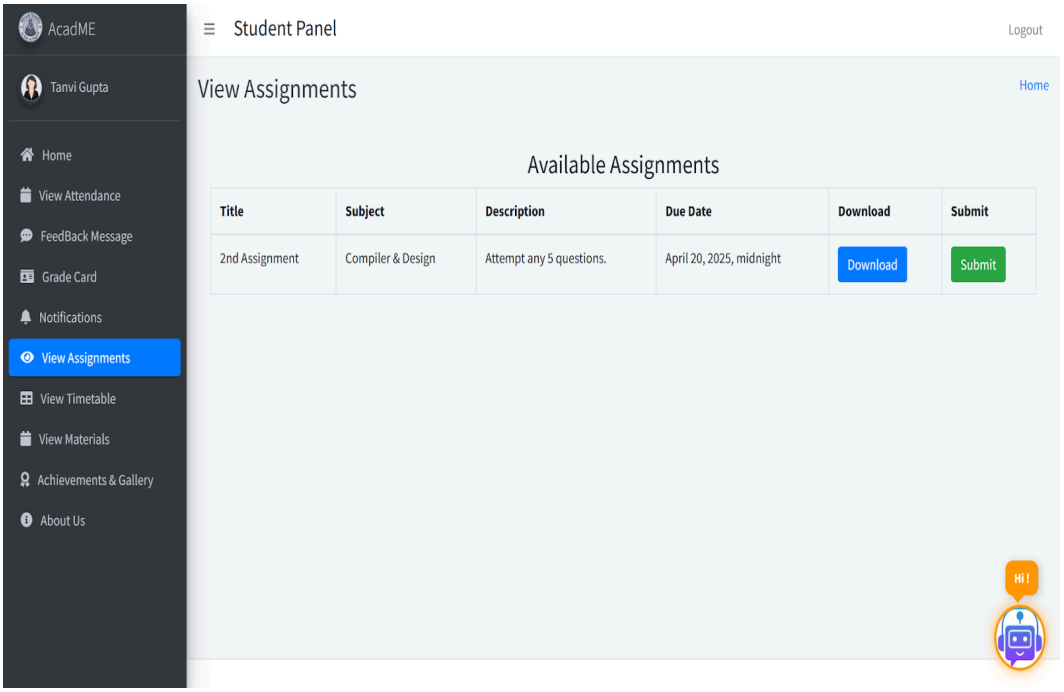
Staff upload assignment



Student View Marks



Student View Assignment



APPENDICES

Development Methodology

- **Requirements Gathering**: Identifying key functionalities like student, teacher, and admin dashboards, attendance tracking, result management, timetable management, feedback handling, etc.
- **System Integration**: Combining frontend (HTML, CSS, JavaScript) with backend (Django, Python, MySQL) for seamless data flow.
- **User Interface Design**: Creating an intuitive, responsive, and user-friendly UI for smooth navigation.
- **Testing & Validation**: Unit, integration, and regression testing to ensure reliability and security.

Technology Stack

- **Frontend**: HTML, CSS, JavaScript
- **Backend**: Python(Django)
- **Database**: MySQL-based relational database

Common Terminologies

- **CRUD**: Create, Read, Update, Delete operations in the database.
- **UI/UX**: User Interface & Experience for smooth navigation.
- **SRS**: Software Requirements Specification.
- **DBMS**: Database Management System.
- **API**: Application Programming Interface for data communication.
- **HTTPS**: Secure communication protocol.

REFERENCES

- K. K. Agarwal, Software Engineering and Testing: An Introduction, Alpha Science International, 2009.

- *Banasthali Vidyapith - Welcome to Banasthali Vidyapith*. (2025). Banasthali.org. http://www.banasthali.org/banasthali/wcms/en/home/

- R. S. Pressman, "Software Engineering: A Practitioner's Approach," 7th ed. McGraw-Hill Education, 2014.

- LucidChart. (2017). *Online Diagram Software & Visual Solution | Lucidchart*. Lucidchart.com. https://www.lucidchart.com/