DTDP DEPARTMENT MANAGEMENT SYSTEM

An Information Technology Project submitted in partial fulfillment of Requirements for the Degree of Bachelor of Technology (DTDP)

By

RADHA KRISHNA DESHPANDE

Regd No : 19011BC038

B.Tech (Digital Techniques for Design and Planning) – VIII Semester

Under the guidance of
ER.S.RAVI KIRAN

Department of Digital Technology

School of Planning and Architecture
Jawaharlal Nehru Architecture and Fine Arts University

Mahaveer Marg, Masab Tank, Hyderabad – 500028
June 2023

JNAFAU SCHOOL OF PLANNING AND ARCHITECTURE

Mahaveer Marg, Masab Tank, Hyderabad – 500028

Department of Digital Technology.

CERTIFICATE

I/We Certify that the IT Project entitled **" DTDP DEPARTMENT MANAGEMENT SYSTEM "** submitted by **MR.RADHA KRISHNA DESHPANDE** bearing Roll No. **19011BC038** on this 5th of JUNE, 2021 in partial fulfillment of the Requirements for the Degree of **BACHELOR OF TECHNOLOGY (DIGITAL TECHNIQUES FOR DESIGN AND PLANNING)** of University is a bonafide work to the best of my/our knowledge and may be placed before the Examination Board for their consideration.

Er.S.Ravi Kiran
Project Guide

M. Sarah Nireekshana
Thesis Co-Ordinator

S. Ravi Kiran
Head of the Department

External Examiner

DECLARATION

I declare that this project report titled "**DTDP DEPARTMENT MANAGEMENT SYSTEM"** is submitted by **MR.RADHA KRISHNA DESHPANDE** bearing Roll No. **19011BC038** in partial fulfillment of the degree of **Bachelor of Technology** in the **Department of Digital Technology** is a record of original work carried out by me under the supervision of **Er.S.RAVI KIRAN** and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Name: ER.S.Ravi Kiran
Designation: Project Guide
Signature:

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Er.S.Ravi Kiran**, Designation and Project Supervisor, Department of Digital Technology, Jawaharlal Nehru Architecture & Fine Arts University, for his able guidance and useful suggestions, which helped us in completing the project in time.

We are particularly thankful to **Er.S.Ravi Kiran**, the Head of the Department, Department of Digital Technology, his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honorable Principal **Ch.Srinivas,** for providing all facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartful thanks to **Dr.N.Kavita Daryani Rao**, Vice-Chancellor, JNAFAU for providing a congenial atmosphere to complete this project successfully.

We also thank all the staff members of Department of Digital Technology for their valuable support and generous advice. Finally thanks to all our friends and family members for their continuous support and enthusiastic help.

Yours Sincerely,

RADHA KRISHNA DESHPANDE

ABSTRACT

The purpose of the DTDP department management system is to automate the existing manual system with the help of computerised equipment and full-fledged computer software, fulfilling their requirements so that the valuable data and information can be stored for a longer period of time with easy access. The system is a web-based application that can be accessed throughout the department. It deals with all kind of student details, academic related reports, course details, curriculum, batch details and other resource related details too. It tracks all the details of a student from the day one to the end of his course which can be used for all reporting purpose, tracking of attendance, progress in the course, completed semesters years, coming semester year curriculum details, project or any other final exam result and all these will be available for future references too.

Our program will have the databases of students details, subjects details, attendance records, marks record and its department details in all aspects. DTDP department management system, as described above, can lead to an error-free, secure, reliable, and fast management system. It can help the user concentrate on their other activities rather than on the record-keeping. This system may be used for monitoring the overall activities as well as performance of the students. This system is being developed to maintain and facilitate easy access to information. For this the users must be registered with the system after which they can access as well as modify data as per the permissions given to them. It is a web based application that aims at providing information to all the levels of department.

This system also contains all other books in the library module which are related to the DTDP department. For a given student/faculty can access the system to either upload or download some information from the database. Thus, it will help department in better utilise their resources.

Keywords—**Admin, Department system, Information, Management System, Student, Faculty**

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The DTDP Department Management system website is a comprehensive and user-friendly platform designed to streamline the management of various aspects within a department. It has been developed to override the problems prevailing in the manual system. It provides administrators, faculty members, and students with a centralized system to efficiently handle tasks related to faculty management, subject management, student management, attendance management, marks management, library management, learning management, alumni management, and thesis management.

This software is supported to eliminate and, in some cases, reduce the hardships faced by this existing system.This work is basically a website which includes attractive designs and proper arrangements of links and images. It can handle all details about a student. The details include college details, subject details, student personnel details, academic details, exam details etc...In case of manual system they need a lot of time, manpower etc.Here almost all work is computerized. So the accuracy is maintained. Maintaining backup is very easy. It can do with in a few minutes.

The website offers a range of functionalities tailored to different user roles. Administrators have access to features such as user authentication, a dashboard for data visualization, and the ability to manage faculty details, subjects, students, attendance records, marks, library resources, learning materials, alumni information, and thesis details. Through an intuitive interface, administrators can effortlessly perform tasks like adding new faculty, assigning subjects to faculty members, managing student records, tracking attendance, managing marks, and maintaining a comprehensive library database.

1.1 Objective

The purpose of the project is to develop a comprehensive and efficient software solution that automates and streamlines various administrative and academic processes for our department, DTDP. The system aims to improve the overall management and coordination of different aspects, enhance communication and collaboration among faculty and students, and provide an intuitive and user-friendly interface for accessing and managing information related to student records, subjects, attendance, grades, assignments, library resources, thesis, etc. This web-oriented application allows us to access all information about the course, faculty, students, facilities, etc. This application provides a virtual tour of DTDP. Here we will get the latest information about the students and faculty. This is totally built at the administrative end, and thus only the administrator is guaranteed access. Here, faculty can mark attendance and grade students. This generic application is designed to assist the students of an institute regarding information on the course, subjects, classes, assignments, marks, results, etc. Students can log into their dashboard to check their attendance and grades. It also provides support in that a faculty member can also check his daily schedule, upload assignments, and send notices to the students. Here, the administrator will manage the accounts of the students and faculties and upload the latest information about the Department.

1.2 Scope

This application typically encompasses a wide range of administrative and operational tasks related to managing the day-to-day operations of the DTDP department. The system is designed to streamline processes, and provide better control and coordination across various functions within the department and. Here are some key areas typically covered in DTDP department management system :

- Student Management: The system manages student information such as assignments, attendance, and results and helps track the progress of individual students throughout their academic journey.

- Faculty Management: It involves maintaining records of faculty.
- Subject Management: It enables administrators to add subjects, manage subject teachers efficiently, and monitor their teaching progress.
- Learning Management: It organises course content and allows learners to access and participate in courses online.
- Attendance Management : It gives the attendance status of students. Faculty will update the attendance periodically, which can be seen by students.
- Marks Management: It helps faculty and administrators efficiently manage and organise student grades and assignments.
- Alumni Management: The system may include features to manage alumni data, alumni networking, and maintaining connections with former students.
- Thesis Management: It includes thesis names with files.
- Library Management: It includes book names,author names, publisher names,and preferred branches.

1.3 Functionalities

- Provides the searching facilities based on various factors. Such as student, semester, courses, faculty, etc.
- It tracks all the information of student, faculty, etc
- Manage the information of Student.
- Shows the information and description of the department.
- To increase efficiency of managing the department, student.
- Manage the information of department.
- Editing, adding and updating of records is improved which results in proper resource management of department data.
- Manage the information of faculty.
- Integration of all records of faculty and students.

1.4 Background

Today, in colleges, student details are entered manually. Manual systems require a lot of time, manpower, and lot of time, other resources. The student details in separate records are a tedious task. Referring to all these records and updating them is needed. There is a chance for more manual errors. The issues that should be addressed in manual department management include securing faculty and student information.

1.4.1 Problems in existing system

- It was limited to a single system.
- It was less user-friendly.
- It has a lot of manual work (a manual system does not mean that we are working with pen and paper, it also includes working on spread sheets and other simple software) .
- It requires more no of employees to work.
- It was time-consuming process.
- The present system was very insecure.

1.4.2 Solution to these problems

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

- User friendliness is provided in the application with various controls.
- The system makes the overall project management much easier and more flexible.
- It can be accessed over the internet.
- Various classes have been used to provide file upload.
- There is no risk of data mismanagement at any level while the project is in process.
- It provides a high level of security.

1.5 Problem Definition

A department management system can help with things like data storage and student profiles. It can also help with better communication and student engagement. Accounts are managed and controlled by the program. It provides complete information about the DTDP department. In which faculty members and students can access the information and will be familiar with the department. It will provide an interactive environment for the faculty and students by providing knowledge of student attendance, assignments, results, grades, etc.

1.6 Aim of the Project

The aim of the DTDP Department Management System project is to develop a comprehensive and efficient system that enables effective management and administration of various departments within an organization. The project aims to streamline departmental processes, enhance communication and collaboration, centralize departmental information, improve decision-making, ensure data security, increase transparency and accountability, and provide scalability and flexibility. Additionally, the project aims to empower faculty members with tools for attendance management and marks management, and provide students with access to attendance details, marks information, results management, library resources, learning materials, alumni details, and thesis management. Overall, the project aims to optimize departmental performance, improve operational efficiency, and enhance the learning and administrative experience within the organization.

.

1.6 Role and responsibilities

Role- Frontend and Backend Developer

Responsibilities

Conducted a literature survey on papers 3 and 4 related to the project.Gathered relevant information and insights from the literature.

System Design

Worked on designing the learning management, alumni management, and thesis management systems.Created UML diagrams such as use-case diagrams, interaction diagrams, and block diagrams for the above systems.Ensured that the designs aligned with the project requirements and objectives.

Implementation

Developed both the frontend and backend of the learning management, alumni management, and thesis management systems.Created HTML pages for the user interface of these systems.Designed and implemented intuitive, user-friendly, and visually appealing interfaces for each system.Utilized the Django framework to render the HTML pages and implemented views and models to handle system functionality.Collaborated with the team to integrate these systems into the main website.Implemented system-specific features, such as course enrollment, progress tracking, assignments submission, alumni profile management, networking tools, event management, thesis submission, review, and tracking.Ensured proper navigation, usability, and seamless data management within each system. Designing the add alumni page with a form-based interface for alumni to input their contact details, employment history, and personal achievements.Implementing the frontend of the add alumni page, providing a user-friendly and intuitive form for data entry and performing client-side validation.Implementing the backend logic to handle the submission and storage of alumni information.Designing the user interface for uploading theses, ensuring a straightforward process for students to submit their work.Leveraging Python programming and utilizing an upload tool to facilitate the file upload

functionality.Implementing the necessary backend logic to handle thesis submissions, storage, and tracking.

Main Website Development

Assisted in developing a few pages of the main website.Contributed to the frontend and backend development of these pages.Ensured that the pages were visually appealing, user-friendly, and functioned correctly.

Documentation

Sourced information related to the learning management, alumni management, and thesis management systems.Organized and arranged chapters 3 and 4 of the project documentation, focusing on the systems you worked on.Ensured that the documentation accurately reflected the design and implementation of these systems.

In summary, as a Frontend and Backend Developer,my responsible for conducting a literature survey, designing the learning management, alumni management, and thesis management systems, implementing the frontend and backend of these systems, developing pages for the main website, and contributing to the documentation of the project.

CHAPTER 2

LITERATURE SURVEY

2.1 Paper 1

Title: School Management System

Author: Abhinav Sekhri

Published year: 2020

Abhinav sekhri proposed school management system which is introduced mainly for a School. This system includes functionality like holidays, classes, accounts, reports etc. On the other hand it do not contain library management module from where students as well staff can issue books related to their interest.

2.2 Paper 2

Title: College Department Management System

Authors: Ms.A.V.Sinhasane, Ms. A.N.Kashid, Ms. P.J.Kumbhar, Ms. P.R.Shirpale, Prof. S.L.Mortale

Published year: 2018

College department management system proposed by Ms.A.V.Sinhasane, Ms. A.N.Kashid, Ms. P.J.Kumbhar, Ms. P.R.Shirpale, Prof. S.L.Mortale from International Research Journal of Engineering and Technology which is introduced to reduce the stress and effort of a staff as well as students. This system have functionalities:like voting event details, feedback, newsline etc. This system is basically useful for students as they get the event details through but there are no basic modules which are important to such as marks, assignments, notes, etc.

2.3 Paper 3

Title: Online Attendance and Feedback System

Authors: Kartiki Datakar

Published year: 2016

Kartiki Datakar, from International Journal of Computer Science and Mobile Computing, quoted that Online Attendance and Feedback System is software developed for daily student attendance in schools, colleges, and institutes. It facilitates to access the information of a particular student in a particular class. It is concluded that a graduated approach to result monitoring is the most effective response, in which sanctions have a place, although only as a last resort. Online Attendance and Feedback System are software developed for daily student attendance in schools, colleges, and institutes.

2.4 Paper 4

Title: Research Paper on College Management System

Authors: Lalit Joshi

Published year: 2015

Lalit Joshi proposed, “A Research Paper on College Management System” , International Journal of Computer Applications, referred that the system utilizes user authentication, displaying only information necessary for an individual’s duties. Additionally, each sub-system has authentication allowing authorized users to create or update information in that subsystem. All data is thoroughly reviewed and validated on the server before actual record alteration occurs. In addition to a staff user interface, the system plans for student user interlace, allowing users lo access Information and submit requests online thus reducing processing time.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Document purpose

This document specifies the requirements for a college management system. The system will be used to manage the day-to-day operations of the department, including student records, faculty records, library resources, etc. The primary purpose of this application is to save time and reduce paperwork as students can view all the details online and faculty can mark the attendance and grade students more conveniently and quickly.

3.2 Intended audience and document overview

This document is intended for the following audiences:

- Developers who will be responsible for developing the system.
- Testers who will be responsible for testing the system.

Users of the system:

- Admin
- Faculty
- Student

3.3 Definitions, Acronyms, and Abbreviations

This document is intended for the following audiences:

- DMS: Department Management System
- DB: Data Base
- GUI:Graphical User Interface
- HTTP: Hypertext Transfer Protocol
- HTML: Hypertext Markup Language
- MySQL: Open-source relational database management system
- SQL:Structured Query Language

3.4 Functional Requirements

- The system must allow users to create, update, and delete student records.
- The system must allow users to create, update, and delete faculty records.
- The system must allow users to register for courses.
- The system must allow users to view their financial aid information.

3.5 Non-Functional Requirements

- The system must be secure.
- The system must be reliable.
- The system must be scalable.
- The system must be easy to use.

3.6 Software Requirements

For developing the application the following are the Software Requirements:

1. **Python** is a popular high-level programming language used for a variety of purposes such as web development, data analysis, artificial intelligence, and scientific computing. It is known for its simplicity, readability, and ease of use. Python supports multiple programming paradigms, including object-oriented, imperative, and functional programming styles. It has a large and active community that contributes to the development of various libraries and tools, making it easy to accomplish complex tasks with minimal effort. Python's popularity also comes from its versatility, as it runs on various operating systems and platforms, including Windows, macOS, and Linux. Overall, Python is a powerful and flexible language that can be used in a wide range of applications.
2. **Django** is a popular open-source web framework written in Python that follows the Model-View-Controller (MVC) architectural pattern. It is designed to simplify the process of building complex web applications by providing developers with a high-level, reusable set of components and tools. Django includes a built-in

administrative interface, an object-relational mapper (ORM), a templating engine,and a routing system, making it easy to develop database-driven web applications quickly. It also includes robust security features and supports various authentication mechanisms, including OAuth and LDAP. Django is used by companies such as Instagram, Mozilla, and Spotify, and it has a large and active community that contributes to its development and maintenance. Overall, Django is a powerful and flexible web framework that allows developers to create scalable and maintainable web applications.

3. **VS Code (Visual Studio Code)** is a popular, free and open-source code editor developed by Microsoft. It supports a wide range of programming languages, and has a robust set of features, including debugging, Git integration, syntax highlighting, code completion, and extensions. It offers a highly customizable interface and supports various themes and settings that can be tailored to suit individual preferences. VS Code also includes a built-in terminal, which allows developers to execute commands without leaving the editor. It runs on various operating systems, including Windows, macOS, and Linux, and is highly regarded for its performance, stability, and ease of use. Overall, VS Code is a powerful and versatile code editor that offers a rich set of features and flexibility for developers.

4. **XAMPP** is one of the widely used cross-platform web servers, which helps developers to create and test their programs on a local webserver. It was developed by the Apache Friends, and its native source code can be revised or modified by the audience. It consists of Apache HTTP Server, MariaDB, and interpreter for the different programming languages like PHP and Perl. It is available in 11 languages and supported by different platforms such as the IA-32 package of Windows & x64 package of macOS and Linux.

5. **MySQL** is one of the most widely used Open-source Relational Database Management Systems that uses a simple Client-Server Model to assist users in managing Relational Databases, or data stored in rows and columns across tables. It makes use of the well-known query language Structured Query Language (SQL), which enables users to conduct all CRUD (Create, Read, Update, and Delete) actions.

3.7 Operating Systems supported

- Windows 10 64 bit OS.

3.8 Debugger and Emulator

- Any Browser (Particularly Chrome).

3.9 Hardware Requirements

For developing the application the following are the Hardware Requirements:

- Processor: Intel i5
- RAM: 4 GB or higher
- Space on Hard Disk: minimum 1 TB

3.10 Technologies Used

- Front end: HTML, CSS, Bootstrap and JavaScript
- Back end: Python Django framework
- RDBMS: MySQL
- Web server: XAMPP

CHAPTER 4

SYSTEM DESIGN

INTRODUCTION

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and system engineering.

4.1 System architecture

The goal of system architecture is to create a system that is well-designed, efficient, and easy to maintain. By defining the components and interfaces of the system, system architecture can help to ensure that the system is built correctly and that it can be easily modified in the future.

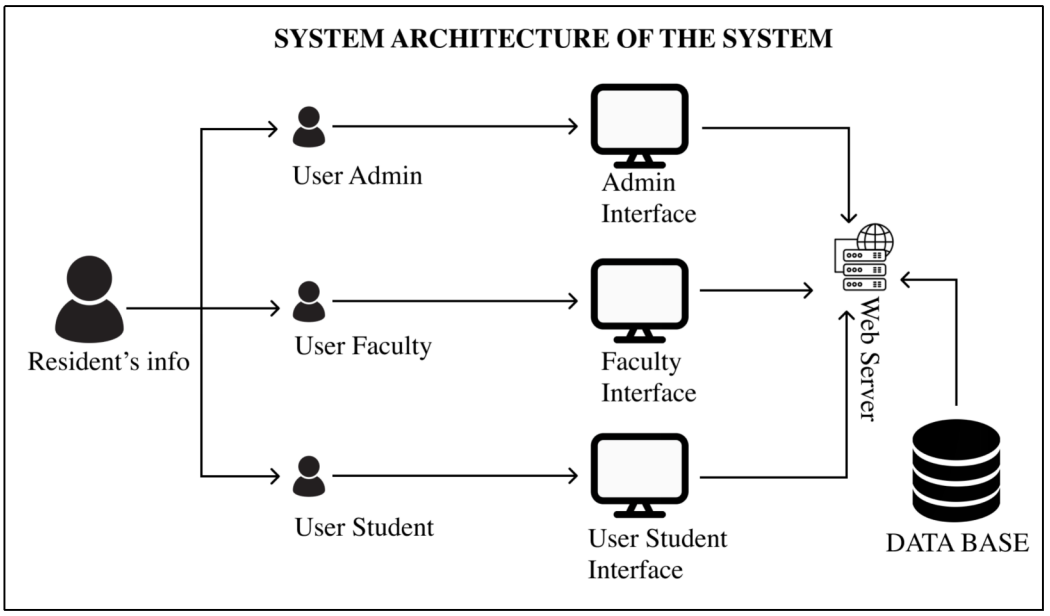


Figure 4.1: System Architecture

4.2 Module Design

The modules can can be divided into diverse types.

They are:

- Admin
- Faculty
- Student

4.2.1 Admin

1. Faculty Management

- Add new faculty: Admin can add new faculty member to the system.
- Manage faculty: This helps admin to change the status of the faculty and to delete existing record from the system

2. Subject Management

- Add subject: Admin can add subjects with subject name and semester name.
- Manage subjects: This helps admin to manage the subjects i.e to delete the subjects.
- Assign subject to the faculty: To allocate subject to the concern faculty.

3. Student Management

- Add student: Admin can add students with their details to the system.
- Manage students: This helps admin to change the status , edit the information of students and to delete the student from the system.

4. Attendance Management

- View Attendance details: Admin can view the attendance of a particular subject of the semester.

5. Marks Management

- Add semester marks: Admin can add marks to the students in subject and semester wise.
- View marks: Admin can view the marks of the students.

6. Library Management

- Add resource: Admin can add the books with the title and author names which helps students to access these resources.
- Manage resources: To delete the books from the system.

7. Learning Management

- Add new course: Admin can add new courses with the course name and price.
- Manage courses: This helps admin to edit the course details and to delete the course from the system.

8. Alumni Management

- Add Alumni: To add details about the alumni.
- Manage Alumni: To edit and delete added records.

9. Thesis Management

- Upload Thesis: Admin can upload thesis files which students can view.
- Manage Thesis: To delete the uploaded thesis from the system.

4.2.2 Faculty

1. Attendance Management

- Add attendance: Faculty can mark the attendance of the students by selecting the semester and subject name.
- View attendance details: To view the attendance of the students.

2. Marks Management

- Add marks: To grade the students.
- View marks: To view mark details of the students.

4.2.3 Student

1. Attendance Management

- View attendance: To view the attendance status of himself/herself.

2. Marks Management

- View marks: To view marks scored in semester wise.

3. Result Management

- View results: To view the status of the result in every semester.

4. Library Management

- View books: To view available books in the library.

5. Learning Management

- View courses: To view the available courses and to purchase the interested course.
- My course: To view the details of the selected course.

6. Alumni Management

- View alumni details: To view the details of the alumni.

7. Thesis Management

- View thesis: To view available theses and view thesis reports.

4.3 UML Diagrams

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.UML diagrams, in this case, are used to communicate different aspects and characteristics of a system.

4.3.1 Block Diagrams

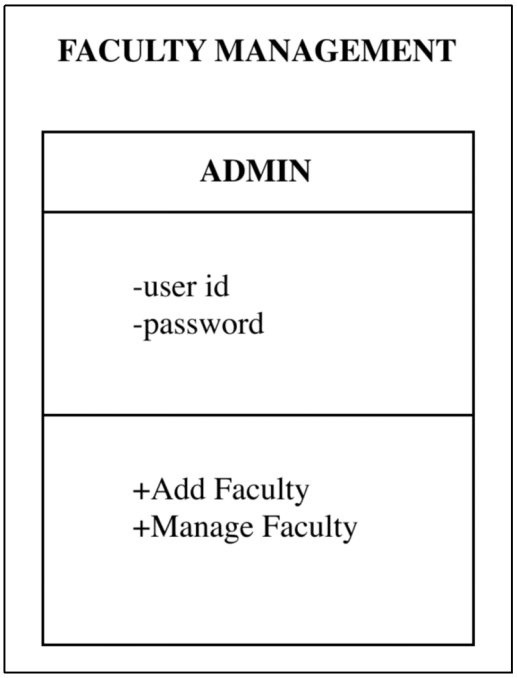


Figure 4.3.1.1: Faculty Management

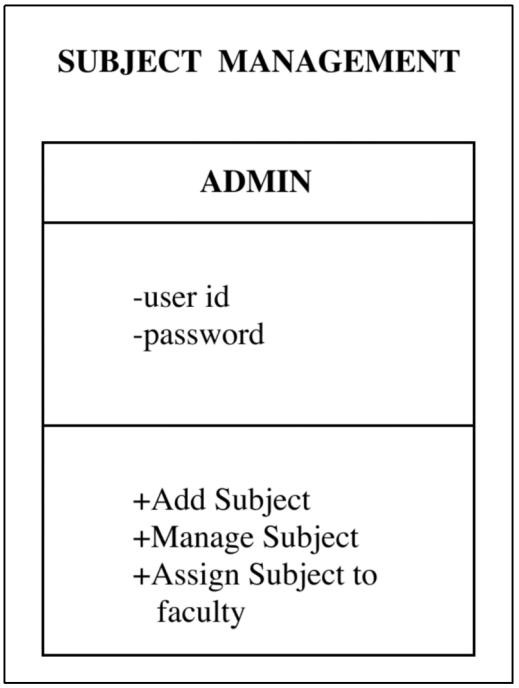


Figure 4.3.1.2: Subject Management

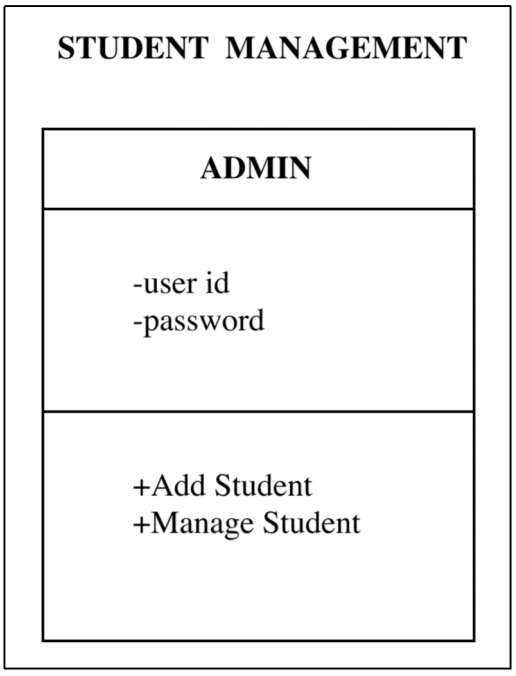


Figure 4.3.1.3: Student Management

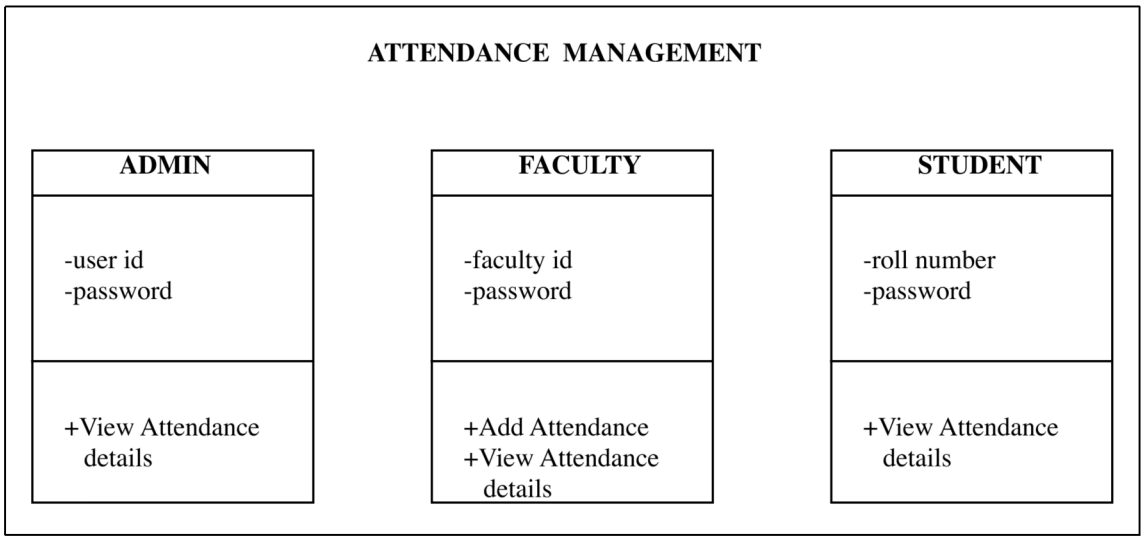


Figure 4.3.1.4: Attendance Management

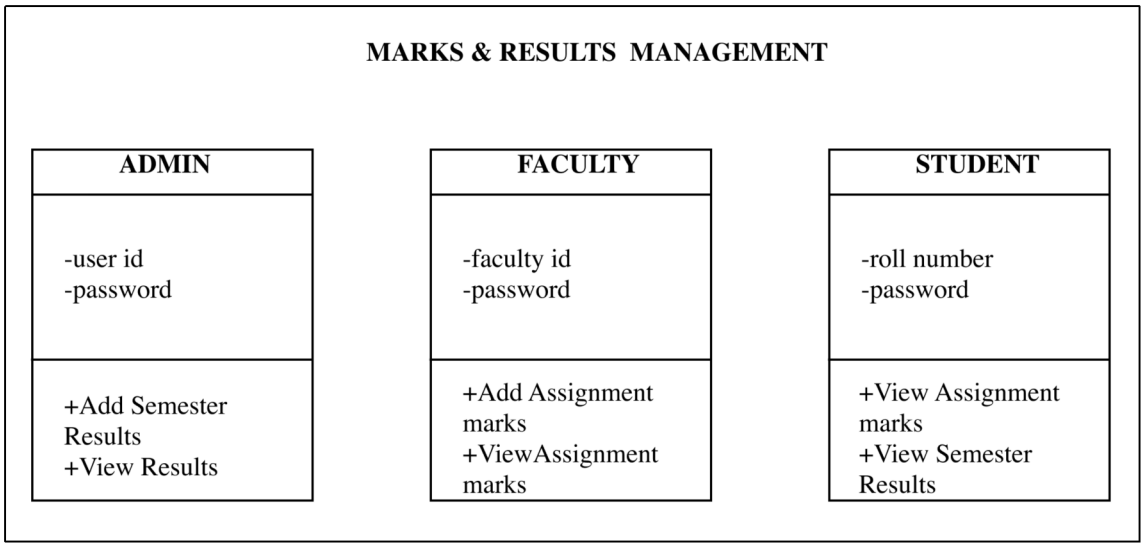


Figure 4.3.1.5: Marks & Results Management

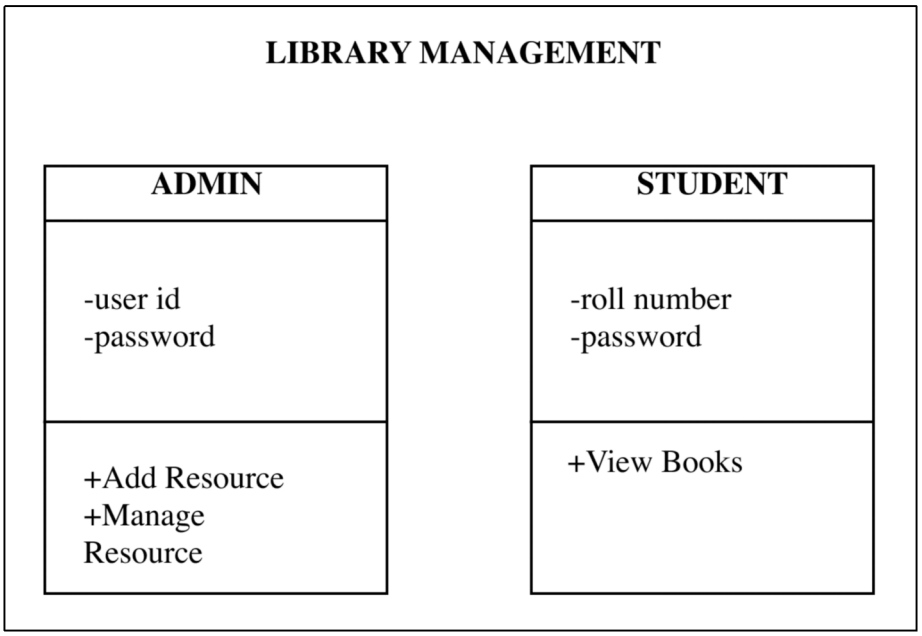


Figure 4.3.1.6: Library Management

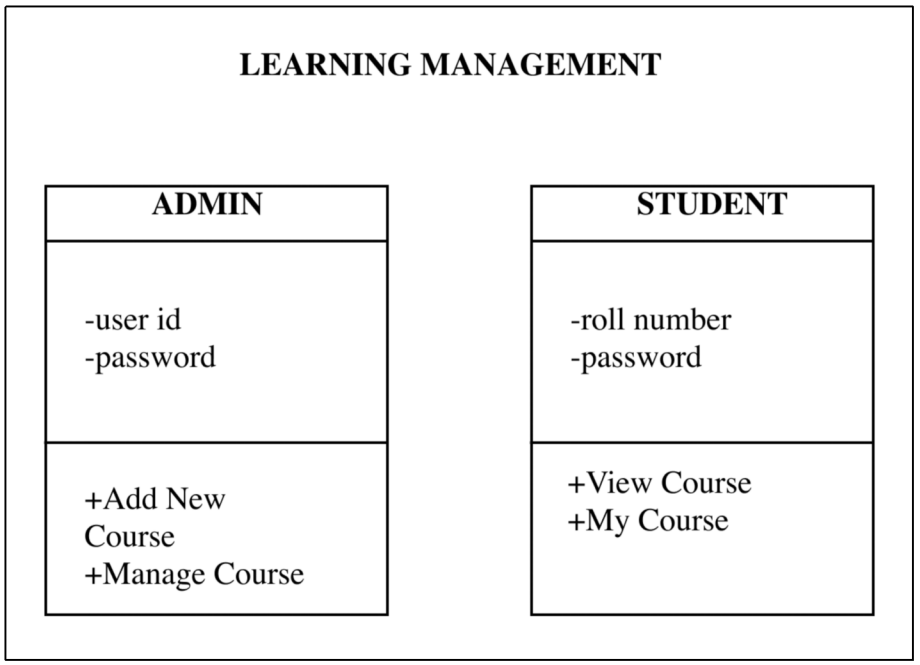


Figure 4.3.1.7: Learning Management

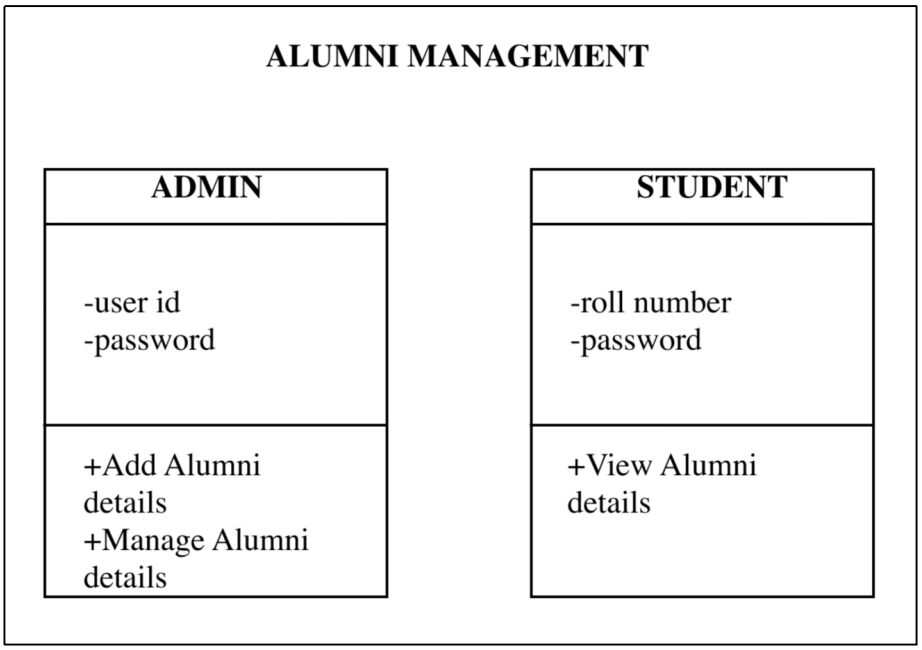


Figure 4.3.1.8: Alumni Management

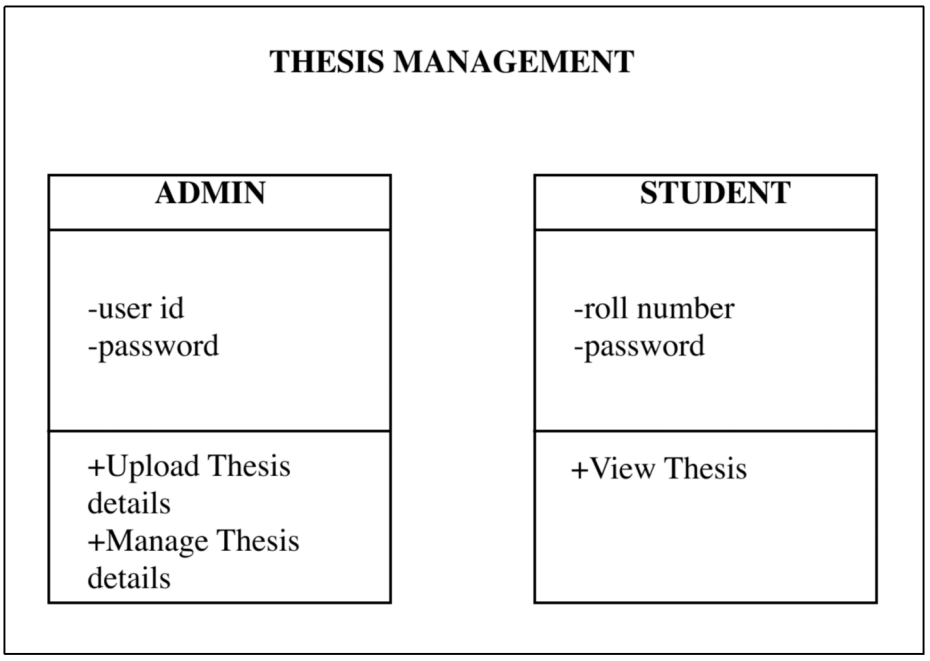


Figure 4.3.1.9: Thesis Management

4.3.2 Activity Diagrams

Main System

- The system starts at the main login page.
- Users can choose to log in as an admin, faculty member, or student.
- After logging in, each user is directed to their respective dashboard.

4.3.2.1 Admin Activities

On the admin dashboard, the admin can perform various tasks:

1. Faculty Management

- Add new faculty: The admin can add details of a new faculty member.
- Manage faculty: The admin can view, edit, or delete faculty member information.

2. Subject Management

- Add subject: The admin can add details of a new subject.
- Manage subject: The admin can view, edit, or delete subject information.
- Assign subjects to faculty: The admin can assign subjects to specific faculty members.

3. Student Management

- Add student: The admin can add details of a new student.
- Manage student: The admin can view, edit, or delete student information.

4. Attendance Management

- View attendance details: The admin can view attendance records for all students.

5. Marks Management

- Add semester marks: The admin can enter marks for students.
- View marks details: The admin can view marks details for all students.

6. Library Management

- Add resources: The admin can add new resources to the library.
- Manage resources: The admin can view, edit, or delete library resources.

7. Learning Management System

- Add new courses: The admin can add new courses to the learning management system.
- Manage courses: The admin can view, edit, or delete courses.

8. Alumni Management

- Add alumni details: The admin can add details of alumni.
- Manage alumni details: The admin can view, edit, or delete alumni information.

9. Thesis Management

- Upload thesis details: The admin can upload thesis details.
- Manage thesis details: The admin can view, edit, or delete thesis information.

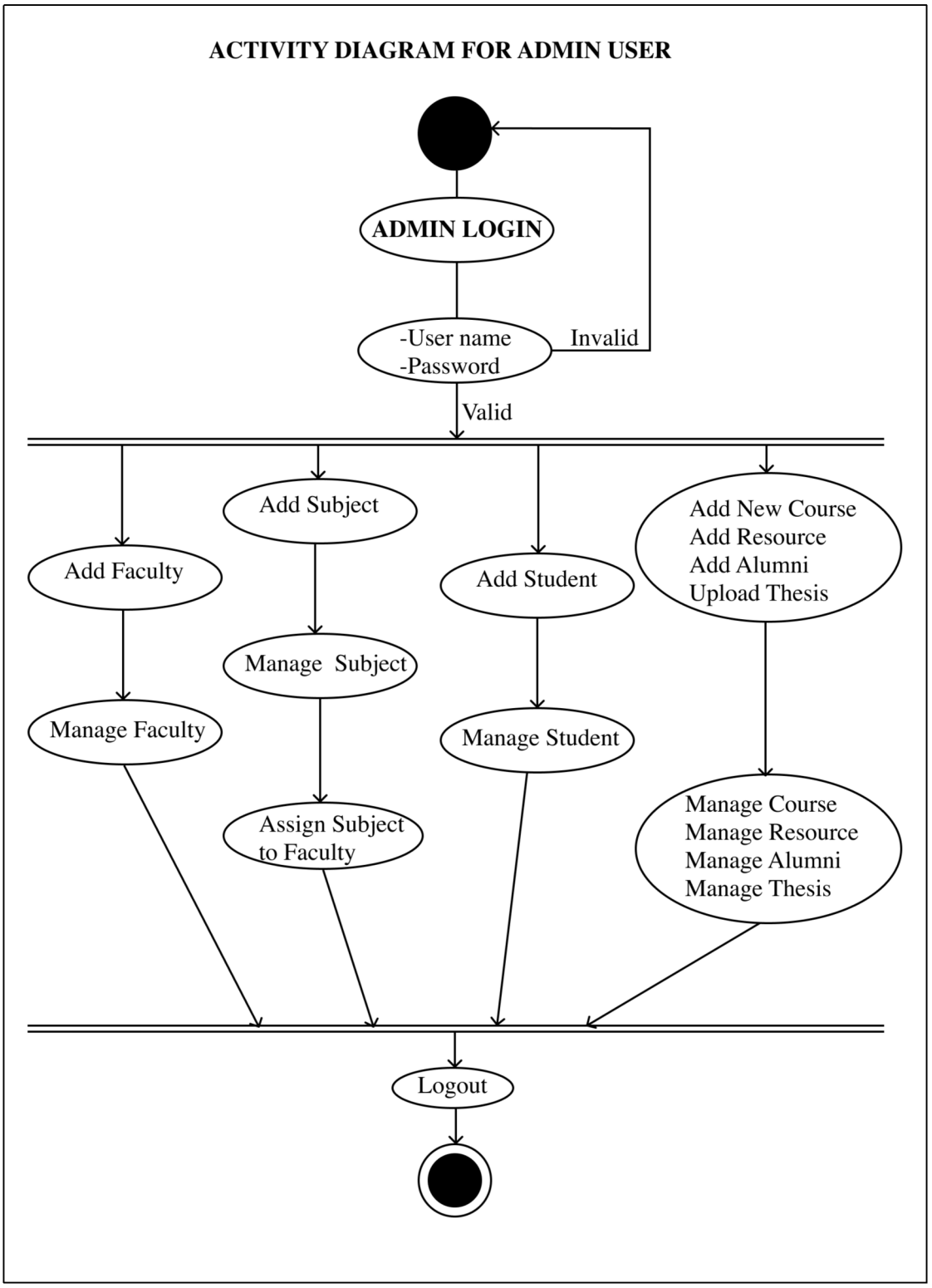


Figure 4.3.2.1: Activity diagram for admin user

4.3.2.2 Faculty Activities

On the faculty dashboard, faculty members can perform various tasks:

1. Attendance Management

Add attendance: Faculty members can mark attendance for their assigned students.

View attendance: Faculty members can view attendance records of their students.

2. Marks Management

Add marks: Faculty members can enter marks for their assigned students.

View marks: Faculty members can view marks details of their students.

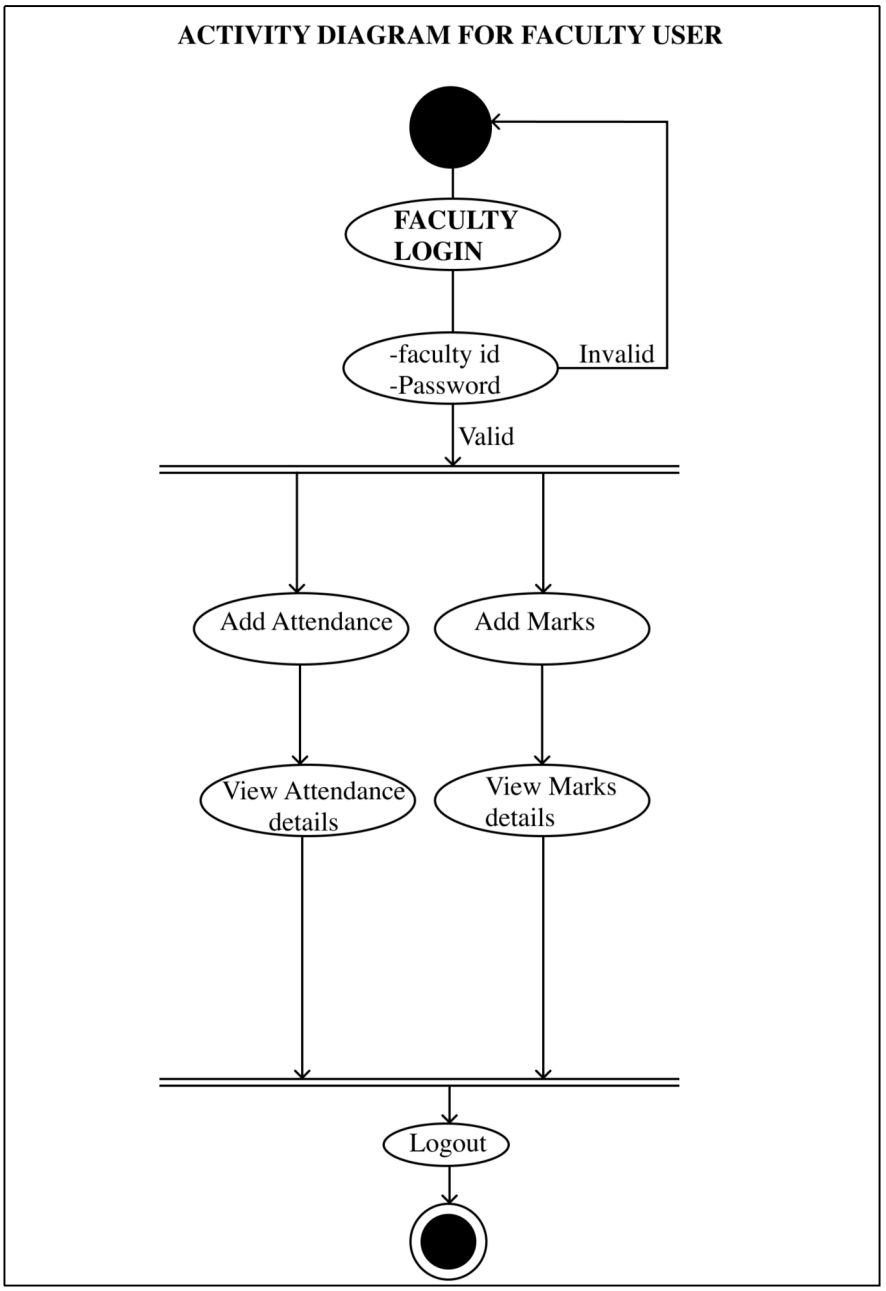


Figure 4.3.2.2: Activity diagram for faculty user

4.3.2.3 Student Activities

On the student dashboard, students can perform various tasks:

1. Attendance Management

- View attendance details: Students can view their attendance records.

2. Marks Management

- View marks details: Students can view their marks details.

3. Results Management

- View results: Students can view their semester results.

4. Library Management System

- View books: Students can browse and view books available in the library.

5. Learning Management System

- View courses: Students can browse and view available courses.
- My course: Students can view the courses they are enrolled in.

6. Alumni Management

- View alumni details: Students can view details of alumni.

7. Thesis Management

View thesis: Students can view thesis details.Today, in colleges, student details are entered manually. Manual systems require a lot of time, manpower, and lot of time, other resources. The student details in separate records are a tedious task. Referring to all these records and updating them is needed. There is a chance for more manual errors. The issues that should be addressed in manual department management include securing faculty and student information.

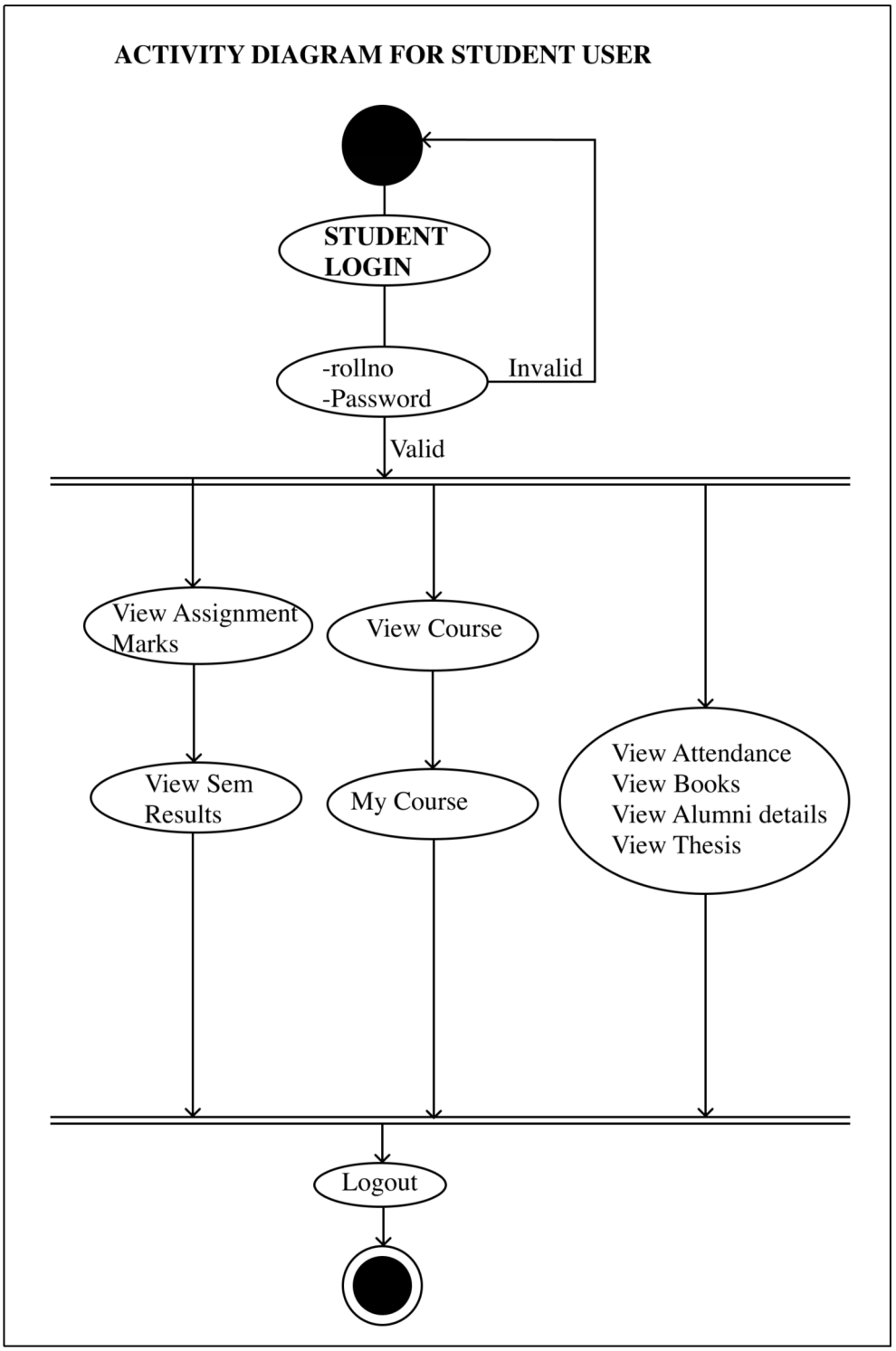


Figure 4.3.2.3: Activity diagram for student user

4.3.3 Use-Case Diagram

4.3.3.1 Admin Use Cases

- Add New Faculty: The admin can add a new faculty member to the system.
- Manage Faculty: The admin can view, edit, or delete faculty member information.
- Add Subject: The admin can add a new subject to the system.
- Manage Subject: The admin can view, edit, or delete subject information.
- Assign Subjects to Faculty: The admin can assign subjects to specific faculty members.
- Add Student: The admin can add a new student to the system.
- Manage Student: The admin can view, edit, or delete student information.
- View Attendance Details: The admin can view attendance records for all students.
- Add Semester Marks: The admin can enter marks for students.
- View Marks Details: The admin can view marks details for all students.
- Add Resources: The admin can add new resources to the library.
- Manage Resources: The admin can view, edit, or delete library resources.
- Add New Courses: The admin can add new courses to the learning management system.
- Manage Courses: The admin can view, edit, or delete courses.
- Add Alumni Details: The admin can add details of alumni.
- Manage Alumni Details: The admin can view, edit, or delete alumni information.
- Upload Thesis Details: The admin can upload thesis details.
- Manage Thesis Details: The admin can view, edit, or delete thesis information.

4.3.3.2 Faculty Use Cases

- Add Attendance: Faculty members can mark attendance for their assigned students.
- View Attendance: Faculty members can view attendance records of their students.
- Add Marks: Faculty members can enter marks for their assigned students.
- View Marks: Faculty members can view marks details of their students.

4.3.3.3 Student Use Cases

- View Attendance Details: Students can view their attendance records.
- View Marks Details: Students can view their marks details.
- View Results: Students can view their semester results.
- View Books: Students can browse and view books available in the library.
- View Courses: Students can browse and view available courses.
- My Course: Students can view the courses they are enrolled in.
- View Alumni Details: Students can view details of alumni.
- View Thesis: Students can view thesis details.

The use case diagram represents the different actors (Admin, Faculty, Student) and their interactions with the system.

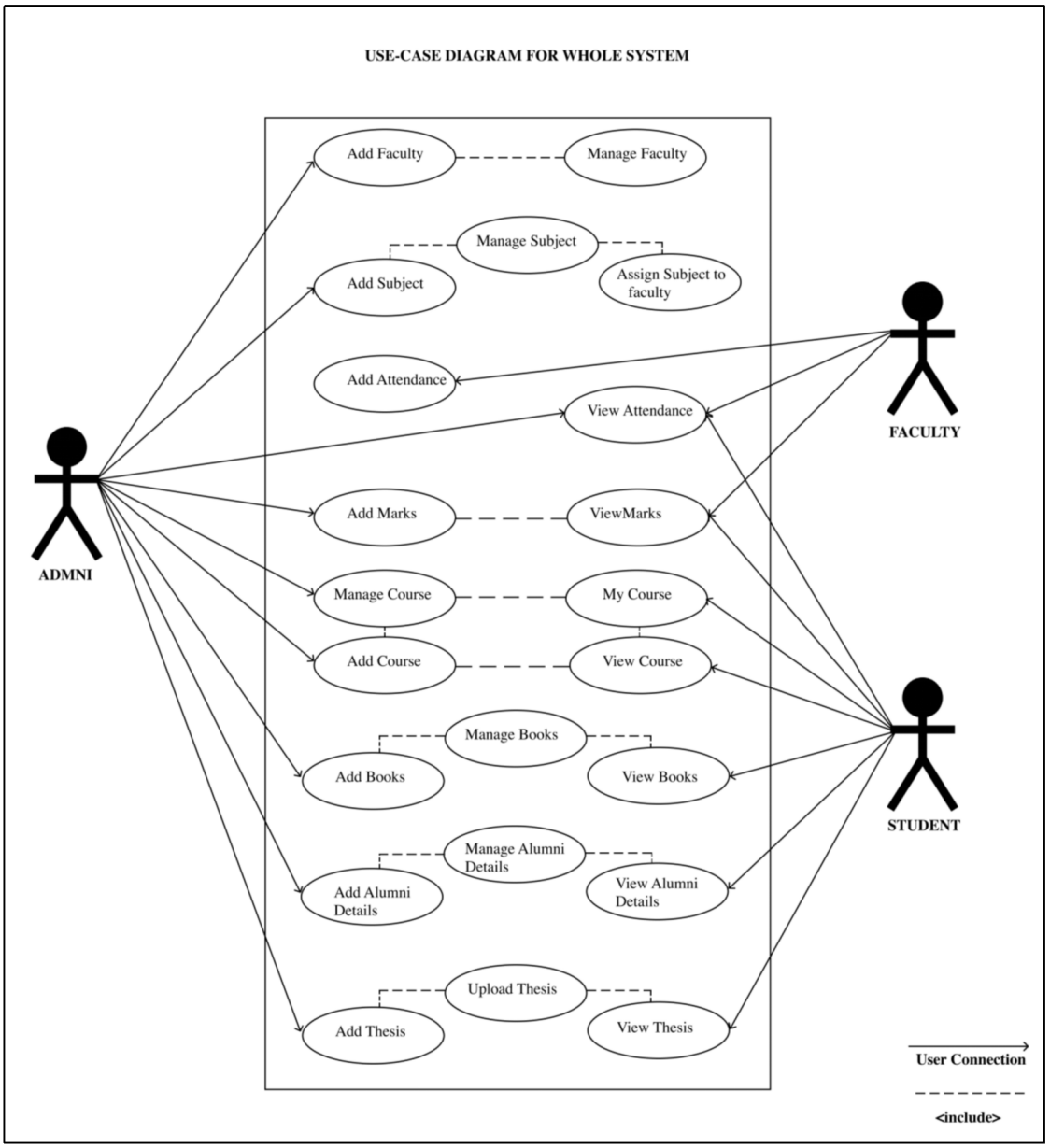


Figure 4.3.3: Use-case diagram for whole system

4.3.4 Interaction Diagram

4.3.4.1 Admin and Faculty Interaction

- The admin logs into the system and accesses the dashboard.
- The admin selects the "Faculty Management" option.
- The system retrieves the list of faculty members from the database and displays it to the admin.
- The admin selects a specific faculty member to manage.
- The system retrieves the faculty member's details and presents them to the admin.
- The admin performs actions such as editing or deleting the faculty member's information.
- The system updates the faculty member's details in the database accordingly.
- The admin may also assign subjects to the faculty member by selecting the "Subject Management" option and following similar steps.
- The faculty member logs into the system and accesses the dashboard.
- The faculty member selects the "Attendance Management" option.
- The system retrieves the list of assigned students and their attendance records.
- The faculty member can mark attendance for the students by selecting the appropriate option.
- The system updates the attendance records in the database accordingly.
- The faculty member may also select the "Marks Management" option and follow similar steps to add or view marks for the assigned students.

4.3.4.2 Faculty and Student Interaction

The faculty member logs into the system and accesses the dashboard.

- The faculty member selects the "Attendance Management" option.
- The system retrieves the list of assigned students and their attendance records.

- The faculty member can view the attendance details for the students.
- The system presents the attendance details to the faculty member.
- The faculty member may also select the "Marks Management" option and follow similar steps to add or view marks for the assigned students.
- The student logs into the system and accesses the dashboard.
- The student selects the "Attendance Management" option.
- The system retrieves the student's attendance details from the database.
- The student can view their attendance details as presented by the system.
- The student may also select the "Marks Management" option and follow similar steps to view their marks details.

4.3.4.3 Student and Admin Interaction

- The student logs into the system and accesses the dashboard.
- The student selects the "Results Management" option.
- The system retrieves the student's semester results from the database and presents them to the student.
- The student can view their results as displayed by the system.
- The student may also select the "Library Management System" option to view available books or the "Learning Management System" option to view available courses.
- The admin logs into the system and accesses the dashboard.
- The admin selects the "Student Management" option.
- The system retrieves the list of students from the database and displays it to the admin.
- The admin selects a specific student to manage.
- The system retrieves the student's details and presents them to the admin.
- The admin can perform actions such as editing or deleting the student's information.
- The system updates the student's details in the database accordingly.

igure 4.3.4: Interaction diagram for whole system

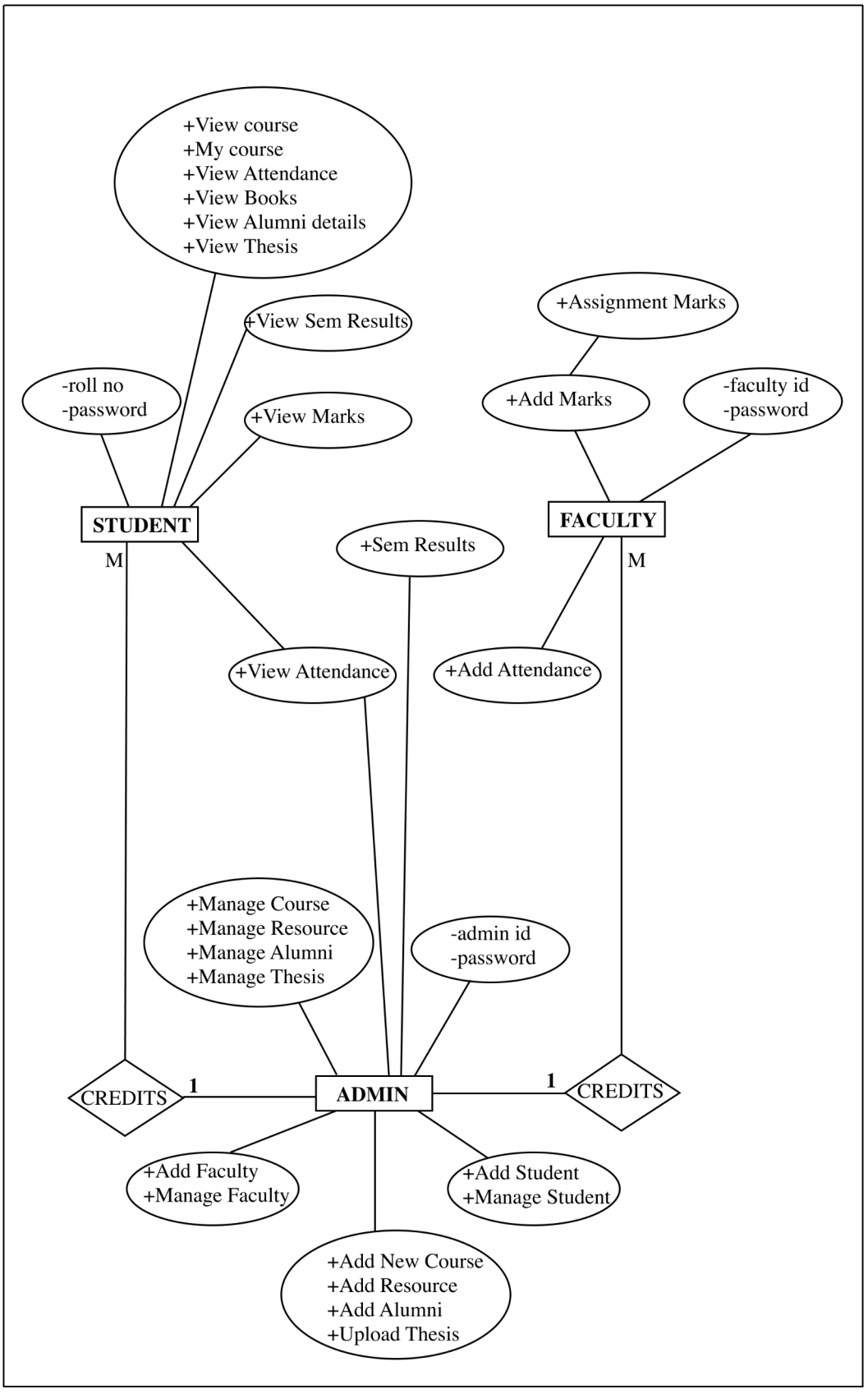


Figure 4.3.5: Entity Relationship diagram

CHAPTER 5
IMPLEMENTATION

5.1 Introduction

Django is based on **MVT (Model-View-Template)** architecture. MVT is a software design pattern for developing a web application.

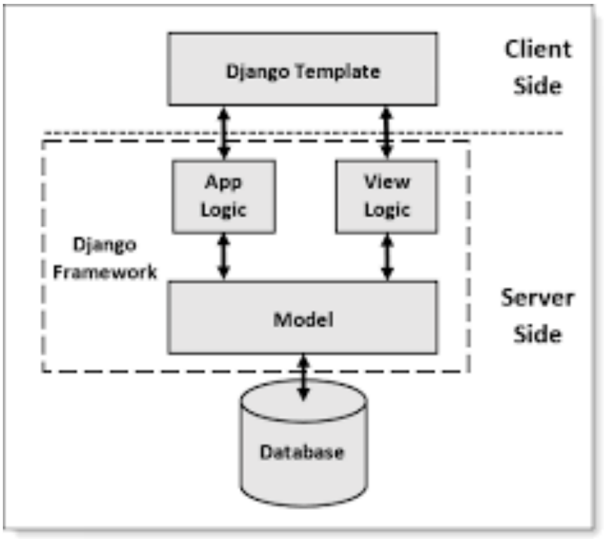


Figure 5.1: Django Architecture.

MVT Structure has the following three parts

- Model: The model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySql, Postgres).
- View: The View is the user interface — what you see in your browser when you render a website. It is represented by HTML/CSS/Javascript and Jinja files.
- Template: A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

5.2 Implementation

Step 1- Install Django.

Step 2- Create a folder with the name department_Management_System and open it with VS Code.

Step 3- Open the terminal and create a new project “collegeproject” using the below command.

~ django-admin startproject collegeproject

Step 4- Enter inside the folder department_Management_System and create the app “adminapp”, “facultyapp”, “studentapp” and “mainapp”

~ python manage.py startapp adminapp

~ python manage.py startapp facultyapp

~ python manage.py startapp studentapp

~ python manage.py startapp mainapp

Step5- Go to department_Management_System -> settings.py -> INSTALLED_APPS and add our app adminapp, facultyapp, studentapp, mainapp

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'adminapp',
    'facultyapp',
    'mainapp',
    'studentapp',
]
```

5.2.1 Creating views for the adminapp

```
from django.shortcuts import render,redirect
from adminapp.models import *
import string,random
from django.contrib import messages

def admin_dashboard(request):
```

```
    return render(request,'admin_template/index.html')

def add_faculty(request):
  guide_char = string.ascii_letters + string.digits
    if request.method == "POST":
        faculty_id = request.POST.get('faculty_id')
        faculty_fname = request.POST.get('faculty_fname')
        faculty_email = request.POST.get('faculty_email')
        faculty_department = request.POST.get('faculty_department')
        faculty_preff_sub = request.POST.get('faculty_preff_sub')
        faculty_profile = request.FILES['faculty_profile']
        faculty_pswd = ''.join(random.choices(guide_char, k=4))
        Faculty_Management.objects.create(F_ID = faculty_id, Full_Name =
faculty_fname, Email = faculty_email, Department = faculty_department,
Preferred_Subject = faculty_preff_sub, Profile = faculty_profile, Faculty_Password =
faculty_pswd)
        return redirect('add_faculty')
    return render(request,'admin_template/admin-add-faculty.html')

def manage_faculty(request):
    mang_faculty = Faculty_Management.objects.all()
    return render(request,'admin_template/admin-manage-
faculty.html',{'mg':mang_faculty})

def faculty_change_status(request,id):
    variable_status = Faculty_Management.objects.get(Faculty_ID=id)
    if(variable_status.Status == 'pending'):
        variable_status.Status = 'accepted'
        variable_status.save()
    elif(variable_status.Status == 'accepted'):
        variable_status.Status = 'rejected'
        variable_status.save()
    elif(variable_status.Status == 'rejected'):
        variable_status.Status = 'accepted'
```

```
            variable_status.save()
        else:
            variable_status.Status = 'accepted'
            variable_status.save()
        return redirect('manage_faculty')

def faculty_Delete(req,id):
    delete111 = Faculty_Management.objects.get(Faculty_ID=id)
    delete111.delete()
    return redirect('manage_faculty')


def add_subject(request):
    if request.method == "POST":
        sub_name = request.POST.get('sub_name')
        sub_branch = request.POST.get('sub_branch')
        sub_semester = request.POST.get('sub_semester')
        Subject_Management.objects.create(Subject_Name = sub_name, Branch_Name =
sub_branch, Semester = sub_semester)
        return redirect('add_subject')
    return render(request,'admin_template/admin-add-subject.html')

def manage_subject(request):
    mg_sub = Subject_Management.objects.all()
    return render(request,'admin_template/admin-manage-
subject.html',{'mg_s':mg_sub})

def delete_subject(request,id):
    del_sub = Subject_Management.objects.get(Subject_ID=id)
    del_sub.delete()
    return redirect('manage_subject')

def assign_sub_faculty(request):
    asssign_sub = Subject_Management.objects.all()
    facultys = Faculty_Management.objects.all()
```

```
    return render(request,'admin_template/admin-assign-subject-to-
faculty.html',{'assign':asssign_sub, 'fac' : facultys})

def assign_subject(request,id):
    return redirect('assign_sub_faculty')


def add_students(request):
    guide_char = string.ascii_letters + string.digits
    if request.method == "POST":
        stu_name = request.POST.get('stu_fname')
        stu_roll_no = request.POST.get('stu_roll_no')
        stu_email = request.POST.get('stu_email')
        stu_branch = request.POST.get('stu_branch')
        stu_semester = request.POST.get('stu_semester')
        stu_address = request.POST.get('stu_address')
        stu_phone = request.POST.get('stu_phone')
        stu_profile = request.FILES['stu_profile']
        stu_pswd = ''.join(random.choices(guide_char, k=4))
        Student_Management.objects.create(Full_Name = stu_name, Roll_No =
stu_roll_no, Email = stu_email, Branch_Name = stu_branch, Semester = stu_semester,
Address = stu_address, Phone_No = stu_phone, Profile = stu_profile,
Student_Password = stu_pswd)
    return render(request,'admin_template/admin-add-student.html')

def stud_status(request,id):
    variable_status = Student_Management.objects.get(Student_ID=id)
    if(variable_status.Status == 'pending'):
        variable_status.Status = 'accepted'
        variable_status.save()
    elif(variable_status.Status == 'accepted'):
        variable_status.Status = 'rejected'
        variable_status.save()
    elif(variable_status.Status == 'rejected'):
        variable_status.Status = 'accepted'
```

```
            variable_status.save()
        else:
            variable_status.Status = 'accepted'
            variable_status.save()
        return redirect('manage_student')

def delete_student(request,id):
    del_sub = Student_Management.objects.get(Student_ID=id)
    del_sub.delete()
    return redirect('manage_student')



def manage_student(request):
    manage_stu = Student_Management.objects.all()
    return render(request,'admin_template/admin-manage-
student.html',{'mg_std':manage_stu})

def add_resources(request):
    if request.method == "POST":
        res_title = request.POST.get('res_title')
        res_author = request.POST.get('res_author')
        res_publisher = request.POST.get('res_publisher')
        res_preferred = request.POST.get('res_preferred')
        Library_Management.objects.create(Title = res_title, Author_Name = res_author,
Publisher_Name = res_publisher, Preferred_Name = res_preferred)
    return render(request,'admin_template/admin-add-resources.html')

def manage_resourses(request):
    manage_res = Library_Management.objects.all()
    return render(request,'admin_template/admin-manage-
resources.html',{'mg_res':manage_res})

def delete_resourse(request,id):
    del_sub = Library_Management.objects.get(Library_ID=id)
```

```
    del_sub.delete()
    return redirect('manage_resources')

def add_courses(request):
    if request.method == "POST":
        course = request.POST.get('course')
        price = request.POST.get('price')
        photo = request.FILES['photo']
        textarea = request.POST.get('textarea')
        Learning_Management.objects.create(Course_Name = course, Price = price,
Photo = photo, Description = textarea)
    return render(request,'admin_template/admin-add-new-courses.html')

def manage_course(request):
    mg_course = Learning_Management.objects.all()
    return render(request,'admin_template/admin-manage-
courses.html',{'mg_course':mg_course})

def delete_course(request,id):
    del_sub = Learning_Management.objects.get(Learning_ID=id)
    del_sub.delete()
    return redirect('manage_course')

def edit_course(request,id):
    edit_course = Learning_Management.objects.get(Learning_ID = id)
    print(edit_course.Course_Name)
    if request.method == 'POST':
        course_name = request.POST.get('course_name')
        course_price = request.POST.get('course_price')
        # course_photo = request.POST.get('course_photo')
        course_desp = request.POST.get('course_desp')
        # stu_profile = request.FILES['edit_profile']

        if len(request.FILES) != 0:
```

```
            stu_pic = request.FILES['course_photo']
            edit_course.Course_Name = course_name
            edit_course.Price =course_price
            edit_course.Description =course_desp
            edit_course.Photo =stu_pic
            edit_course.save()
        else:
            edit_course.Course_Name = course_name
            edit_course.Price =course_price
            edit_course.Description =course_desp
            # edit_course.Profile =stu_profile
            edit_course.save()
        return redirect('edit_course', id=id)
    return render(request,'admin_template/admin-course-edit.html', {'stu':edit_course})


def add_sem_marks(request):
    if request.method == "POST":
        branch = request.POST.get('branch')
        semester = request.POST.get('semester')
        subject_name = request.POST.get('subject')
        print(subject_name, 'djufd', branch, "jygfdb")
        a = Student_Management.objects.filter(Branch_Name = branch, Semester =
semester)
        return render(request,'admin_template/admin-add-semester-marks.html',
{'stu_details' : a, 'sub':subject_name})
    return render(request,'admin_template/admin-add-semester-marks.html')


def admin_view_marks(request):
    if request.method == "POST":
        branch = request.POST.get('branch')
        semester = request.POST.get('semester')
        subjects = request.POST.get('subject')
        # print(subject, 'sjhc')
```

```
        view_marks = Student_Management.objects.filter(Branch_Name = branch,
Semester = semester)
        return render(request,'admin_template/admin-view-marks-
details.html',{'v_marks':view_marks, 'sub': subjects})
    return render(request,'admin_template/admin-view-marks-details.html')

# def edit_course(request):
#     return render(request,'admin_template/admin-course-edit.html')

def edit_student(request,id):
    edit_stud = Student_Management.objects.get(Student_ID = id)
    print(edit_stud.Full_Name)
    if request.method == 'POST':
        stu_name = request.POST.get('edit_fname')
        # stu_roll_no = request.POST.get('edit_roll_no')
        # stu_email = request.POST.get('edit_email')
        stu_branch = request.POST.get('edit_branch')
        stu_semester = request.POST.get('edit_semester')
        stu_address = request.POST.get('edit_address')
        stu_phone = request.POST.get('edit_phone_no')
        # stu_profile = request.FILES['edit_profile']

        if len(request.FILES) != 0:
            stu_pic = request.FILES['edit_profile']
            edit_stud.Full_Name = stu_name
            edit_stud.Branch_Name =stu_branch
            edit_stud.Semester =stu_semester
            edit_stud.Address =stu_address
            edit_stud.Phone_No =stu_phone
            edit_stud.Profile =stu_pic
            edit_stud.save()
        else:
            edit_stud.Full_Name = stu_name
            # edit_stud.Roll_No = stu_roll_no
```

```
#             edit_stud.Email = stu_email
            edit_stud.Branch_Name =stu_branch
            edit_stud.Semester =stu_semester
            edit_stud.Address =stu_address
            edit_stud.Phone_No =stu_phone
            # edit_stud.Profile =stu_profile
            edit_stud.save()
        return redirect('edit_student', id=id)
    return render(request,'admin_template/admin-edit-student.html', {'stu':edit_stud})

def admin_add_alumni(request):
    add_alumni = Student_Management.objects.all()
    return render(request,'admin_template/admin-add-alumni-
details.html',{'alumni':add_alumni})

def add_alumni_btn(req, id):
    stu_details = Student_Management.objects.get(Student_ID = id)
    stu_details.Alumni = 'Alumni'
    stu_details.save()

    return redirect("admin_add_alumni")

def manage_alumni(request):
    stu_details = Student_Management.objects.filter(Alumni = 'Alumni')
    return render(request,'admin_template/admin-manage-alumni-
details.html',{'add':stu_details})

def delete_alumni(request,id):
    del_sub = Student_Management.objects.get(Student_ID = id)
    del_sub.Alumni = 'Not-Alumni'
    del_sub.save()
    return redirect('manage_alumni')

def admin_view_attendance(request):
```

```
    if request.method == "POST":
        branch = request.POST.get('branch')
        semester = request.POST.get('semester')
        subjects = request.POST.get('subject')
        attendance = Student_Management.objects.filter(Branch_Name = branch,
Semester = semester)
        return render(request,'admin_template/admin-view-attendance-
details.html',{'attd':attendance, 'sub': subjects})
    return render(request,'admin_template/admin-view-attendance-details.html')


def upload_thesis(request):
    if request.method == "POST":
        thesis_name = request.POST.get('thesis_name')
        thesis_file = request.FILES['thesis_file']
        Thesis_Management.objects.create(Thesis_Name = thesis_name, Thesis_File =
thesis_file)
        return redirect('upload_thesis')
    return render(request,'admin_template/admin-upload-thesis-details.html')

def manage_thesis(request):
    mg_thesis = Thesis_Management.objects.all()
    return render(request,'admin_template/admin-manage-thesis-
details.html',{'mg_t':mg_thesis})

def delete_thesis(request,id):
    del_sub = Thesis_Management.objects.get(Thesis_ID=id)
    del_sub.delete()
    return redirect('manage_thesis')

def admin_logout(req):
    messages.info(req,"Logged Out")
    return redirect('admin_login')
```

5.2.2 Creating facultyapp views

```
from django.shortcuts import render,redirect
from adminapp.models import *
from datetime import datetime
import time
from facultyapp.models import *
from django.contrib import messages
from datetime import date
from django.db.models import Q


# Create your views here.
def faculty_dashboard(request):
    return render(request,'faculty_template/index.html')


def faculty_view_attendance(request):
    view  = Subject_Management.objects.all()
    if request.method == "POST":
        branch = request.POST.get('branch')
        semester = request.POST.get('semester')
        subjects = request.POST.get('subject')
        # print(subject, 'sjhc')
        view_att = Attendence_Management.objects.filter(Stu_Branch = branch, Stu_Sem
= semester, Att_Subject = subjects )
        for i in view_att:
            i.Subject = subjects
            i.save()
        return render(request,'faculty_template/faculty-view-attendance-
details.html',{'view_att':view_att, 'sub': subjects, 'v_attend' : view})
    return render(request,'faculty_template/faculty-view-attendance-
details.html',{'v_attend' : view})
```

```
def faculty_add_attendance(request):
    current_date = time.strftime('%Y-%m-%d')
    subj  = Subject_Management.objects.all()
    if request.method == "POST":
        branch = request.POST.get('branch')
        semester = request.POST.get('semester')
        subject_name = request.POST.get('subject')
        # print(subject_name, 'djufd', branch, "jygfdb")
        a = Student_Management.objects.filter(Branch_Name = branch, Semester =
semester)
        for i in a:
            i.Subject = subject_name
            i.save()
        return render(request,'faculty_template/faculty-add-attendance.html',
{'stu_details' : a,'subjj':subj, 'date':current_date,})

    return render(request,'faculty_template/faculty-add-attendance.html', {'subjj':subj,
'date':current_date,})
    # return render(request,'faculty_template/faculty-add-attendance.html', {'sub':subj})

def mark_attendance(req):
    students = Student_Management.objects.filter(~Q(Subject = ""))
    if req.method == 'POST':
        for student in students:
            atten_status = req.POST.get(str(student.Student_ID))
            print(atten_status, "atten")
            att = Attendence_Management(Att_Date = date.today(), Att_Subject =
student.Subject, Student_Name  = student.Full_Name, Student_Foregin = student,
Att_Status = atten_status, Stu_Sem = student.Semester, Stu_Branch =
student.Branch_Name, Student_Roll = student.Roll_No)
            att.save()
            student.Subject = ""
            student.save()
        return redirect("faculty_add_attendance")
```

```
    return render(req, "faculty_template/faculty-add-attendance.html")



def faculty_add_marks(request):
    marks = Subject_Management.objects.all()
    if request.method == "POST":
        branch = request.POST.get('branch')
        semester = request.POST.get('semester')
        subject_name = request.POST.get('subject')
        print(branch)
        a = Student_Management.objects.filter(Branch_Name = branch, Semester =
semester)
        print(a,'no of objects')
        for j in a:
            j.Marks_Subject = subject_name
            j.Marks = 1
            j.save()
            print(j)
            print(j.Marks_Subject, "subject name ")
            return render(request,'faculty_template/faculty-add-marks.html',
{'marks_details' : a, 'subjj':marks})
    return render(request,'faculty_template/faculty-add-marks.html',{'subjj':marks})


def mark_add(req):
    print("mark_add")
    students = Student_Management.objects.filter(Marks = 1)
    print(students)
    if req.method == 'POST':
        # ab = req.POST.get("name1")
        for student in students:
            atten_status = req.POST.get(str(student.Student_ID))
            print(atten_status, "marks")
```

```
            att = Marks_Management(Student_Subject = student.Marks_Subject,
Add_Marks = atten_status, Student_Name  = student.Full_Name, Stu_Foregin =
student, Student_Semester = student.Semester, Student_Branch =
student.Branch_Name, Student_Roll_No = student.Roll_No)
            att.save()
            student.Marks_Subject = ""
            student.Marks = 0
            student.save()
        return redirect("faculty_add_marks")

    return render(req, "faculty_template/faculty-add-marks.html")


def faculty_view_marks(request):
    view_m = Subject_Management.objects.all()
    if request.method == "POST":
        branch = request.POST.get('branch')
        semester = request.POST.get('semester')
        subject_name = request.POST.get('subject')
        view = Student_Management.objects.filter(Branch_Name = branch, Semester =
semester)
        for i in view:
            i.Subject = subject_name
            i.save()
            return render(request,'faculty_template/faculty-view-
marks.html',{'view_marks' : view, 'v_marks' : view_m, 'subjects' : subject_name })
    return render(request,'faculty_template/faculty-view-marks.html',{'v_marks' :
view_m})

def faculty_logout(req):
    messages.info(req,"Logged Out")
    return redirect('faculty_login')
```

5.2.3 Creating studentapp views

```
from django.shortcuts import render,redirect
from adminapp.models import *
from django.contrib import messages

# Create your views here.

def student_dashboard(request):
    return render(request,'student_template/index.html')


def student_alumni_details(request):
    return render(request,'student_template/student-view-alumni-details.html')

def student_attendance_details(request):
    return render(request,'student_template/student-view-attendance-details.html')

def student_view_book(request):
    stud_books = Library_Management.objects.all()
    return render(request,'student_template/student-view-
books.html',{'stud_b':stud_books})

def student_view_courses(request):
    stud_course = Learning_Management.objects.all()
    return render(request,'student_template/student-view-
courses.html',{'stud_c':stud_course})

def student_my_courses(request):
    stu_details = Learning_Management.objects.all()
    return render(request,'student_template/student-my-
courses.html',{'add':stu_details})

def add_course_btn(req, id):
    stu_details = Learning_Management.objects.get(Learning_ID = id)
    stu_details.Alumni = 'Alumni'
    stu_details.save()
```

```
    return redirect("student_view_courses")


def student_marks_details(request):
    return render(request,'student_template/student-view-marks-details.html')

def student_view_results(request):
    return render(request,'student_template/student-view-results.html')

def student_view_thesis(request):
    stud_thesis = Thesis_Management.objects.all()
    return render(request,'student_template/student-view-
thesis.html',{'stud_the':stud_thesis})

def stud_logout(req):
    messages.info(req,"Logged Out")
    return redirect('student_login')
```

5.3 Creating models

In Django, models are the classes that represent the data in your application. They are used to define the structure of your data, as well as the relationships between different data points. Models are also used to perform CRUD (Create, Read, Update, Delete) operations on your data.

Each model in Django maps to a single database table. The fields in the model define the columns in the database table. The values in the fields are the data that is stored in the database.

Models are defined using the models.Model class. The models.Model class provides a number of methods and properties that can be used to define the structure of your data.

5.3.1 Adminapp Models

```
from django.db import models


class Faculty_Management(models.Model):
    Faculty_ID = models.AutoField(primary_key=True)
    F_ID = models.TextField(max_length=100)
    Full_Name = models.TextField(max_length=100)
    Email = models.EmailField()
    Department = models.TextField(max_length=100)
    Preferred_Subject = models.TextField(max_length=100)
    Profile = models.FileField(upload_to='Images')
    Status = models.TextField(default='pending',null=True)
    Faculty_Password = models.TextField(max_length=4)
    class Meta:
        db_table = 'faculty_management'


class Subject_Management(models.Model):
    Subject_ID = models.AutoField(primary_key=True)
    Subject_Name = models.TextField(max_length=100)
    Branch_Name = models.TextField(max_length=100)
    Semester = models.TextField(max_length=100)
    Faculty = models.TextField(max_length=50,null=True)
    Status = models.TextField(default='pending',null=True)
    class Meta:
        db_table = 'subject_management'


class Student_Management(models.Model):
    Student_ID = models.AutoField(primary_key=True)
    Full_Name = models.TextField(max_length=100)
    Roll_No = models.TextField(max_length=100)
    Email = models.EmailField()
    Branch_Name = models.TextField(max_length=100)
    Semester = models.TextField(max_length=100)
```

```
    Address = models.TextField(max_length=100)
    Phone_No = models.TextField(max_length=10)
    Profile = models.FileField(upload_to='Images')
    Student_Password = models.TextField(max_length=4)
    Alumni = models.TextField(default='Not-Alumni', null=True)
    Status = models.TextField(default='pending',null=True)
    Subject = models.TextField(max_length=20, null=True)
    Marks = models.IntegerField(default=0, null=True)
    Marks_Subject = models.TextField(max_length=50, null=True)

    class Meta:
        db_table = 'student_management'


class Marks_Management(models.Model):
    Marks_ID = models.AutoField(primary_key=True)
    Student_Branch = models.TextField(max_length=100, null=True)
    Student_Semester = models.TextField(max_length=100, null=True)
    Student_Subject = models.TextField(max_length=100, null=True)
    # Student_Profile = models.FileField(upload_to='Images')
    Student_Name = models.TextField(max_length=100)
    Student_Roll_No = models.TextField(max_length=100)
    Add_Marks = models.IntegerField(default=0, null=True)
    Stu_Foregin = models.ForeignKey(Student_Management, on_delete =
models.CASCADE, null = True)

    class Meta:
        db_table = 'marks_management'


class Library_Management(models.Model):
    Library_ID = models.AutoField(primary_key=True)
    Title = models.TextField(max_length=100)
    Author_Name = models.TextField(max_length=100)
    Publisher_Name = models.TextField(max_length=100)
```

```
    Preferred_Name = models.TextField(max_length=100)
    class Meta:
        db_table = 'library_management'


class Learning_Management(models.Model):
    Learning_ID = models.AutoField(primary_key=True)
    Course_Name = models.TextField(max_length=100, null=True)
    Price = models.TextField(max_length=100, null=True)
    Photo = models.FileField(upload_to='Images',null=True)
    Description = models.TextField(max_length=100, null=True)

    class Meta:
        db_table = 'learning_management'


class Alumni_Management(models.Model):
    Alumni_ID = models.AutoField(primary_key=True)
    Student_Name = models.TextField(max_length=100,null=True)
    Student_Profile = models.FileField(upload_to='Images',null=True)
    Student_Roll_No = models.TextField(max_length=100,null=True)
    Student_Branch = models.TextField(max_length=100,null=True)

    class Meta:
        db_table = 'alumni_management'


class Thesis_Management(models.Model):
    Thesis_ID = models.AutoField(primary_key=True)
    Thesis_Name = models.TextField(max_length=100,null=True)
    Thesis_File = models.FileField(upload_to='Images',null=True)

    class Meta:
        db_table = 'thesis_management'
```

5.3.2 Facultyapp Models

```
from django.db import models
from adminapp.models import *

class Attendence_Management(models.Model):
    Attendence_ID = models.AutoField(primary_key=True)
    Student_Name = models.TextField(max_length=100)
    Student_Roll = models.TextField(max_length=100)
    Stu_Sem = models.TextField(max_length=100)
    Stu_Branch = models.TextField(max_length=100)
    Att_Date = models.DateField(null = True)
    Att_Subject = models.TextField(max_length=20, null=True)
    Att_Status = models.TextField(max_length = 20, null=True)
    Student_Foregin = models.ForeignKey(Student_Management, on_delete =
models.CASCADE, null = True)

    class Meta:
        db_table = 'attendance_management'
```

5.4 URLs in the college project

```
from django.contrib import admin
from django.urls import path
from mainapp import views as main_views
from adminapp import views as admin_views
from facultyapp import views as faculty_views
from studentapp import views as student_views

from django.conf.urls.static import static
from django.conf import settings

#main_views
urlpatterns = [
```

```
    path('admin/', admin.site.urls),
    path('',main_views.home,name='home'),
    path('student_login', main_views.student_login, name="student_login"),
    path('faculty_login', main_views.faculty_login, name="faculty_login"),
    path('admin_login', main_views.admin_login, name="admin_login"),
    path('about',main_views.about,name='about'),
    path('contact',main_views.contact,name='contact'),

#admin_views
    path('admin_dashboard',admin_views.admin_dashboard,name='admin_dashboard'),
    path('add_faculty',admin_views.add_faculty,name='add_faculty'),
    path('manage_faculty',admin_views.manage_faculty,name='manage_faculty'),
    path('add_subject',admin_views.add_subject,name='add_subject'),
    path('manage_subject',admin_views.manage_subject,name='manage_subject'),
    path('assign_sub_faculty',admin_views.assign_sub_faculty,name='assign_sub_faculty
'),
    #
path('assign_sub_faculty/<int:id>/',admin_views.assign_sub_faculty,name='assign_sub_
faculty'),

    path('add_students',admin_views.add_students,name='add_students'),
    path('manage_student',admin_views.manage_student,name='manage_student'),
    path('admin_view_attendance',admin_views.admin_view_attendance,name='admin_vi
ew_attendance'),
    path('add_sem_marks',admin_views.add_sem_marks,name='add_sem_marks'),
    path('admin_view_marks',admin_views.admin_view_marks,name='admin_view_marks')
,
    path('add_resources',admin_views.add_resources,name='add_resources'),
    path('manage_resources',admin_views.manage_resourses,name='manage_resources
'),
    path('add_courses',admin_views.add_courses,name='add_courses'),
    path('manage_course',admin_views.manage_course,name='manage_course'),
    path('admin_add_alumni',admin_views.admin_add_alumni,name='admin_add_alumni'),
    path('manage_alumni',admin_views.manage_alumni,name='manage_alumni'),
```

```
    path('upload_thesis',admin_views.upload_thesis,name='upload_thesis'),
    path('manage_thesis',admin_views.manage_thesis,name='manage_thesis'),
    path('edit_course/<int:id>',admin_views.edit_course,name='edit_course'),
    path('edit_student/<int:id>',admin_views.edit_student,name='edit_student'),
    path('add_alumni_btn/<int:id>',admin_views.add_alumni_btn,name='add_alumni_btn')
,

    # path('add_marks',admin_views.add_marks,name='add_marks'),
    path('faculty_change_status/<int:id>',admin_views.faculty_change_status,name='fac
ulty_change_status'),
    path('faculty_Delete/<int:id>',admin_views.faculty_Delete,name='faculty_Delete'),
    path('delete_subject/<int:id>',admin_views.delete_subject,name='delete_subject'),
    path('assign_subject/<int:id>',admin_views.assign_subject,name='assign_subject'),
    path('stud_status/<int:id>',admin_views.stud_status,name='stud_status'),
    path('delete_student/<int:id>',admin_views.delete_student,name='delete_student'),
    path('delete_thesis/<int:id>',admin_views.delete_thesis,name='delete_thesis'),
    path('delete_course/<int:id>',admin_views.delete_course,name='delete_course'),
    path('delete_resourse/<int:id>',admin_views.delete_resourse,name='delete_resourse')
,
    path('delete_alumni/<int:id>',admin_views.delete_alumni,name='delete_alumni'),
    path('admin_logout',admin_views.admin_logout,name='admin_logout'),

#faculty_views

    path('faculty_dashboard',faculty_views.faculty_dashboard,name='faculty_dashboard')
,
    path('faculty_add_attendance',faculty_views.faculty_add_attendance,name='faculty_
add_attendance'),
    path('faculty_view_attendance',faculty_views.faculty_view_attendance,name='faculty
_view_attendance'),
    path('faculty_add_marks',faculty_views.faculty_add_marks,name='faculty_add_mark
s'),
    path('faculty_view_marks',faculty_views.faculty_view_marks,name='faculty_view_mar
ks'),
```

```
    # path('present_btn/<int:id>',faculty_views.present_btn,name='present_btn'),
    # path('absent_btn/<int:id>',faculty_views.absent_btn,name='absent_btn'),
    path('faculty_logout',faculty_views.faculty_logout,name='faculty_logout'),
    path('mark_attendance',faculty_views.mark_attendance,name='mark_attendance'),
    path('mark_add',faculty_views.mark_add,name='mark_add'),

#student_views

    path('student_dashboard',student_views.student_dashboard,name='student_dashboa
rd'),
    path('student_attendance_details',student_views.student_attendance_details,name='
student_attendance_details'),
    path('student_marks_details',student_views.student_marks_details,name='student_m
arks_details'),
    path('student_view_results',student_views.student_view_results,name='student_view
_results'),
    path('student_view_book',student_views.student_view_book,name='student_view_bo
ok'),
    path('student_view_courses',student_views.student_view_courses,name='student_vie
w_courses'),
    path('student_my_courses',student_views.student_my_courses,name='student_my_c
ourses'),
    path('student_alumni_details',student_views.student_alumni_details,name='student_a
lumni_details'),
    path('student_view_thesis',student_views.student_view_thesis,name='student_view_t
hesis'),
    path('add_course_btn/<int:id>',student_views.add_course_btn,name='add_course_bt
n'),
    path('stud_logout',student_views.stud_logout,name='stud_logout'),



]+ static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

5.5 HTML pages for the admin dashboard

```
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <meta name="description" content="">
  <meta name="author" content="">
  <link href="{% static 'admin_static/img/logo/logo.png' %}" rel="icon">
  <title>Admin – Dashboard</title>
  <link href="{% static 'admin_static/vendor/fontawesome-free/css/all.min.css' %}"
rel="stylesheet" type="text/css">
  <link href="{% static 'admin_static/vendor/bootstrap/css/bootstrap.min.css' %}"
rel="stylesheet" type="text/css">
  <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>

  <link href="{% static 'admin_static/css/ruang-admin.min.css' %}" rel="stylesheet">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css"></head>

<body id="page-top">
    <!-- swal message start -->

    {% if messages %}
    {% for message in messages %}
    {% if message.level == DEFAULT_MESSAGE_LEVELS.SUCCESS %}
    <script>swal({
            title: "Success!",
            text: "{{message}}",
```

```
            icon: "success",
            button: "OK",
        });
    </script>

{% elif message.level == DEFAULT_MESSAGE_LEVELS.WARNING %}
    <script>swal({
            title: "Warning :)",
            text: "{{message}}",
            icon: "warning",
            button: "OK",

        });
    </script>
    {% elif message.level == DEFAULT_MESSAGE_LEVELS.INFO %}
    <script>swal({
            title: "info :)",
            text: "{{message}}",
            icon: "info",
            button: "OK",
        });
    </script>
    {% elif message.level == DEFAULT_MESSAGE_LEVELS.ERROR %}
    <script>swal({
            title: "error :)",
            text: "{{message}}",
            icon: "error",
            button: "OK",
        });
    </script>
    {% endif %}
    {% endfor %}
    {% endif %}
```

```
    <!-- swal message ends -->
  <div id="wrapper">
    <!-- Sidebar -->
    <ul class="navbar-nav sidebar sidebar-light accordion" id="accordionSidebar">
      <a class="sidebar-brand d-flex align-items-center justify-content-center"
href="{% url 'admin_dashboard' %}">
        <div class="sidebar-brand-icon">
          <!-- <img src="img/logo/logo2.png"> -->
          <h3>ADMIN</h3>
        </div>
        <!-- <div class="sidebar-brand-text mx-3">RuangAdmin</div> -->
      </a>
      <hr class="sidebar-divider my-0">
      <li class="nav-item active">
        <a class="nav-link" href="{% url 'admin_dashboard' %}">
          <i class="fas fa-fw fa-tachometer-alt"></i>
          <span>Dashboard</span></a>
      </li>
      <hr class="sidebar-divider">
      <div class="sidebar-heading">
        Features
      </div>
      <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseBootstrap"
          aria-expanded="true" aria-controls="collapseBootstrap">
          <i class="far fa-fw fa-window-maximize"></i>
          <span>Faculty Management</span>
        </a>
        <div id="collapseBootstrap" class="collapse" aria-labelledby="headingBootstrap"
data-parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Bootstrap UI</h6> -->
```

```
          <a class="collapse-item" href="{% url 'add_faculty' %}">Add New
Faculty</a>
          <a class="collapse-item" href="{% url 'manage_faculty' %}">Manage
Faculty</a>
        </div>
      </div>
    </li>
    <li class="nav-item">
      <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseForm" aria-expanded="true"
        aria-controls="collapseForm">
        <i class="fab fa-fw fa-wpforms"></i>
        <span>Subject Management</span>
      </a>
      <div id="collapseForm" class="collapse" aria-labelledby="headingForm" data-
parent="#accordionSidebar">
        <div class="bg-white py-2 collapse-inner rounded">
          <!-- <h6 class="collapse-header">Forms</h6> -->
          <a class="collapse-item" href="{% url 'add_subject' %}">Add Subject</a>
          <a class="collapse-item" href="{% url 'manage_subject' %}">Manage
Subject</a>
          <a class="collapse-item" href="{% url 'assign_sub_faculty' %}">Assign
Subject to Faculty</a>
        </div>
      </div>
    </li>
    <li class="nav-item active">
      <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTable" aria-expanded="true"
        aria-controls="collapseTable">
        <i class="fas fa-fw fa-table"></i>
        <span>Student Management</span>
      </a>
```

```
        <div id="collapseTable" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <h6 class="collapse-header">Tables</h6>
            <a class="collapse-item active" href="{% url 'add_students' %}">Add
Student</a>
            <a class="collapse-item" href="{% url 'manage_student' %}">Manage
Student</a>
          </div>
        </div>
      </li>

      <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTabl" aria-expanded="true"
          aria-controls="collapseTabl">
          <i class="fas fa-fw fa-table"></i>
          <span>Attendance Management</span>
        </a>
        <div id="collapseTabl" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'admin_view_attendance' %}">View
Attendance Deatils</a>
          </div>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTab" aria-expanded="true"
          aria-controls="collapseTab">
          <i class="fas fa-fw fa-table"></i>
          <span>Marks Management</span>
```

```
        </a>
        <div id="collapseTab" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'add_sem_marks' %}">Add Semester
Marks </a>
            <a class="collapse-item" href="{% url 'admin_view_marks' %}">View Marks
Deatils</a>
          </div>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTa" aria-expanded="true"
          aria-controls="collapseTa">
          <i class="fas fa-fw fa-table"></i>
          <span>Library Management</span>
        </a>
        <div id="collapseTa" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'add_resources' %}">Add
Resources</a>
            <a class="collapse-item" href="{% url 'manage_resources' %}">Manage
Resources</a>
          </div>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseT" aria-expanded="true"
          aria-controls="collapseT">
```

```
          <i class="fas fa-fw fa-table"></i>
          <span>Learning Management</span>
        </a>
        <div id="collapseT" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'add_courses' %}">Add New
Course</a>
            <a class="collapse-item" href="{% url 'manage_course' %}">Manage
Courses</a>
          </div>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTab1" aria-expanded="true"
          aria-controls="collapseTab1">
          <i class="fas fa-fw fa-table"></i>
          <span>Alumni Management</span>
        </a>
        <div id="collapseTab1" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'admin_add_alumni' %}">Add Alumni
Details </a>
            <a class="collapse-item" href="{% url 'manage_alumni' %}">Manage Alumni
Deatils</a>
          </div>
        </div>
      </li>
      <li class="nav-item">
```

```
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTab2" aria-expanded="true"
          aria-controls="collapseTab2">
          <i class="fas fa-fw fa-table"></i>
          <span>Thesis Management</span>
        </a>
        <div id="collapseTab2" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'upload_thesis' %}">Upload Thesis
Details </a>
            <a class="collapse-item" href="{% url 'manage_thesis' %}">Manage Thesis
Deatils</a>
          </div>
        </div>
      </li>
    </ul>
    <!-- Sidebar -->
    <div id="content-wrapper" class="d-flex flex-column">
      <div id="content">
        <!-- TopBar -->
        <nav class="navbar navbar-expand navbar-light bg-navbar topbar mb-4 static-
top">
          <button id="sidebarToggleTop" class="btn btn-link rounded-circle mr-3">
            <i class="fa fa-bars"></i>
          </button>
          <ul class="navbar-nav ml-auto">
            <a href="{% url 'admin_logout' %}">
              <button class="btn btn-warning"> Logout </button>
            </a>
          </ul>
        </nav>
        <!-- Topbar -->
```

```
        <!-- Container Fluid-->
        <div class="container-fluid" id="container-wrapper">
          <div class="d-sm-flex align-items-center justify-content-between mb-4">
            <ol class="breadcrumb">
              <li class="breadcrumb-item"><a>Student Management</a></li>
              <li class="breadcrumb-item active" aria-current="page">Add Student</li>
            </ol>
          </div>

          <div class="row mb-5">
            <div class="col-8 card m-auto">
               <form method="post" enctype = 'multipart/form-data' class="mb-4 mt-
4">
                {% csrf_token %}
                  <h2 class="text-center">Add Student</h2><hr><br>
                  <div class="d-flex col-12">
                    <div class="d-flex flex-column col-6">
                       <label style="color:black" for="">Full Name</label>
                       <input name="stu_fname" class="form-control" type="text"
placeholder="Enter Student Name" required>
                    </div>
                    <div class="d-flex flex-column col-6">
                       <label style="color:black" for="">RollNumber </label>
                       <input class="form-control" type="text" name="stu_roll_no" id=""
placeholder="Enter Student Rollnumber " required>
                    </div>
                  </div><br>
                  <div class="d-flex col-12">
                    <div class="d-flex flex-column col-6">
                       <label style="color:black" for="">Email</label>
                       <input name="stu_email" class="form-control" type="email"
placeholder="Enter Student Email" required>
                    </div>
```

```
                    <div class="d-flex flex-column col-6">
                        <label style="color:black" for="">Branch </label>
                        <select class = "form-control"  name="stu_branch" id="" required>
                          <option value="" selected disabled>Select Branch</option>
                          <option value="CIVIL">CIVIL</option>
                          <option value="CSE">CSE</option>
                          <option value="MECH">MECH</option>
                          <option value="EEE">EEE</option>
                          <option value="ECE">ECE</option>
                        </select>
                      </div>
                  </div><br>
                  <div class="d-flex col-12">
                    <div class="d-flex flex-column col-6">
                        <label style="color:black" for="">Semester</label>
                        <select class = "form-control"  name="stu_semester" id=""
required>
                          <option value="" selected disabled>Select Semester</option>
                          <option value="semester-1">Semister-1</option>
                          <option value="semester-2">Semister-2</option>
                          <option value="semester-3">Semister-3</option>
                          <option value="semester-4">Semister-4</option>
                          <option value="semester-5">Semister-5</option>
                          <option value="semester-6">Semister-6</option>
                          <option value="semester-7">Semister-7</option>
                          <option value="semester-8">Semister-8</option>
                        </select>
                      </div>
                    <div class="d-flex flex-column col-6">
                        <label style="color:black" for="">Address </label>
                        <input class="form-control" type="text" name="stu_address" id=""
placeholder="Enter Student Address " required>
                    </div>
                  </div><br>
```

```
                    <div class="d-flex col-12">
                      <div class="d-flex flex-column col-6">
                          <label style="color:black" for="">Phone Number </label>
                          <input name="stu_phone" class="form-control" type="text"
placeholder="Enter Student Contact" required>
                      </div>
                      <div class="d-flex flex-column col-6">
                          <label style="color:black" for="">Profile </label>
                          <input class="form-control" type="file" name="stu_profile" >
                      </div>
                    </div><br>

                    <input type="submit" class="btn btn-primary d-flex m-auto" name=""
id="" value="Submit">
                  </form>
              </div>
            </div>

        </div>
        <!---Container Fluid-->
      </div>
      <!-- Footer -->
      <footer class="sticky-footer bg-white">
        <div class="container my-auto">
          <div class="copyright text-center my-auto">
            <span>copyright &copy; <script> document.write(new Date().getFullYear());
</script> - developed by
              <b><a href="https://www.codebook.in/"
target="_blank">Codebook.in</a></b>
            </span>
          </div>
        </div>
      </footer>
      <!-- Footer -->
```

```
    </div>
  </div>

  <!-- Scroll to top -->
  <a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
  </a>

  <script src="{% static 'admin_static/vendor/jquery/jquery.min.js' %}"></script>
  <script src="{% static
'admin_static/vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
  <script src="{% static 'admin_static/vendor/jquery-
easing/jquery.easing.min.js' %}"></script>
  <script src="{% static 'admin_static/js/ruang-admin.min.js' %}"></script>
  <script src="{% static 'admin_static/vendor/chart.js/Chart.min.js' %}"></script>
  <script src="{% static 'admin_static/js/demo/chart-area-demo.js' %}"></script>
</body>

</html>
```

5.6 HTML pages for faculty dashboard

```
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <meta name="description" content="">
  <meta name="author" content="">
  <link href="{% static 'admin_static/img/logo/logo.png' %}" rel="icon">
  <title>Faculty - Dashboard</title>
```

```
  <link href="{% static 'admin_static/vendor/fontawesome-free/css/all.min.css' %}"
rel="stylesheet" type="text/css">
  <link href="{% static 'admin_static/vendor/bootstrap/css/bootstrap.min.css' %}"
rel="stylesheet" type="text/css">
  <link href="{% static 'admin_static/css/ruang-admin.min.css' %}" rel="stylesheet">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>

</head>

<body id="page-top">

    <!-- swal message start -->

    {% if messages %}
    {% for message in messages %}
    {% if message.level == DEFAULT_MESSAGE_LEVELS.SUCCESS %}
    <script>swal({
            title: "Success!",
            text: "{{message}}",
            icon: "success",
            button: "OK",
        });
    </script>

    {% elif message.level == DEFAULT_MESSAGE_LEVELS.WARNING %}
    <script>swal({
            title: "Warning :)",
            text: "{{message}}",
            icon: "warning",
            button: "OK",

        });
```

```
    </script>
    {% elif message.level == DEFAULT_MESSAGE_LEVELS.INFO %}
    <script>swal({
            title: "info :)",
            text: "{{message}}",
            icon: "info",
            button: "OK",
        });
    </script>
    {% elif message.level == DEFAULT_MESSAGE_LEVELS.ERROR %}
    <script>swal({
            title: "error :)",
            text: "{{message}}",
            icon: "error",
            button: "OK",
        });
    </script>
    {% endif %}
    {% endfor %}
    {% endif %}

    <!-- swal message ends -->

  <div id="wrapper">
    <!-- Sidebar -->
    <ul class="navbar-nav sidebar sidebar-light  accordion" id="accordionSidebar">
      <a class="sidebar-brand d-flex align-items-center justify-content-center"
style="background-color: green;" href="{% url 'faculty_dashboard' %}">
        <div class="sidebar-brand-icon ">
          <!-- <img src="img/logo/logo2.png"> -->
          <h4 class="mt-3">Faculty ID</h4>
          <p>123456A</p>
        </div>
        <!-- <div class="sidebar-brand-text mx-3">RuangAdmin</div> -->
```

```
    </a>
    <hr class="sidebar-divider my-0">
    <li class="nav-item active">
      <a class="nav-link" href="{% url 'faculty_dashboard' %}">
        <i class="fas fa-fw fa-tachometer-alt"></i>
        <span>Dashboard</span></a>
    </li>
    <hr class="sidebar-divider">

    <li class="nav-item active">
      <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseBootstrap"
        aria-expanded="true" aria-controls="collapseBootstrap">
        <i class="far fa-fw fa-window-maximize"></i>
        <span>Attendance Management</span>
      </a>
      <div id="collapseBootstrap" class="collapse" aria-labelledby="headingBootstrap"
data-parent="#accordionSidebar">
        <div class="bg-white py-2 collapse-inner rounded">
          <!-- <h6 class="collapse-header">Bootstrap UI</h6> -->
          <a class="collapse-item active" href="{% url
'faculty_add_attendance' %}">Add Attendance</a>
          <a class="collapse-item" href="{% url 'faculty_view_attendance' %}">View
Attendance Deatils</a>
        </div>
      </div>
    </li>
    <li class="nav-item">
      <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseForm" aria-expanded="true"
        aria-controls="collapseForm">
        <i class="fab fa-fw fa-wpforms"></i>
        <span>Marks Management</span>
      </a>
```

```
        <div id="collapseForm" class="collapse" aria-labelledby="headingForm" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Forms</h6> -->
            <a class="collapse-item" href="{% url 'faculty_add_marks' %}">Add
Marks</a>
            <a class="collapse-item" href="{% url 'faculty_view_marks' %}">View
Marks</a>
          </div>
        </div>
      </li>


    </ul>
    <!-- Sidebar -->
    <div id="content-wrapper" class="d-flex flex-column">
      <div id="content">
        <!-- TopBar -->
        <nav class="navbar navbar-expand navbar-light bg-navbar bg-success topbar
mb-4 static-top">
          <button id="sidebarToggleTop" class="btn btn-link rounded-circle mr-3">
            <i class="fa fa-bars"></i>
          </button>
          <ul class="navbar-nav ml-auto">
            <a href="{% url 'faculty_logout' %}">
              <button class="btn btn-warning"> Logout </button>
            </a>
          </ul>
        </nav>
        <!-- Topbar -->

        <!-- Container Fluid-->
        <div class="container-fluid" id="container-wrapper">
          <div class="d-sm-flex align-items-center justify-content-between mb-4">
            <ol class="breadcrumb">
```

```
            <li class="breadcrumb-item"><p >Attendance Management</p></li>
            <li class="breadcrumb-item active" aria-current="page">Add
Attendance</li>
          </ol>
        </div>


          <div class="row ">
            <div class=" col-12 ">
              <form action="faculty_add_attendance" method="post" enctype =
'multipart/form-data'>
                {% csrf_token %}
                <div class="d-flex justify-content-center">
                  <div class="col-4">
                    <label style="color:black" for="">Branch </label>
                    <select class = "form-control"  name="branch" id="" required>
                      <option value="" selected disabled>Select Branch</option>
                      <option value="CIVIL">CIVIL</option>
                      <option value="CSE">CSE</option>
                      <option value="MECH">MECH</option>
                      <option value="EEE">EEE</option>
                      <option value="ECE">ECE</option>
                    </select>
                  </div>
                  <div class="col-4">
                    <label style="color:black" for="">Semester </label>
                    <select class = "form-control"  name="semester" id="" required>
                      <option value="" selected disabled>Select Semester</option>
                      <option value="semester-1">Semister-1</option>
                          <option value="semester-2">Semister-2</option>
                          <option value="semester-3">Semister-3</option>
                          <option value="semester-4">Semister-4</option>
                          <option value="semester-5">Semister-5</option>
                          <option value="semester-6">Semister-6</option>
```

```
                        <option value="semester-7">Semister-7</option>
                        <option value="semester-8">Semister-8</option>
                    </select>
                  </div>
                  <div class="col-4">
                    <label style="color:black" for="">Subject </label>
                    <select class = "form-control"  name="subject" id="" required>
                      <option value="" selected disabled>Select Subject</option>
                      {% for j in subjj %}
                      <option >{{j.Subject_Name}}</option>
                      {% endfor %}
                    </select>
                  </div>
                </div><br>
                <input type="submit" class="btn btn-primary d-flex m-auto"
value="Submit">
              </form><br>
            </div>


            <div class="col-lg-12">
              <div class="card mb-4">
                <form action="{% url 'mark_attendance' %}" method="post" enctype =
'multipart/form-data'>
                  {% csrf_token %}
                  <div class="table-responsive p-3">
                    <table class="table align-items-center table-flush table-hover"
id="dataTableHover">
                      <thead class="thead-light">
                        <tr class="text-center">
                          <th>#</th>
                          <th>Student Profile</th>
                          <th>Student Name</th>
                          <th>Student Rollnumber</th>
                          <th>Attendance Date</th>
```

```
                        <th>Action</th>
                      </tr>
                    </thead>
                    <tbody class="text-center">
                     {% for i in stu_details %}
                     <tr>
                        <td>{{forloop.counter}}</td>
                        <td><img src="media/{{i.Profile}}" style="border-radius: 50%;
width:50px; height:50px"></td>
                        <td>{{i.Full_Name}}</td>
                        <td>{{i.Roll_No}}</td>
                        <td>{{date}}</td>
                        <td>
                            <input type="checkbox" name ="{{i.Student_ID}}" value =
"present"> Present
                            <input type="checkbox" name ="{{i.Student_ID}}" value =
"absent"> Absent
                            {% comment %} <input type="hidden" name ="check" value =
"i.Student_ID"> Absent {% endcomment %}
                        </td>
                      </tr>
                      {% endfor %}
                     </tbody>
                   </table><br>
                   <input type="submit" class="btn btn-primary d-flex m-
auto"  value="Submit">
                  </div>
                 </form>
               </div>
              </div>
             </div>

          <!--Row-->
```

```
        <!-- Modal Logout -->
        <div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabelLogout"
          aria-hidden="true">
          <div class="modal-dialog" role="document">
            <div class="modal-content">
              <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabelLogout">Ohh No!</h5>
                <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
                  <span aria-hidden="true">&times;</span>
                </button>
              </div>
              <div class="modal-body">
                <p>Are you sure you want to logout?</p>
              </div>
              <div class="modal-footer">
                <button type="button" class="btn btn-outline-primary" data-
dismiss="modal">Cancel</button>
                <a href="login.html" class="btn btn-primary">Logout</a>
              </div>
            </div>
          </div>
        </div>

      </div>
      <!---Container Fluid-->
    </div>
    <!-- Footer -->
    <footer class="sticky-footer bg-white">
      <div class="container my-auto">
        <div class="copyright text-center my-auto">
          <span>copyright &copy; <script> document.write(new Date().getFullYear());
</script> - developed by
```

```
            <b><a href="https://www.codebook.in/"
target="_blank">Codebook.in</a></b>
          </span>
        </div>
      </div>


    </footer>
    <!-- Footer -->
  </div>
 </div>

 <!-- Scroll to top -->
 <a class="scroll-to-top rounded" href="#page-top">
  <i class="fas fa-angle-up"></i>
 </a>

 <script src="{% static 'admin_static/vendor/jquery/jquery.min.js' %}"></script>
 <script src="{% static
'admin_static/vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
 <script src="{% static 'admin_static/vendor/jquery-
easing/jquery.easing.min.js' %}"></script>
 <script src="{% static 'admin_static/js/ruang-admin.min.js' %}"></script>
 <script src="{% static 'admin_static/vendor/chart.js/Chart.min.js' %}"></script>
 <script src="{% static 'admin_static/js/demo/chart-area-demo.js' %}"></script>
</body>

</html>
```

5.7 HTML pages for student dashboard

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <meta name="description" content="">
  <meta name="author" content="">
  <link href="{% static 'admin_static/img/logo/logo.png' %}" rel="icon">
  <title>Student – Dashboard</title>
  <link href="{% static 'admin_static/vendor/fontawesome-free/css/all.min.css' %}"
rel="stylesheet" type="text/css">
  <link href="{% static 'admin_static/vendor/bootstrap/css/bootstrap.min.css' %}"
rel="stylesheet" type="text/css">
  <link href="{% static 'admin_static/css/ruang-admin.min.css' %}" rel="stylesheet">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet"
type="text/css">
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>

</head>

<body id="page-top">
    <!-- swal message start -->

    {% if messages %}
    {% for message in messages %}
    {% if message.level == DEFAULT_MESSAGE_LEVELS.SUCCESS %}
    <script>swal({
            title: "Success!",
            text: "{{message}}",
            icon: "success",
            button: "OK",
        });
```

```
  </script>

  {% elif message.level == DEFAULT_MESSAGE_LEVELS.WARNING %}
  <script>swal({
          title: "Warning :)",
          text: "{{message}}",
          icon: "warning",
          button: "OK",

      });
  </script>
  {% elif message.level == DEFAULT_MESSAGE_LEVELS.INFO %}
  <script>swal({
          title: "info :)",
          text: "{{message}}",
          icon: "info",
          button: "OK",
      });
  </script>
  {% elif message.level == DEFAULT_MESSAGE_LEVELS.ERROR %}
  <script>swal({
          title: "error :)",
          text: "{{message}}",
          icon: "error",
          button: "OK",
      });
  </script>
  {% endif %}
  {% endfor %}
  {% endif %}

  <!-- swal message ends -->
<div id="wrapper">
  <!-- Sidebar -->
```

```
<ul class="navbar-nav sidebar sidebar-light accordion" id="accordionSidebar">
  <a class="sidebar-brand d-flex align-items-center justify-content-center"
style="background-color: rgb(230, 102, 51);" href="{% url 'student_dashboard' %}">
    <div class="sidebar-brand-icon">
      <!-- <img src="img/logo/logo2.png"> -->
      <h5>Student RollNumber</h5>
    </div>
    <!-- <div class="sidebar-brand-text mx-3">RuangAdmin</div> -->
  </a>
  <hr class="sidebar-divider my-0">
  <li class="nav-item active">
    <a class="nav-link" href="{% url 'student_dashboard' %}">
      <i class="fas fa-fw fa-tachometer-alt"></i>
      <span>Dashboard</span></a>
  </li>
  <hr class="sidebar-divider">

  <li class="nav-item active">
    <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseBootstrap"
      aria-expanded="true" aria-controls="collapseBootstrap">
      <i class="far fa-fw fa-window-maximize"></i>
      <span>Attendance Management</span>
    </a>
    <div id="collapseBootstrap" class="collapse" aria-labelledby="headingBootstrap"
data-parent="#accordionSidebar">
      <div class="bg-white py-2 collapse-inner rounded">
        <!-- <h6 class="collapse-header">Bootstrap UI</h6> -->
        <a class="collapse-item active" href="{% url
'student_attendance_details' %}">View Attendance Details</a>
      </div>
    </div>
  </li>
  <li class="nav-item">
```

```
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseForm" aria-expanded="true"
          aria-controls="collapseForm">
          <i class="fab fa-fw fa-wpforms"></i>
          <span>Marks Management</span>
        </a>
        <div id="collapseForm" class="collapse" aria-labelledby="headingForm" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Forms</h6> -->
            <a class="collapse-item" href="{% url 'student_marks_details' %}">View
Marks Details </a>
          </div>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTable" aria-expanded="true"
          aria-controls="collapseTable">
          <i class="fas fa-fw fa-table"></i>
          <span>Result Management</span>
        </a>
        <div id="collapseTable" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'student_view_results' %}">View
Results</a>
          </div>
        </div>
      </li>

      <li class="nav-item">
```

```
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTabl" aria-expanded="true"
          aria-controls="collapseTabl">
          <i class="fas fa-fw fa-book"></i>
          <span>Library Management </span>
        </a>
        <div id="collapseTabl" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'student_view_book' %}">View
Books</a>
          </div>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTab" aria-expanded="true"
          aria-controls="collapseTab">
          <i class="material-icons" style="font-size: 18px;">local_library</i>
          <span>Learning Management </span>
        </a>
        <div id="collapseTab" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'student_view_courses' %}">View
Courses </a>
            <a class="collapse-item" href="{% url 'student_my_courses' %}">My
Course</a>
          </div>
        </div>
      </li>
      <li class="nav-item">
```

```
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTa" aria-expanded="true"
          aria-controls="collapseTa">
          <i class="material-icons" style="font-size: 18px;">school</i>
          <span>Alumni Management</span>
        </a>
        <div id="collapseTa" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'student_alumni_details' %}">View
Alumni Details</a>
          </div>
        </div>
      </li>

      <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapseTab2" aria-expanded="true"
          aria-controls="collapseTab2">
          <i class="material-icons" style="font-size: 18px;">history_edu</i>
          <span>Thesis Management</span>
        </a>
        <div id="collapseTab2" class="collapse" aria-labelledby="headingTable" data-
parent="#accordionSidebar">
          <div class="bg-white py-2 collapse-inner rounded">
            <!-- <h6 class="collapse-header">Tables</h6> -->
            <a class="collapse-item" href="{% url 'student_view_thesis' %}">View Thesis
</a>
          </div>
        </div>
      </li>
    </ul>
    <!-- Sidebar -->
```

```
    <div id="content-wrapper" class="d-flex flex-column">
      <div id="content">
        <!-- TopBar -->
        <nav class="navbar navbar-expand navbar-light bg-navbar topbar mb-4 static-
top" style="background-color: rgb(248, 160, 59);">
          <button id="sidebarToggleTop" class="btn btn-link rounded-circle mr-3">
            <i class="fa fa-bars"></i>
          </button>
          <ul class="navbar-nav ml-auto">

            <a href="{% url 'stud_logout' %}">
              <button class="btn text-light" style="background-color: rgb(226, 79, 79);">
Logout </button>
            </a>
          </ul>
        </nav>
        <!-- Topbar -->

        <!-- Container Fluid-->
        <div class="container-fluid" id="container-wrapper">
          <div class="d-sm-flex align-items-center justify-content-between mb-4">
            <!-- <h1 class="h3 mb-0 text-gray-800">Dashboard</h1> -->
            <ol class="breadcrumb">
              <li class="breadcrumb-item"><p >Attendance Management</p></li>
              <li class="breadcrumb-item active" aria-current="page">View Attendance
Details</li>
            </ol>
          </div>

          <div class="row ">
            <div class=" col-12 ">
              <form method="post" enctype = 'multipart/form-data'>
                <div class="d-flex justify-content-center">
                  <div class="col-4">
```

```
                <label style="color:black" for="">Semester </label>
                <select class = "form-control"  name="semester" id="" required>
                  <option value="" selected disabled>Select Semester</option>
                  <option value="semester-1">Semister-1</option>
                      <option value="semester-2">Semister-2</option>
                      <option value="semester-3">Semister-3</option>
                      <option value="semester-4">Semister-4</option>
                      <option value="semester-5">Semister-5</option>
                      <option value="semester-6">Semister-6</option>
                      <option value="semester-7">Semister-7</option>
                      <option value="semester-8">Semister-8</option>
                </select>
              </div>
              <div class="col-4">
                <label style="color:black" for="">Subject </label>
                <select class = "form-control"  name="subject" id="" required>
                  <option value="" selected disabled>Select Subject</option>
                  <option value="CIVIL">Python</option>
                  <option value="CSE">MATHS</option>
                  <option value="MECH">ENGLISH</option>
                  <option value="EEE">JAVA</option>
                  <option value="ECE">DJANGO</option>
                </select>
              </div>
            </div><br>
            <input type="submit" class="btn btn-primary d-flex m-auto"
style="height:3%;" value="Submit">
          </form><br>
        </div>


        <div class="col-lg-10 m-auto">
          <div class="card mb-4">
            <div class="table-responsive p-3">
```

```
                <table class="table align-items-center table-flush table-hover"
id="dataTableHover">
                  <thead class="thead-light">
                    <tr class="text-center">
                      <th>#</th>
                      <th>Date</th>
                      <th>Status</th>
                    </tr>
                  </thead>
                  <tbody class="text-center">
                    <tr>
                      <td>1</td>
                      <td>01/05/2023</td>
                      <td class="text-success">Present</td>
                    </tr>
                  </tbody>
                </table><br>
              </div>
            </div>
          </div>
        </div>
        <!--Row-->



        <!-- Modal Logout -->
        <div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabelLogout"
          aria-hidden="true">
          <div class="modal-dialog" role="document">
            <div class="modal-content">
              <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabelLogout">Ohh No!</h5>
```

```
                <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
                  <span aria-hidden="true">&times;</span>
                </button>
              </div>
              <div class="modal-body">
                <p>Are you sure you want to logout?</p>
              </div>
              <div class="modal-footer">
                <button type="button" class="btn btn-outline-primary" data-
dismiss="modal">Cancel</button>
                <a href="login.html" class="btn btn-primary">Logout</a>
              </div>
            </div>
          </div>
        </div>

      </div>
      <!---Container Fluid-->
    </div>
    <!-- Footer -->
    <footer class="sticky-footer bg-white">
      <div class="container my-auto">
        <div class="copyright text-center my-auto">
          <span>copyright &copy; <script> document.write(new Date().getFullYear());
</script> - developed by
            <b><a href="https://www.codebook.in/"
target="_blank">Codebook.in</a></b>
          </span>
        </div>
      </div>
    </footer>
    <!-- Footer -->
  </div>
```

```
  </div>

  <!-- Scroll to top -->
  <a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
  </a>

  <script src="{% static 'admin_static/vendor/jquery/jquery.min.js' %}"></script>
  <script src="{% static
'admin_static/vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
  <script src="{% static 'admin_static/vendor/jquery-
easing/jquery.easing.min.js' %}"></script>
  <script src="{% static 'admin_static/js/ruang-admin.min.js' %}"></script>
  <script src="{% static 'admin_static/vendor/chart.js/Chart.min.js' %}"></script>
  <script src="{% static 'admin_static/js/demo/chart-area-demo.js' %}"></script>
</body>

</html>
```

5.8 SCREENSHOTS

5.8.1 Login pages

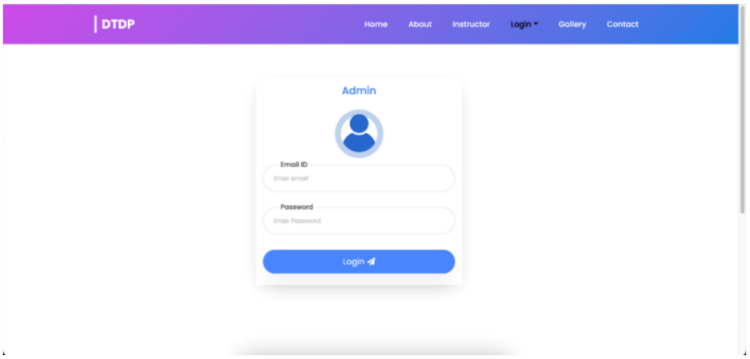


Figure 5.8.1.1 Admin login page
This is the login page for admin when admin can log into the dashboard by entering the credentials.

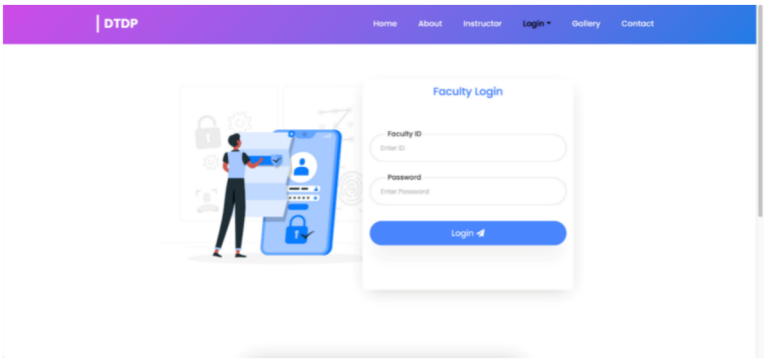


Figure 5.8.1.2 Faculty Login page
This is the login page for faculty to log into the dashboard by entering faculty ID and password

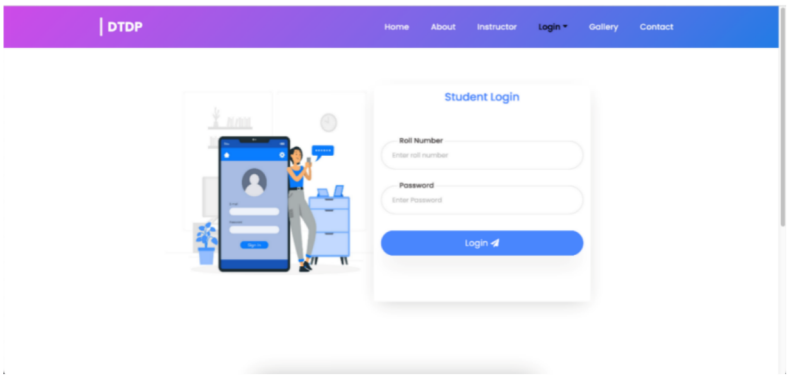
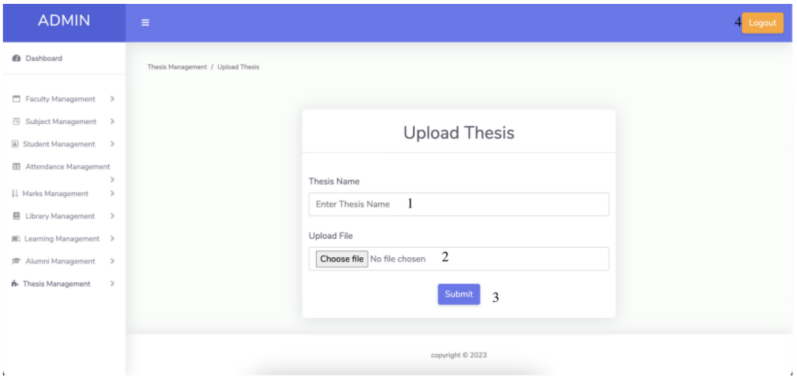


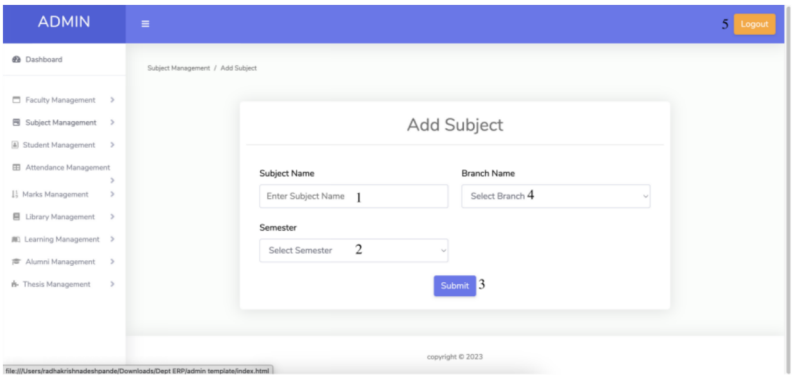
Figure 5.8.1.3 Student Login page
This is the login page for students where students can log into their dashboard by entering email and password.

5.8.2 Admin dashboard pages



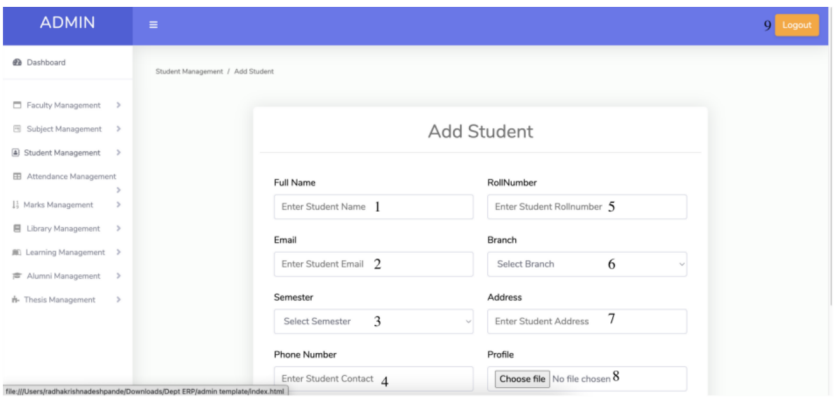
1.We have to enter the thesis name here.
2.We have to choose the file which we wanted to upload.
3.We have to submit the file which we uploaded.
4.Logout .

Figure 5.8.2.1 Upload thesis



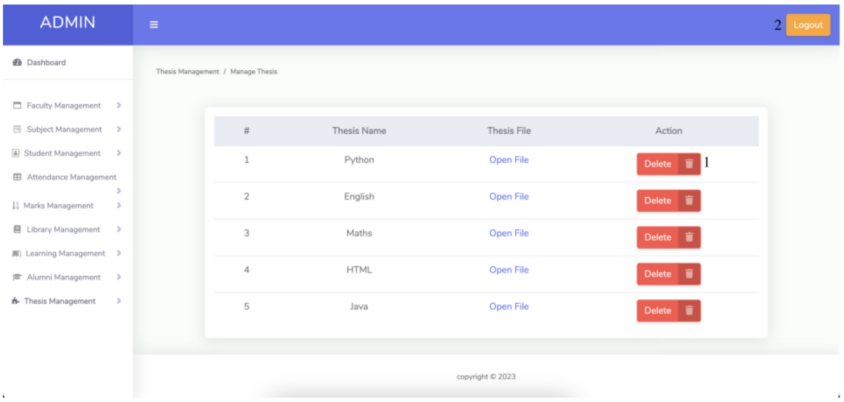
1.We have to enter the subject name.
2.We have to choose which semester student is styding .
3.We have to submit after the selection.
4.We have to select the branch by default it is DTDP.
5.Logout .

Figure5.8.2.2 Add subject



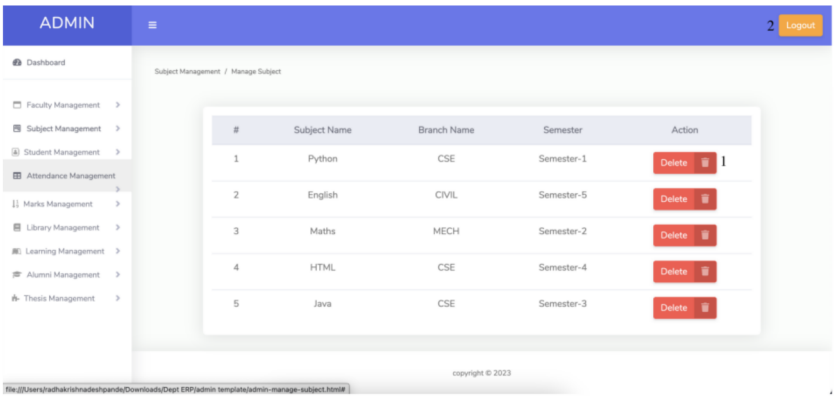
1.We have to enter the student name.
2.We have to enter student email.
3.We have to select the semester.
4.We have to enter the student contact number.
5. We have to enter the student roll number..
6.We have to select the branch by default it is DTDP.
7.We have to enter the student Address .
8.We have to Choose the profile pic of the student.
9.Logout

Figure 5.8.2.3 Add student



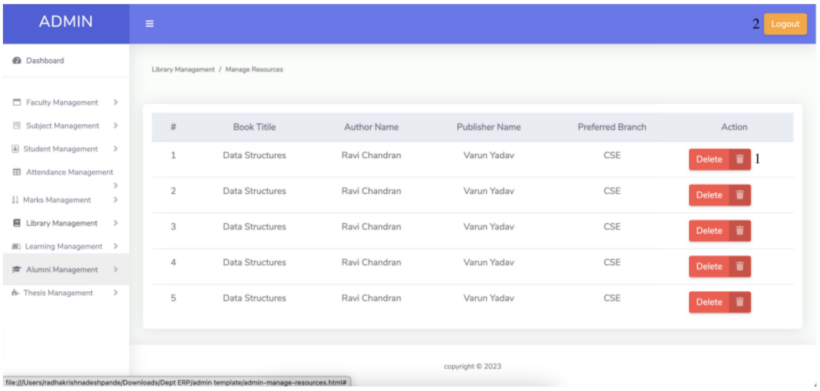
1.We can delete the Thesis.
2.Logout

Figure 5.8.2.4 Manage thesis



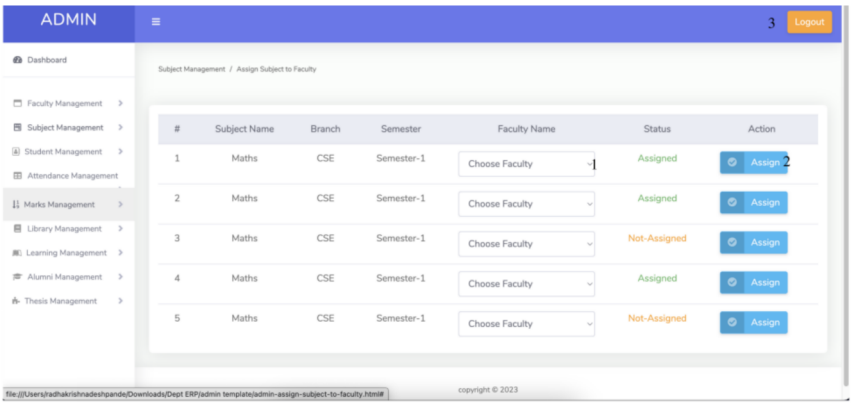
1.We can delete the Subjects.
2.Logout

Figure 5.8.2.5 Manage subjects



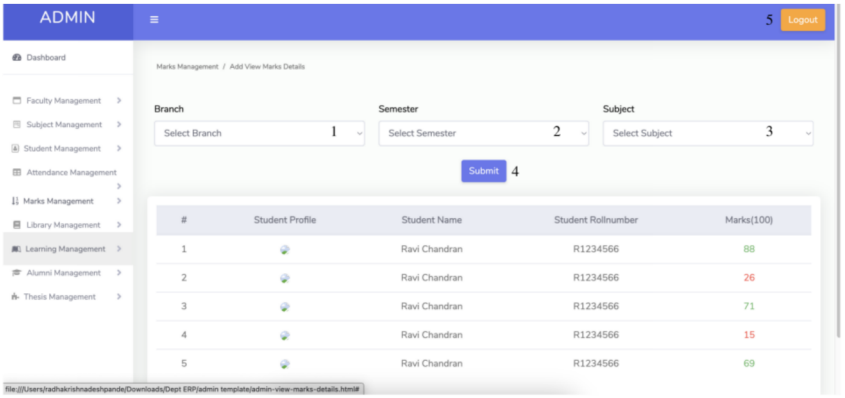
1.We can delete the Library Resources
2.Logout

Figure 5.8.2.6 Manage resources



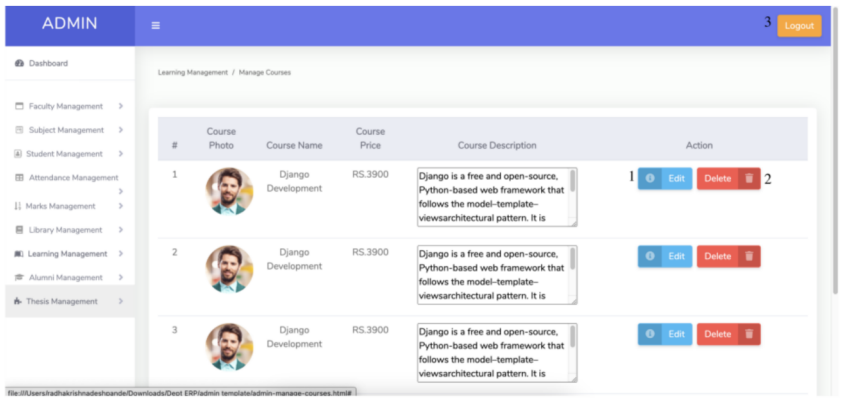
1.We can Choose the Faculty
2.We can assign by click button.
3.Logout

Figure5.8.2.7 Assign subject to faculty



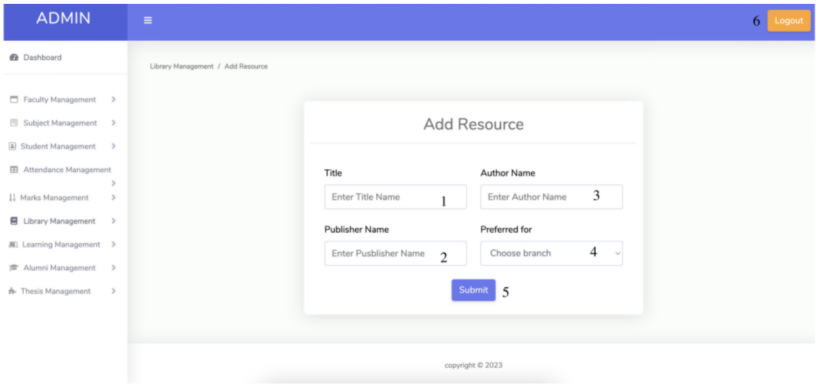
1.We can Choose the Branch by default it is DTDP.
2.We select the semester .
3.We select the subject.
4.If we submit the customized selections we can see the marks which have entered.
5.Logout

Figure 5.8.2.8 View marks



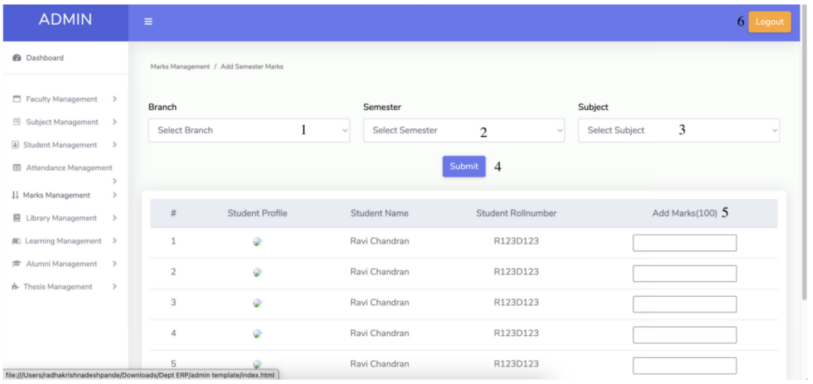
1.We can Edit the Course.
2.We can Delete the Course.
3.Logout.

Figure 5.8.2.9 Manage reourses



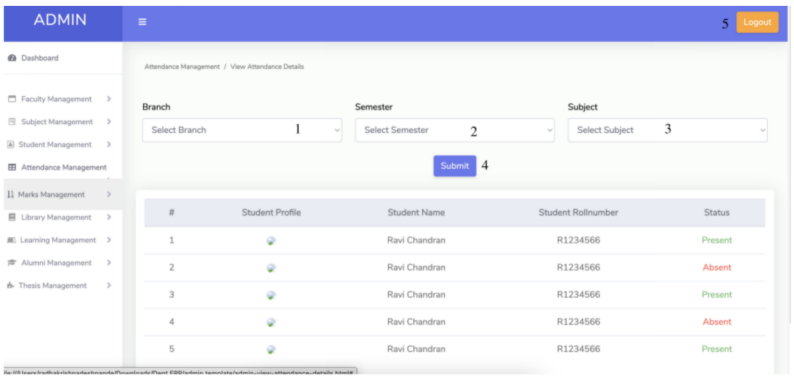
1.We have to add the title name.
2.We have to add the publisher name.
3.We have to add the author name.
4.We have to select the branch.
5.If we submit the resource to it the data will store in database.
6.Logout.

Figure 5.8.2.10 Add resources



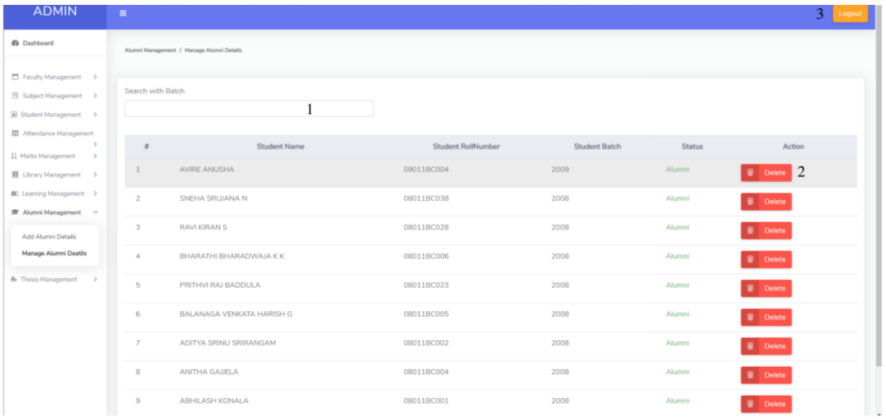
1.We have to select the branch by default DTDP.
2.We have to select the semester in which we wanted to add the marks.
3.We have to select the subject .
4.If we submit the given selection we redirect to the students of that section.
5.We have to add the marks.
6.Logout.

Figure 5.8.2.11 Add semester marks



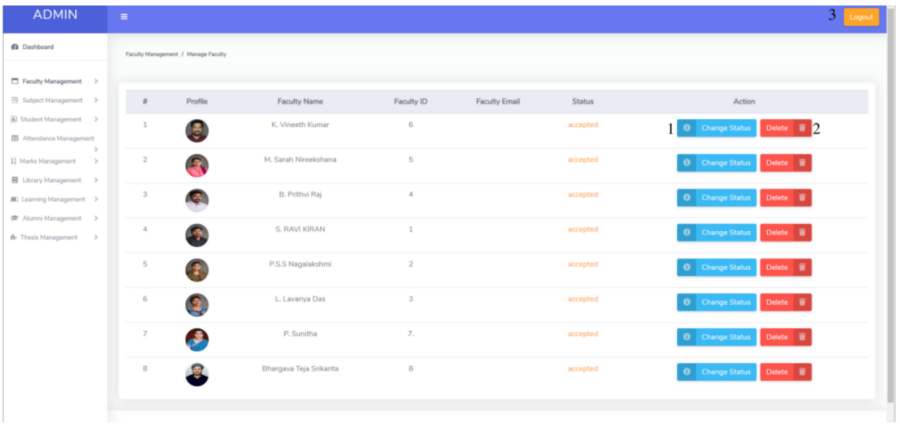
1.We have to select the branch by default DTDP.
2.We have to select the semester in which we wanted to add the marks.
3.We have to select the subject .
4.If we submit the given selection we redirect to the students attendance of that section.
5.Logout

Figure 5.8.2.12 View attendance



1.We have to Search the alumni with batch.
2.Delete the alumni.
3.Logout.

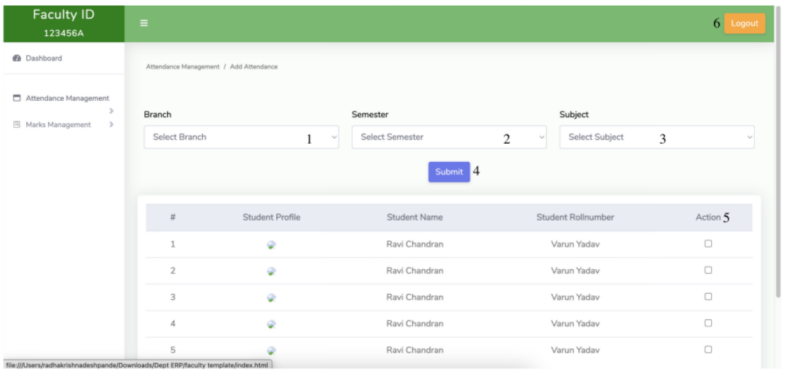
Figure 5.8.2.13 Manage alumni details



1.We have to Change the Status of acceptance/reject.
2.Delete the faculty.
3.Logout.

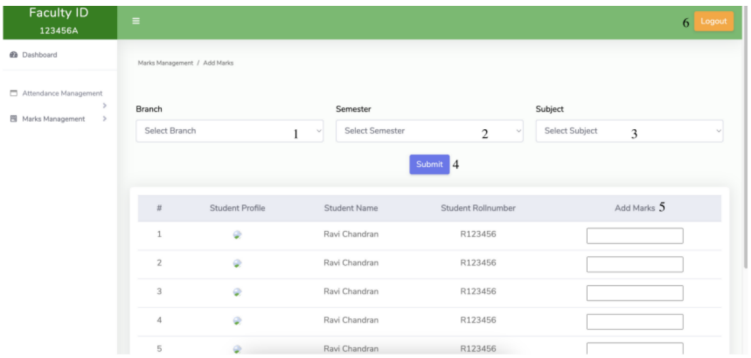
Figure 5.8.2.14 Manage faculty

5.8.3 Faculty dashboard screenshots



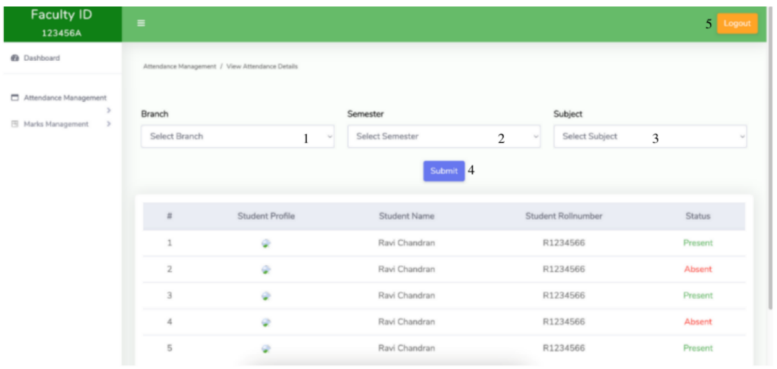
1.Faculty have to Select the Brach by default DTDP.
2.Faculty have to select the semester.
3.Faculty have to select the subject to which we can add.
4.If faculty click on submit then he can view the students who are there in those customized selection.
5.Faculty can action the present if clicked absent if not clicked.
6.Logout

Figure 5.8.3.1 Add attendance



1.Faculty have to Select the Brach by default DTDP.
2.Faculty have to select the semester.
3.Faculty have to select the subject to which we can add.
4.If faculty click on submit then he can view the students who are there in those customized selection.
5.Faculty can add Marks.
6.Logout

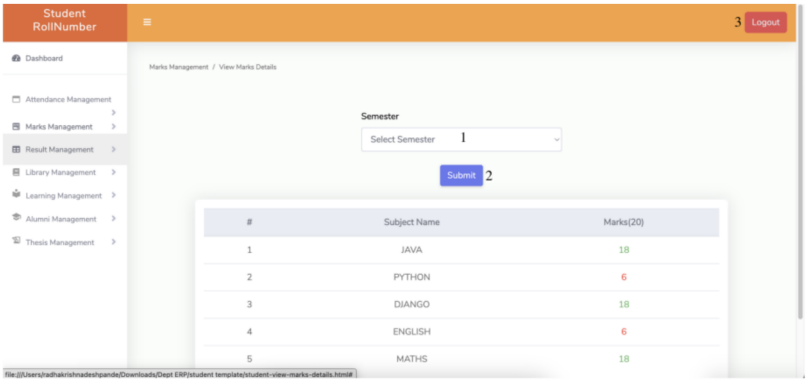
Figure 5.8.3.2 Add marks



1.Faculty have to Select the Brach by default DTDP.
2.Faculty have to select the semester.
3.Faculty have to select the subject to which we can add.
4.If faculty click on submit then he can view the students who are there in those customized selection.
5.Logout.

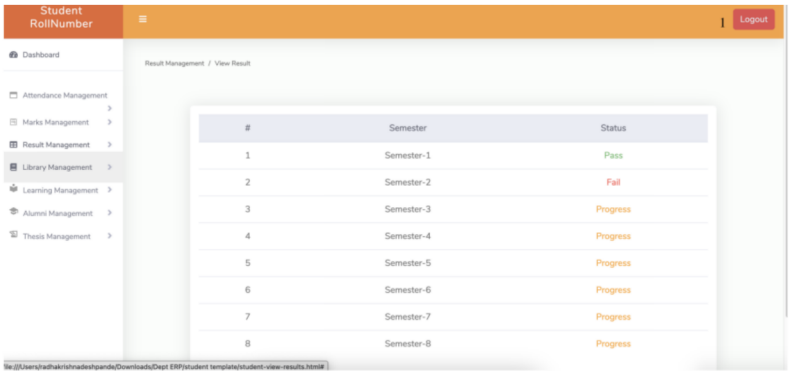
Figure 5.8.3.3 View attendance

5.8.4 Student dashboard screenshots



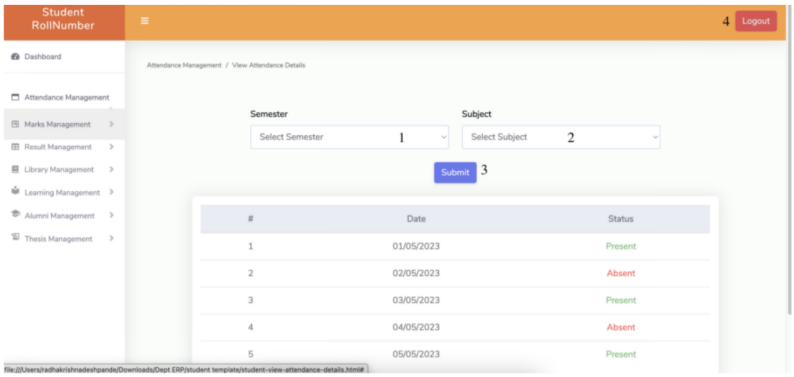
1.Student have to Select the Semester.
2.If student submit the semester he can view the marks he scores.
3.Logout.

Figure 5.8.4.1 View marks



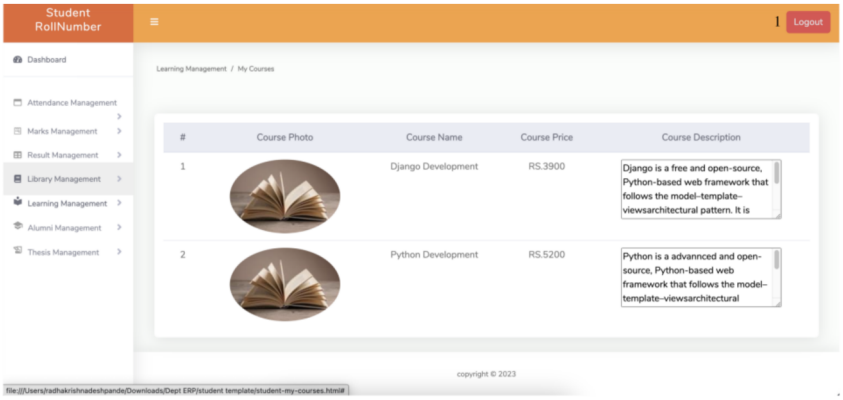
1.Logout

Figure 5.8.4.2 View result



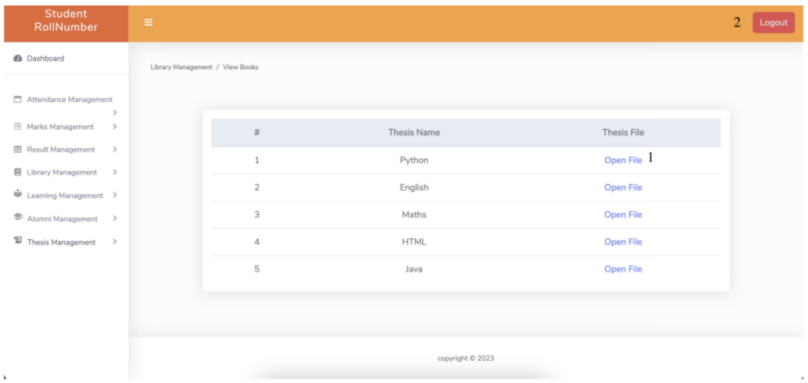
1.Student have to Select the Semester.
2.Student have to submit the Subject which he wanted to see.
3.If student submit the semester he can view the attendance as per the dates.
4.Logout.

Figure 5.8.4.3 View attendance



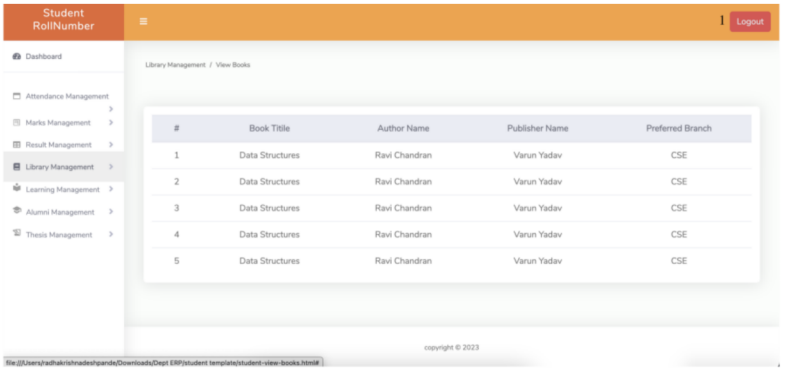
1.Logout.
Student can view Courses here.

Figure 5.8.4.4 View courses



1.Student can open the respective thesis file for requirement .
2.Logout.

Figure 5.8.4.5 View thesis



1.Logout.
Student can view library resources here

Figure 5.8.4.6 View resources

5.8.5 Database screenshots

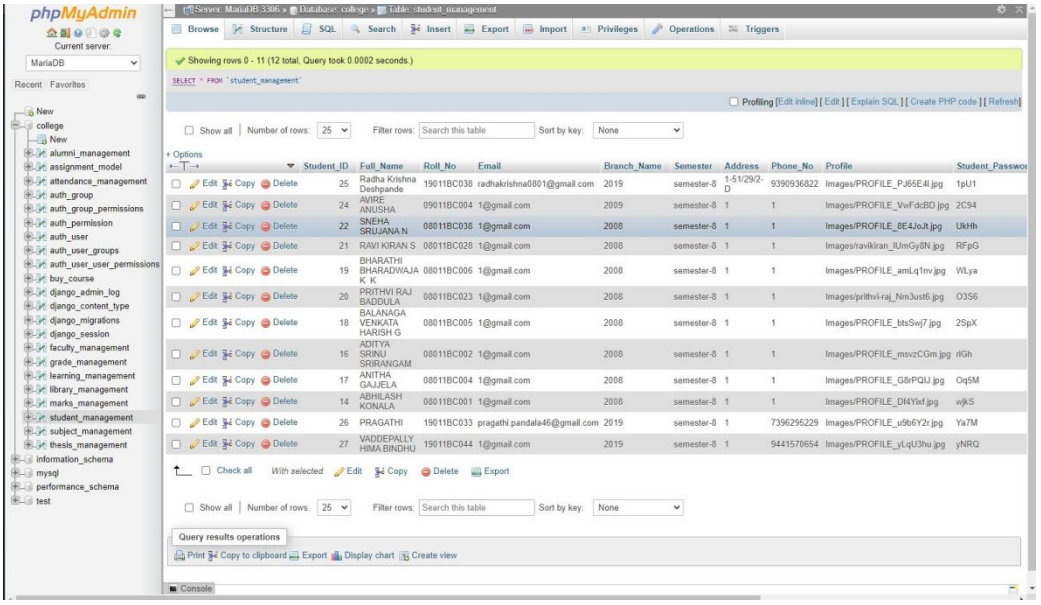


Figure 5.8.5.1 Student records
This is the student details table in the database

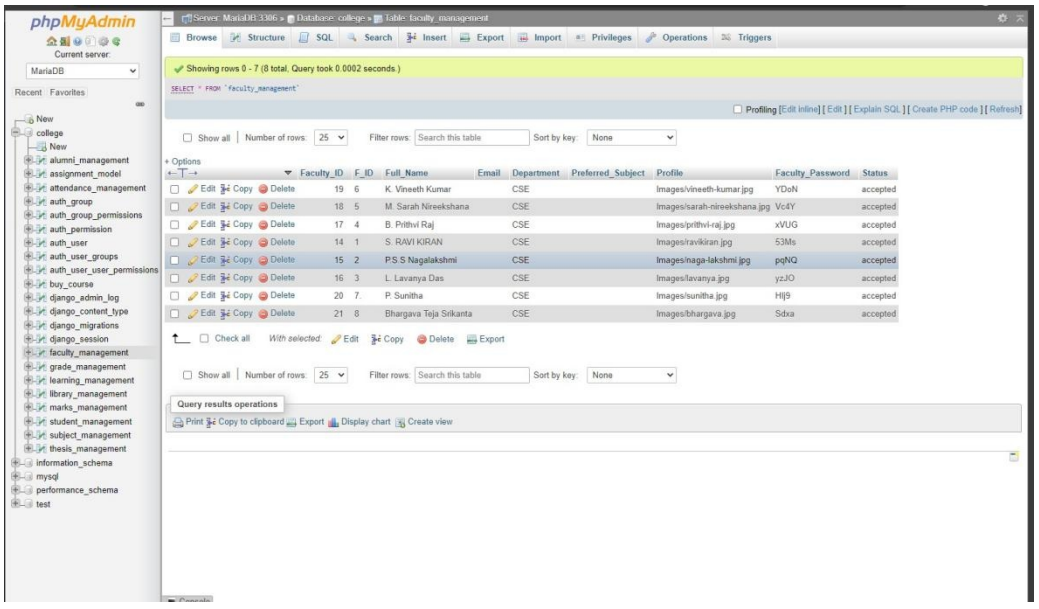


Figure 5.8.5.1 Faculty record
This is this faculty details table in the database.

CHAPTER 6
TESTING

6.1 Unit Testing in Django

Unit testing in your Django project involves testing individual software units, such as login, dashboard, and specific functionalities within each module. You can design test cases to validate the internal program logic and ensure that inputs produce the expected outputs. This includes testing components like faculty management, subject management, student management, attendance management, marks management, library management, learning management system, alumni management, and thesis management.

6.2 Integration Testing in Django

Integration testing in your Django project ensures that the integrated software components work together seamlessly. It involves testing the interaction between different modules, such as faculty management, subject management, student management, attendance management, marks management, library management, learning management system, alumni management, and thesis management. Integration testing helps identify and address any interface defects between these components.

6.3 Functional Testing in Django

Functional testing in your Django project focuses on testing the functionalities and features provided by the different modules. This includes testing the login and dashboard functionalities for administrators, faculty, and students. Additionally, you need to test specific functionalities such as adding and managing faculty, subjects, students, attendance, marks, library resources, courses, alumni details, and thesis details. Functional testing ensures that these functionalities work as expected and meet the specified requirements.

6.4 System Testing in Django

System testing in your Django project verifies that the entire integrated software system, including all the modules, works together smoothly. It involves testing the DTDP Department Management System website as a whole to ensure that all the modules, features, and functionalities are functioning correctly. System testing validates the overall performance, reliability, and usability of the system.

6.5 Acceptance Testing in Django

Acceptance testing in your Django project involves testing the system from the end users' perspective to ensure that it meets their functional requirements and expectations. This includes engaging administrators, faculty, and students to perform testing activities such as logging in, accessing the dashboard, managing various entities, viewing attendance details, marks details, results, library resources, courses, alumni details, and thesis details. Acceptance testing validates that the DTDP Department Management System website satisfies the users' needs and provides a satisfactory user experience.

CHAPTER 7

7.1 Conclusion

In conclusion, the DTDP Department Management system website offers a comprehensive solution for managing various aspects of the department, including faculty, subjects, students, attendance, marks, library resources, learning materials, alumni, and thesis details.

Through the website, the administrator has access to essential functionalities such as login, a user-friendly dashboard, and the ability to manage faculty, subjects, students, attendance, marks, library resources, learning materials, alumni, and thesis details. The system ensures efficient management and organization of departmental operations.

Faculty members can log in to the system, access their personalized dashboard, manage attendance, add and view marks, and stay updated with important departmental information. The system enhances faculty efficiency by automating routine tasks and providing easy access to relevant information.

Students benefit from the website by accessing their login portal, where they can view attendance details, marks, results, library resources, learning materials, alumni information, and thesis details. The system facilitates streamlined communication between students and the department and helps students stay informed about their academic progress.

Overall, the DTDP Department Management system website significantly contributes to enhancing the efficiency, organization, and communication within the department. By incorporating a user-friendly interface, robust functionalities, and effective management of various departmental aspects, the website serves as a valuable tool for administrators, faculty members, and students alike.

7.2 Future Scope

Integration with Student Information System: Integrate the department management system with the university's Student Information System to streamline data synchronization, including student enrollment, course registration, and academic records.

Advanced Reporting and Analytics: Implement advanced reporting and analytics features to generate insightful reports, track trends, and provide data-driven insights on student performance, faculty workload, attendance patterns, and resource utilization. This can help in making informed decisions and identifying areas for improvement.

Online Course Registration: Develop a module for online course registration, allowing students to select and enroll in their preferred courses through the website. This would automate the course registration process and provide students with real-time information on course availability and prerequisites.

Communication and Notifications: Enhance the system to include robust communication features, such as announcements, notifications, and messaging, enabling seamless communication between administrators, faculty members, and students. This can facilitate quick updates, event reminders, and important announcements.

Document Management: Integrate a document management system to store and manage important documents such as syllabi, study materials, research papers, and administrative forms. This would ensure easy access, version control, and efficient collaboration among users.

Integration with Learning Management Systems: Integrate the department management system with popular Learning Management Systems (LMS) to provide a unified platform for managing courses, assignments, online quizzes, and discussions. This integration would enhance the learning experience and promote a seamless transition between administrative tasks and academic activities.

Mobile Application: Develop a mobile application for the DTDP Department Management system, allowing administrators, faculty, and students to access and manage information on the go. This would provide greater flexibility and convenience in interacting with the system.

Feedback and Surveys: Implement a feedback and survey module to collect feedback from students regarding courses, faculty, resources, and overall satisfaction. This feedback can be used to improve the quality of education and address any concerns or issues raised by the students.

Alumni Engagement: Expand the alumni management module to include features such as alumni events, career opportunities, and alumni networking. This would help in fostering a strong alumni network and facilitating collaboration between current students and alumni.

Integration with External Systems: Explore opportunities for integration with external systems and services, such as academic research databases, industry partnerships, or career services platforms, to enhance the resources available to faculty and students.

BIBLIOGRAPHY

Ambler, Scott W. (2002). Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. John Wiley & Sons.

Bass, Len, Clements, Paul, & Kazman, Rick. (2012). Software Architecture in Practice. Addison-Wesley Professional.

Beck, Kent. (2000). Extreme Programming Explained: Embrace Change. Addison-Wesley Professional.

Brooks, Frederick P. (1995). The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley Professional.

Cockburn, Alistair. (2001). Writing Effective Use Cases. Addison-Wesley Professional.

Davis, Alan M. (2013). Just Enough Requirements Management: Where Software Development Meets Marketing. Dorset House.

DeMarco, Tom, & Lister, Timothy. (1999). Peopleware: Productive Projects and Teams. Addison-Wesley Professional.

Fowler, Martin. (2003). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

Fowler, Martin, & Scott, Kendall. (2000). UML Distilled: Applying the Standard Object Modeling Language. Addison-Wesley Professional.

Gamma, Erich, Helm, Richard, Johnson, Ralph, & Vlissides, John. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

Graham, Ian, & Yeh, Susan. (2018). Django for Beginners: Build websites with Python and Django. Leanpub.

Horstmann, Cay S., & Cornell, Gary. (2017). Core Java Volume I -- Fundamentals. Pearson.

Hunt, Andrew, & Thomas, David. (1999). The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley Professional.

Larman, Craig. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. Prentice Hall.

Martin, Robert C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.

McConnell, Steve. (1996). Rapid Development: Taming Wild Software Schedules. Microsoft Press.

McConnell, Steve. (2006). Code Complete: A Practical Handbook of Software Construction. Microsoft Press.

Meyer, Bertrand. (2009). Object-Oriented Software Construction. Prentice Hall.

Pressman, Roger S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.

Schwaber, Ken, & Sutherland, Jeff. (2013). The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. Scrum.org.

Shalloway, Alan, & Trott, James R. (2004). Design Patterns Explained: A New Perspective on Object-Oriented Design. Addison-Wesley Professional.

Sommerville, Ian. (2015). Software Engineering. Pearson.

Vossoughi, Sohrab, & Matthes, Florian. (2016). Web Development with Django Cookbook. Packt Publishing.

W3C (World Wide Web Consortium). (2014). HTML5: A Vocabulary and Associated APIs for HTML and XHTML. W3C Recommendation.

W3C (World Wide Web Consortium). (2018). CSS: The Definitive Guide: Visual Presentation for the Web. O'Reilly Media.


DTDP DMS FILES