University Management System

2025-29

Submitted by:

Muhammad Abubaker	2025-SE-269
Anas Rehman	2025-SE-286

AICT and PF Project Report

Supervised by:

Mam Taliah

Mam Irum

Institute of Data Science

University of Engineering and Technology, Lahore

AICT

Table of Contents

1. Introduction

Project Title: University Management system

Project Type: Web-Based University Management System

Overview:

The University Management System is a web-based academic management application designed to streamline and automate the handling of student information and academic records within a university environment. The primary objective of this system is to reduce manual data handling, minimize errors, and improve the efficiency of academic administration.

2. Problem Statement:

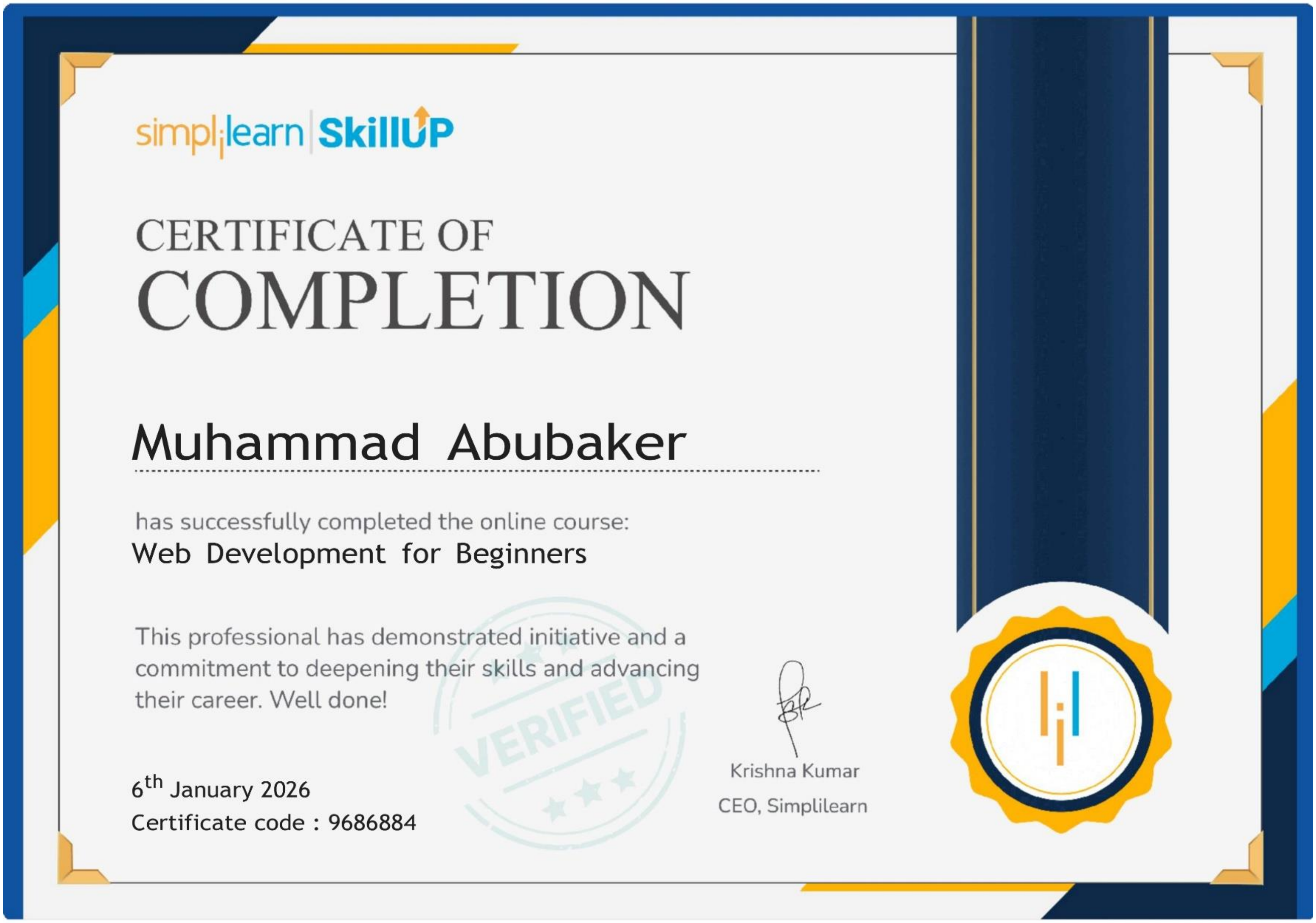
In many educational institutions, the management of student academic records such as personal information, course enrollment, marks, and GPA is often handled manually or through unstructured digital methods. These traditional approaches are time-consuming, prone to human error, and make data retrieval and updates inefficient. Maintaining accurate records becomes increasingly difficult as the number of students.

- GPA calculation done manually is time-consuming and prone to inaccuracies.
- Managing a growing number of student records becomes difficult without proper digital tools.
- Academic staff face challenges in editing and updating student records efficiently.
- There is a need for a secure, organized, and user-friendly web-based academic management system.

3. Learning Perspectives:

This project provided valuable hands-on experience in developing a complete web-based University Management System. It helped in understanding how frontend and backend technologies work together to create a functional application. Through this project, practical knowledge of web development, data handling, and system integration was significantly improved.

- Learned to design responsive and structured user interfaces using HTML and CSS.
- Implemented automatic GPA calculation logic.
- Improved debugging and error-handling skills.
- Gained understanding of project structuring and modular development.
- Enhanced problem-solving and analytical thinking abilities.
- Learned to design responsive and structured user interfaces using HTML and CSS.
- Learned how to store, retrieve, and update data efficiently on the server side.
- Understood how to use Python Flask for backend development.
- Gained confidence in building a complete mini project



simplilearn SkillUP

CERTIFICATE OF
COMPLETION

Muhammad Abubaker

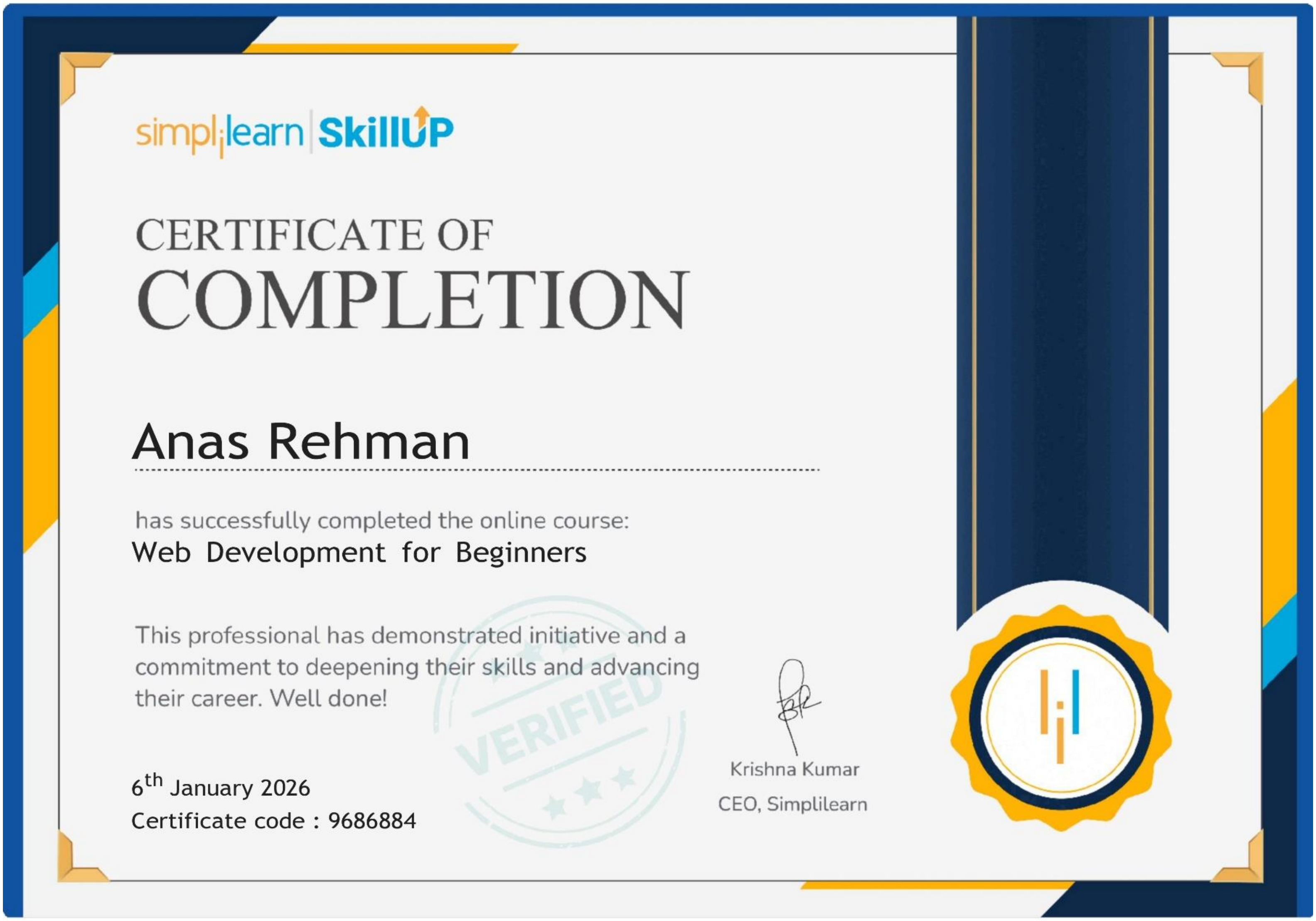
has successfully completed the online course:
Web Development for Beginners

This professional has demonstrated initiative and a commitment to deepening their skills and advancing their career. Well done!

VERIFIED

6th January 2026
Certificate code : 9686884

Krishna Kumar
CEO, Simplilearn



simplilearn SkillUP

CERTIFICATE OF COMPLETION

Anas Rehman

has successfully completed the online course:

Web Development for Beginners

This professional has demonstrated initiative and a commitment to deepening their skills and advancing their career. Well done!

VERIFIED

6th January 2026

Certificate code : 9686884

Krishna Kumar

CEO, Simplilearn

Conclusion:

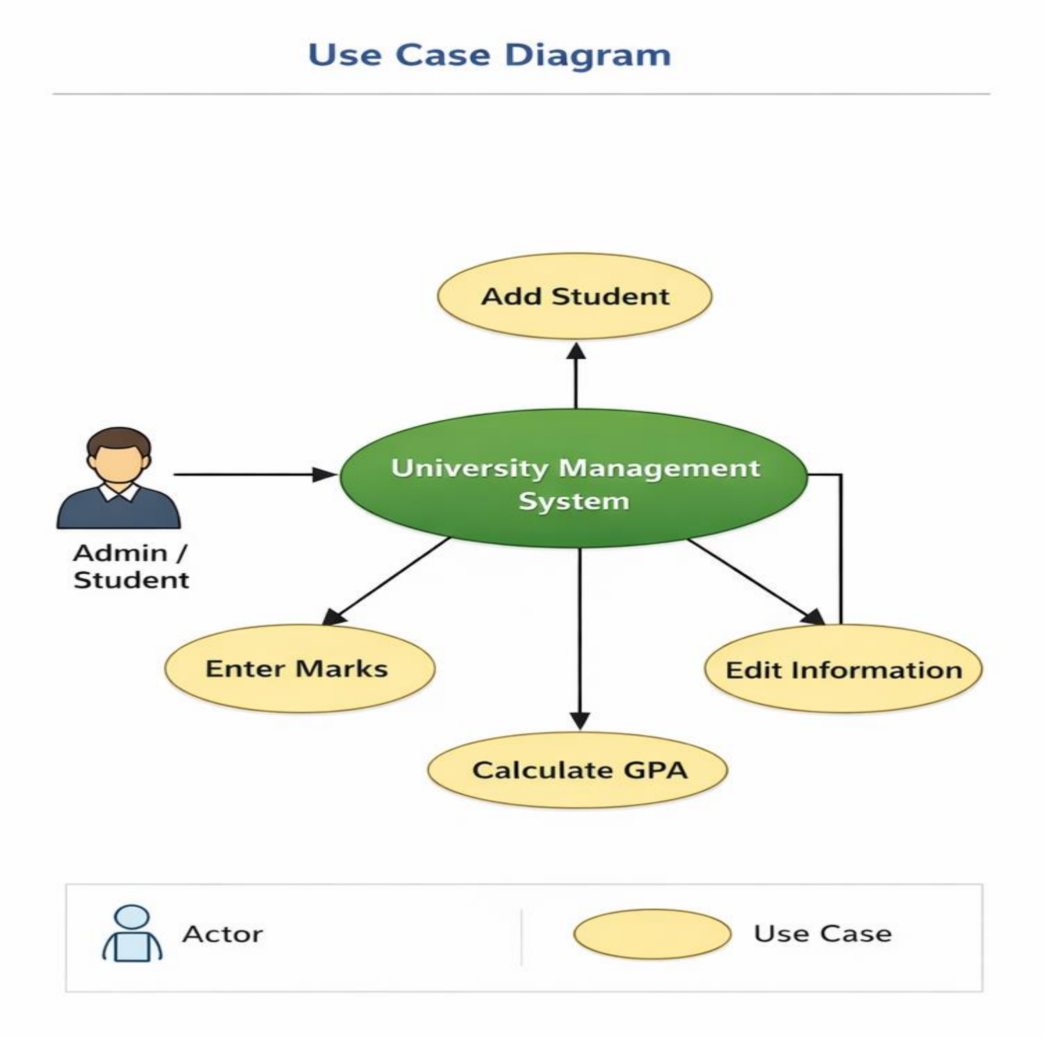
The University Management System is a simple and efficient web-based application developed to manage student information, academic records, and marks in an organized way. This system reduces manual work and minimizes errors by automating student record management and GPA calculation. Through this project, important concepts of web development were implemented using HTML, CSS, JavaScript, and Python Flask.

Overall, the project successfully meets its objectives and provides a strong foundation for future improvements such as database integration, user authentication, and enhanced reporting features.

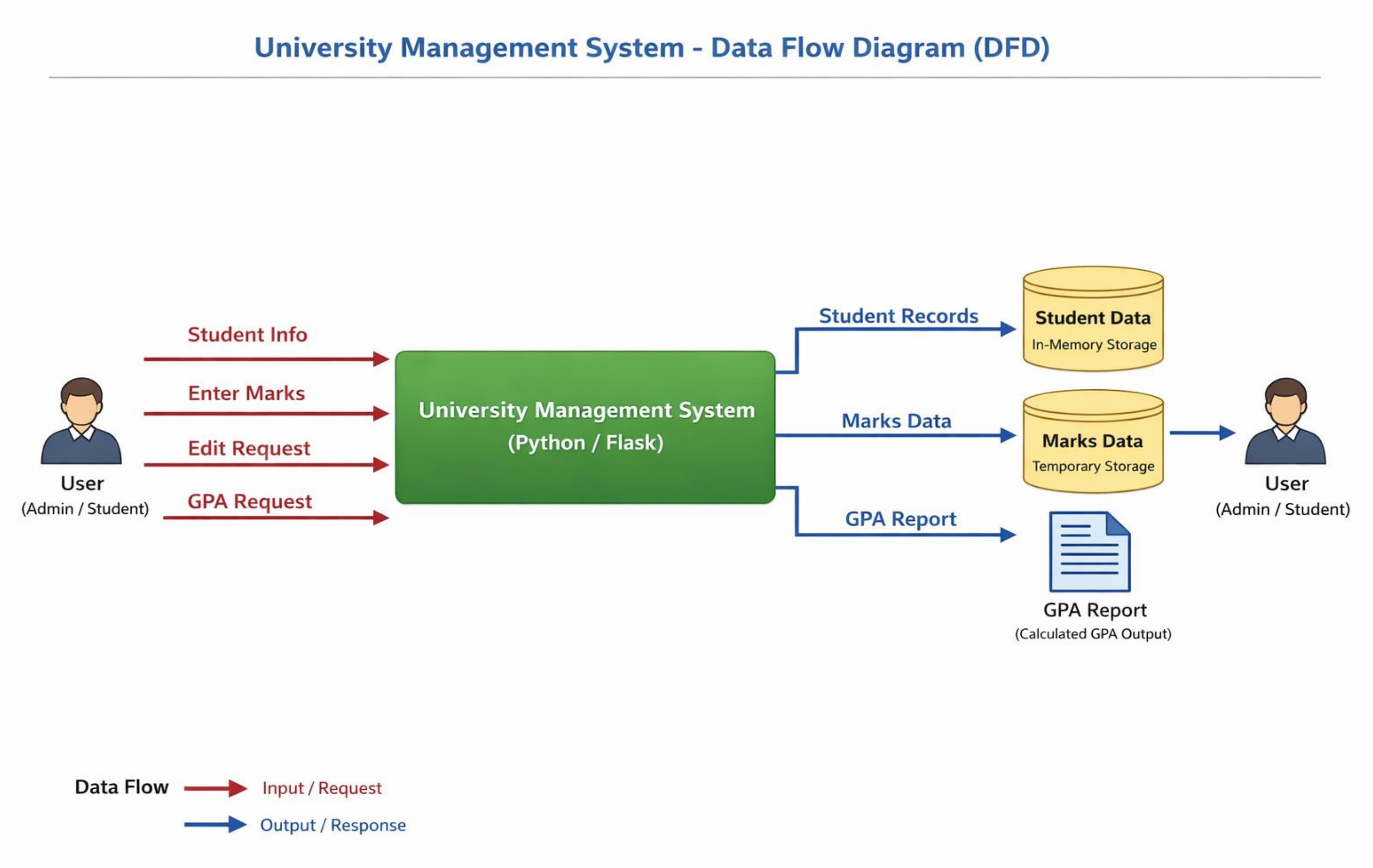
4. Key Technologies:

- **HTML** – Used to create the structure of web pages.
- **CSS** – Used for styling and designing the user interface.
- **JavaScript** – Used to handle user interactions and communicate with the backend.
- **Python** – Used as the main programming language for backend development.
- **Flask Framework** – Used to build the server-side application and manage routes.
- **Web Browser** – Used to run and test the application.
- **Visual Studio Code** – Used as the development environment for coding

Use case Diagram:



Flow Diagram:



- **Start System**
The user opens the University Management System through a web browser.
- **Home Page Display**
The system displays the home page with navigation options.
- **Add Student Information**
The admin enters student details such as name, roll number, department, course, section, and semester.
The system saves the student data in the backend.
- **View Student Records**
The system retrieves and displays all stored student records in a table format.

- **Access Student by Roll Number**

The admin enters a student’s roll number.

The system fetches and displays the student’s information.

- **Enter Student Marks**

Subject-wise marks are entered for the selected student.

The system stores the marks in the backend.

- **Automatic GPA Calculation**

The system calculates GPA automatically based on entered marks.

Grade is assigned according to GPA.

- **Edit Student Information**

The admin edits student details or marks if required.

Updated data is saved and reflected in the records.

- **Generate GPA Report**

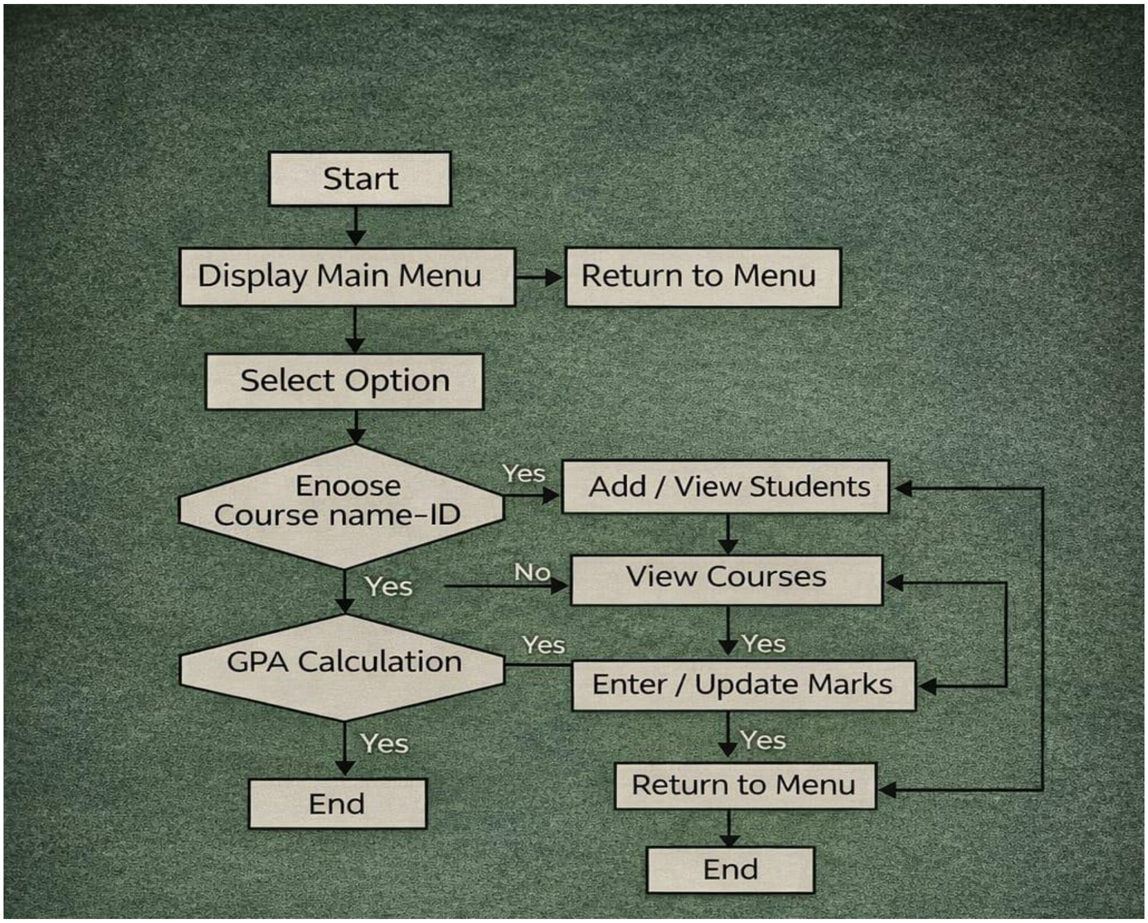
The system displays GPA and grade reports for students.

- **View Course Details**

The user views course-related information.

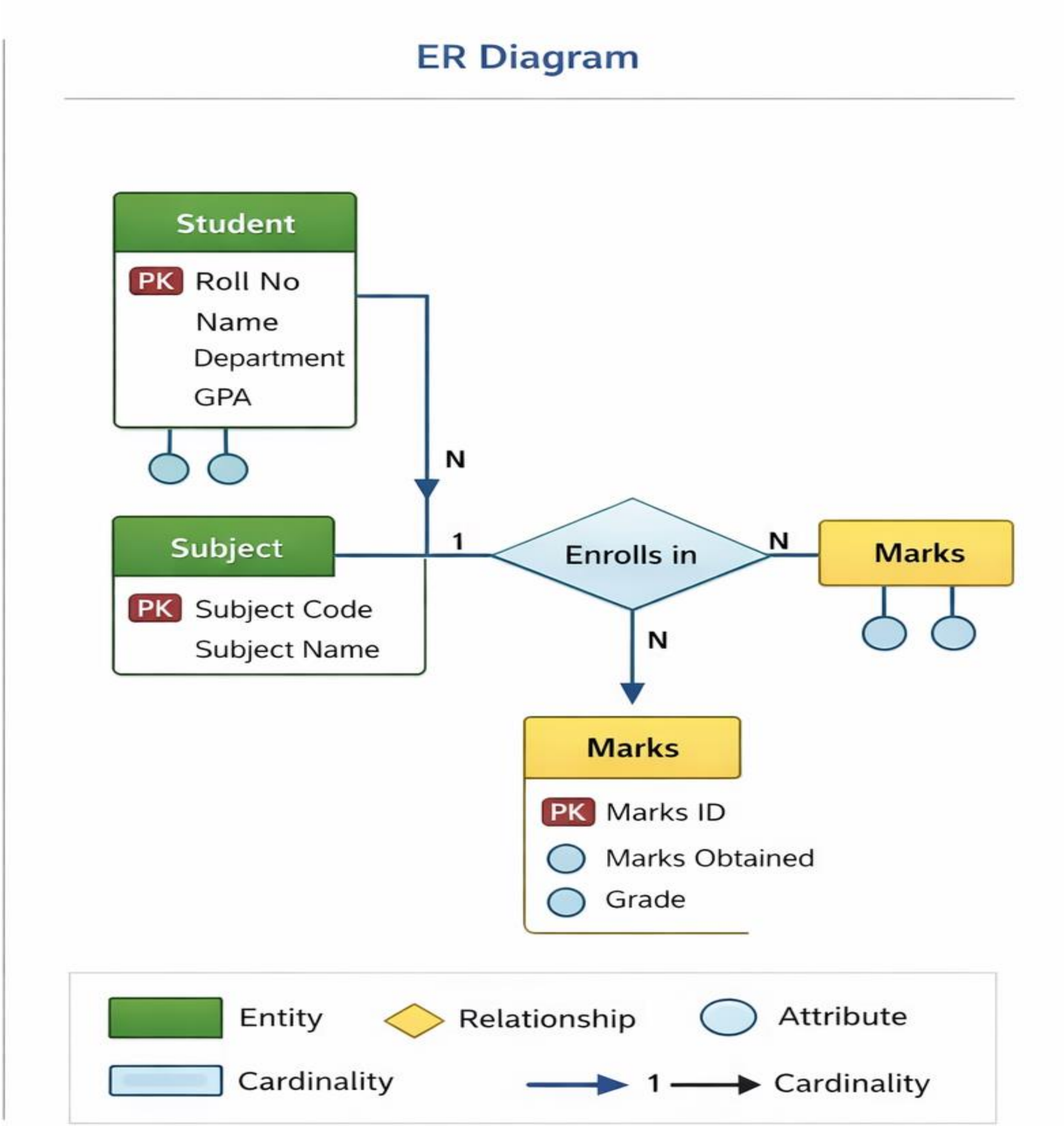
- **End Process**

The system completes the requested operation and remains ready for further actions.



Er Diagram:

The ER (Entity–Relationship) Diagram is used to visually represent the structure of data and the relationships between different entities in a system. It helps in understanding how data is organized and connected before actual implementation.



Functional Requirements:

- The system shall allow users to add new student records by entering personal and academic details.
- The system shall store student information temporarily in backend memory without using a database.
- The system shall display all stored student records in a structured tabular format.
- The system shall allow users to search for a student using a unique roll number.
- The system shall enable users to enter subject-wise marks for a selected student.
- The system shall associate marks with the correct student record using roll numbers.
- The system shall allow users to edit existing student information and academic marks.
- The system shall update and reflect modified records immediately without page reload.
- The system shall calculate GPA automatically based on predefined grading criteria.
- The system shall display GPA reports along with student details in a readable format.
- The system shall provide a menu or page-based navigation for accessing different modules.

Non Functional Requirements:

- The system shall provide a simple and user-friendly interface for easy interaction.
- The system shall respond to user actions within a reasonable time.
- The system shall be lightweight and require minimal system resources.
- The system shall run on any standard web browser without special software installation.
- The backend shall handle multiple requests reliably during execution.
- The system shall maintain data consistency during runtime.
- The system shall be easy to understand and maintain for future improvements.
- The system shall be developed using modular and readable code.
- The system shall provide basic input validation to reduce user errors.

- The system shall follow standard coding practices for better reliability.
- The system shall follow standard coding practices for better reliability.
- The system shall ensure smooth frontend-backend communication.
- The system shall be suitable for academic and learning purposes.

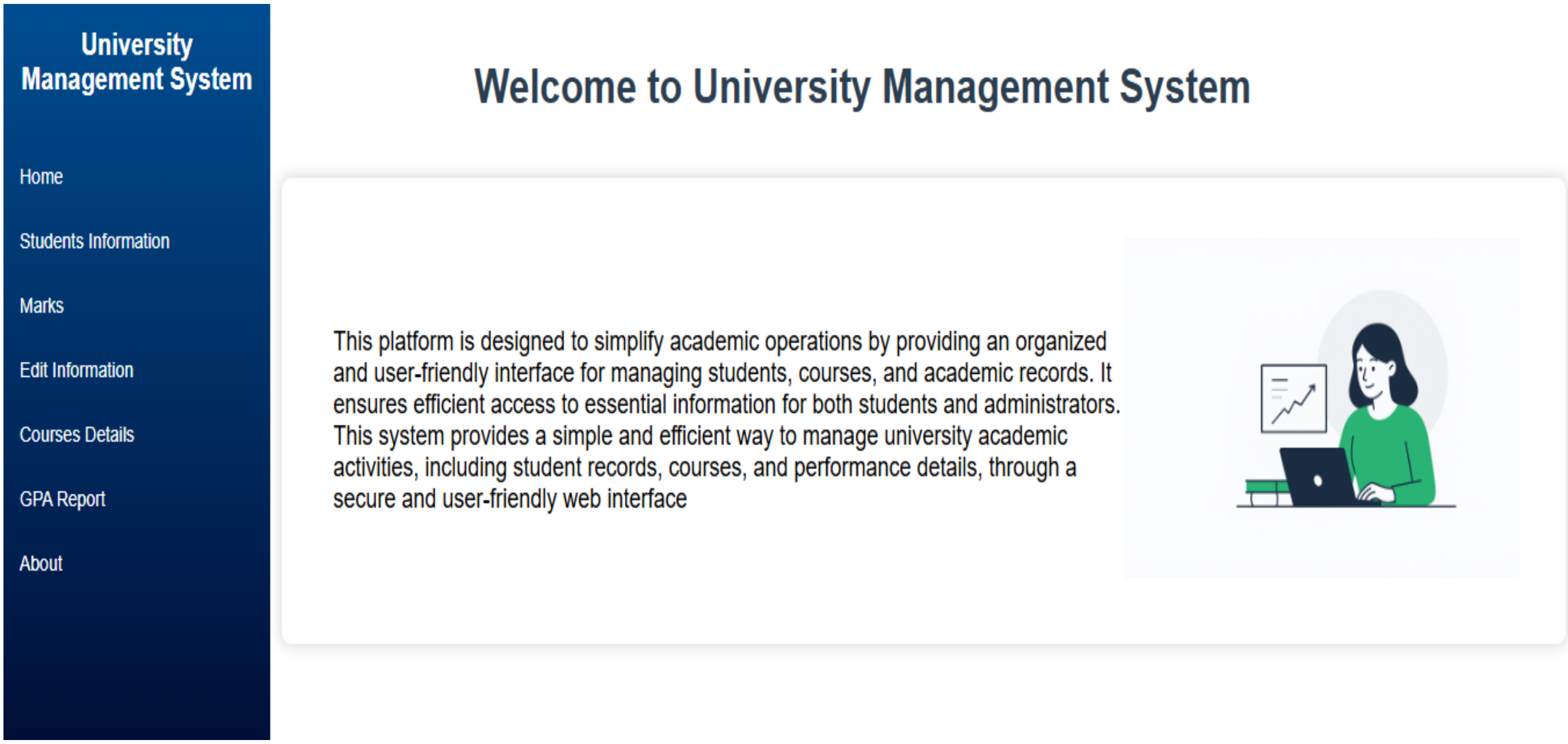
5. Functionality behind Each Major Part:

5.1 Home Page

- Home page Acts as the landing page of the University Management System
- It Welcomes users with a system introduction message
- it explains the purpose of the system
- It provides a central navigation sidebar
- The Sidebar links to all major modules:

Students Information
Marks Management
Edit Information
GPA Report
Courses & About pages

- Maintains a consistent layout across the website
- Improves user experience and usability
- Contains informational content only (no data processing)
- Helps users understand what the system does before using features
- Confirms the backend routing is working correctly

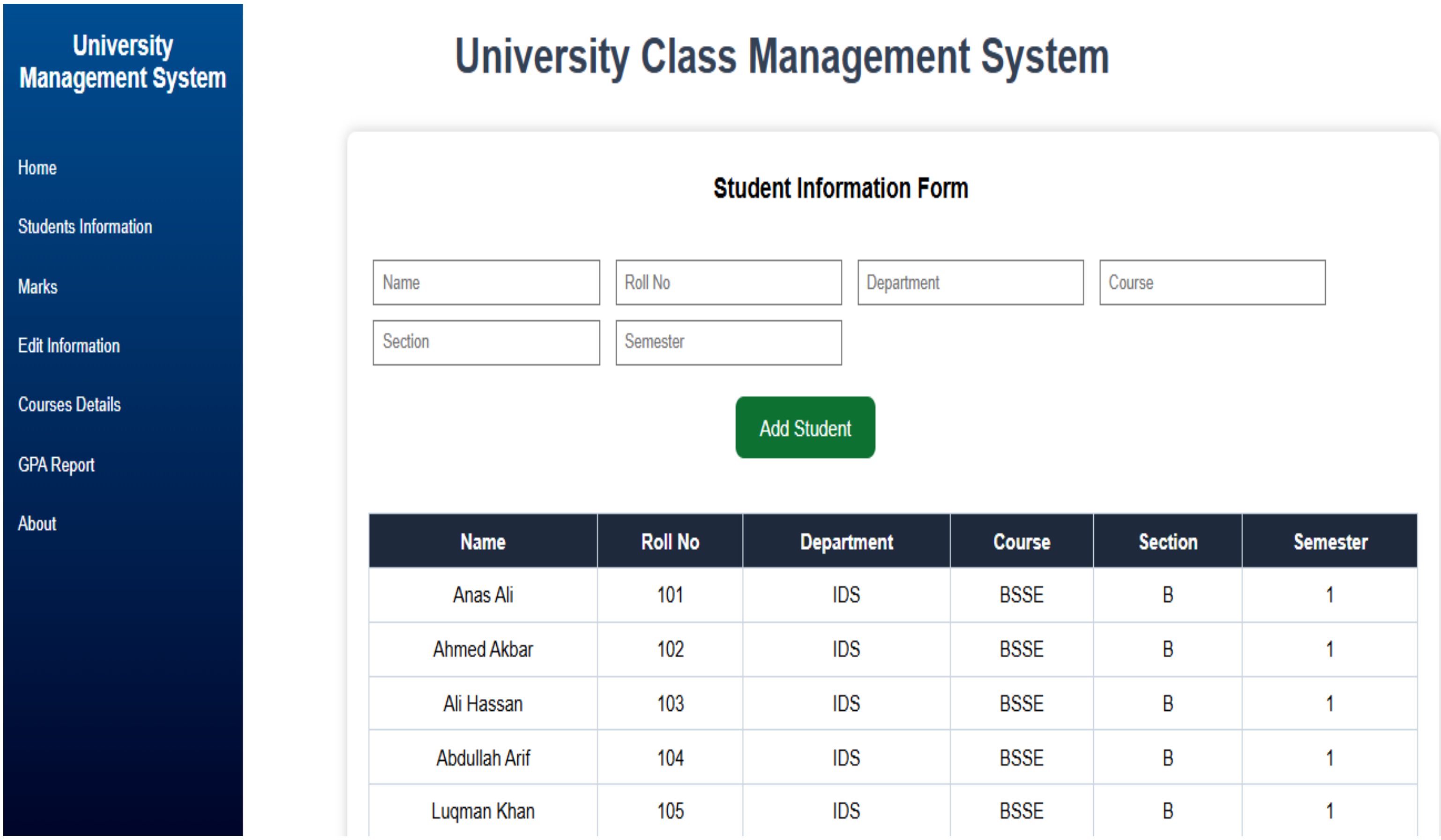


Welcome to University Management System

This platform is designed to simplify academic operations by providing an organized and user-friendly interface for managing students, courses, and academic records. It ensures efficient access to essential information for both students and administrators. This system provides a simple and efficient way to manage university academic activities, including student records, courses, and performance details, through a secure and user-friendly web interface

5.2 Student Page

- The Students Information page is used to add and manage student records in the system.
- It provides a form where users can enter student details such as name, roll number, department, course, email, and semester.
- The page sends student data to the backend using JavaScript and the Fetch API without reloading the page.
- Student records are stored temporarily in backend memory instead of using a database.
- The page retrieves all stored student records from the backend when it loads.
- Student data is displayed dynamically in a table below the form.
- The table updates automatically whenever a new student is added.
- The form fields are cleared after successful submission to improve user experience.
- The page ensures smooth interaction between frontend and backend components.
- A sidebar navigation menu is included to allow easy access to other system modules.
- The Students Information page acts as the main module for managing student data in the University Management System.



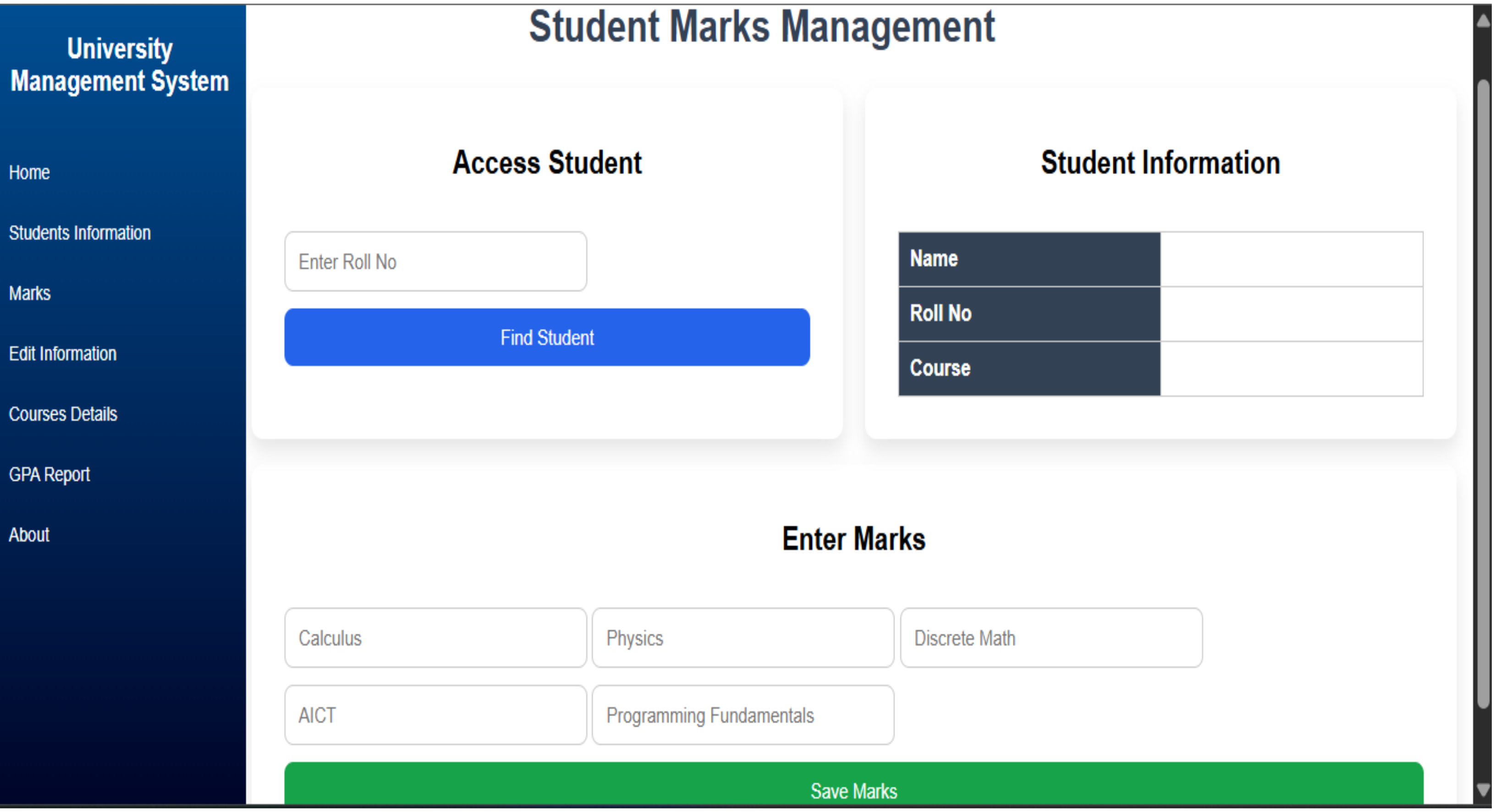
University Class Management System

Student Information Form

Name	Roll No	Department	Course	Section	Semester
Anas Ali	101	IDS	BSSE	B	1
Ahmed Akbar	102	IDS	BSSE	B	1
Ali Hassan	103	IDS	BSSE	B	1
Abdullah Arif	104	IDS	BSSE	B	1
Luqman Khan	105	IDS	BSSE	B	1

5.3 Marks Page

- The Marks page is used to access and manage academic marks of students.
- It allows the user to search for a student by entering the roll number.
- The page retrieves the student’s basic information from the backend based on the entered roll number.
- Student details such as name, roll number, and course are displayed after successful search.
- The page provides input fields to enter marks for different subjects.
- Subject marks are submitted to the backend using JavaScript and the Fetch API.
- Marks data is stored in backend memory without using a database.
- The system ensures that marks are linked to the correct student using the roll number.
- The page prevents page reload while saving marks for a smooth user experience.
- The Marks page demonstrates interaction between multiple frontend sections and backend logic.
- A consistent sidebar navigation is provided for easy movement across system pages.
- The Marks page serves as the core module for academic performance management in the system.



Student Marks Management

Access Student

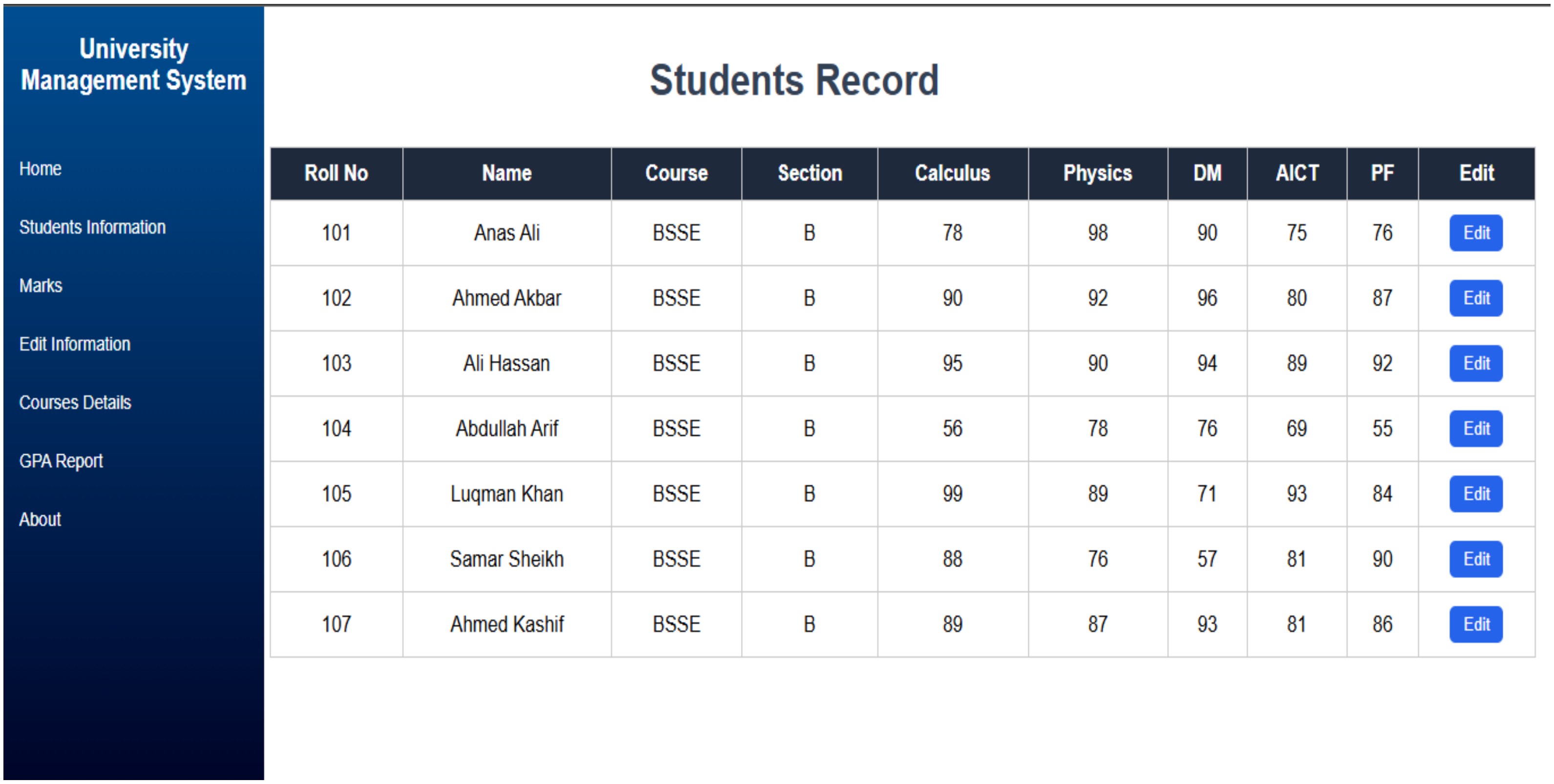
Student Information

Name	
Roll No	
Course	

Enter Marks

5.4 Edit Page

- The Edit Information page is used to view and update existing student records.
- It displays student details and subject marks in a structured table format.
- The page allows users to edit student information using an interactive edit option.
- When the edit button is clicked, the selected student's data is loaded into an editable form.
- Users can modify personal details as well as subject marks of a student.
- Updated information is saved and reflected immediately in the displayed table.
- The page ensures data consistency by editing records linked to student roll numbers.
- JavaScript is used to control the edit functionality and update the displayed data dynamically.
- The page provides a user-friendly interface for correcting or updating student records.
- A consistent sidebar navigation menu is included for easy access to other system modules.
- The Edit Information page demonstrates dynamic data manipulation without page reloads.
- This page plays an important role in maintaining accurate student and academic records.

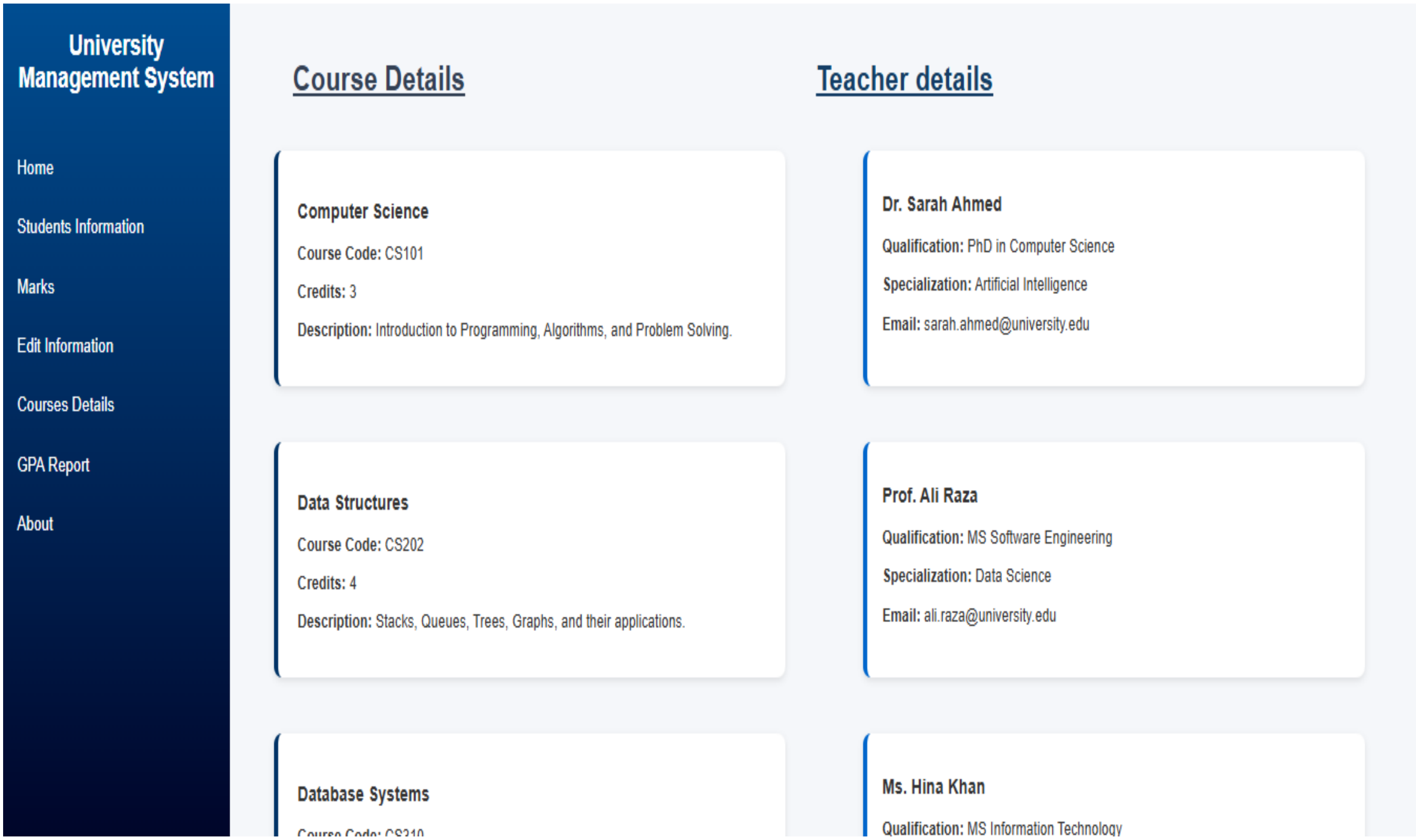


Students Record

Roll No	Name	Course	Section	Calculus	Physics	DM	AICT	PF	Edit
101	Anas Ali	BSSE	B	78	98	90	75	76	Edit
102	Ahmed Akbar	BSSE	B	90	92	96	80	87	Edit
103	Ali Hassan	BSSE	B	95	90	94	89	92	Edit
104	Abdullah Arif	BSSE	B	56	78	76	69	55	Edit
105	Luqman Khan	BSSE	B	99	89	71	93	84	Edit
106	Samar Sheikh	BSSE	B	88	76	57	81	90	Edit
107	Ahmed Kashif	BSSE	B	89	87	93	81	86	Edit

5.5 Course Page

- The Courses Details page is used to display information about academic courses offered by the university.
- It provides details such as course name, course code, and related academic information.
- The page helps users understand the structure and offerings of different programs.
- Course information is presented in an organized and readable format.
- The page does not modify student data and is mainly informational in nature.
- It supports students and administrators by providing clear course-related details.
- A consistent sidebar navigation menu is used to maintain uniformity across the system.
- The Courses page improves overall system usability by centralizing course information.
- It acts as a reference module within the University Management System.
- The page enhances the completeness and professionalism of the system interface.



Course Details

Computer Science

Course Code: CS101

Credits: 3

Description: Introduction to Programming, Algorithms, and Problem Solving.

Data Structures

Course Code: CS202

Credits: 4

Description: Stacks, Queues, Trees, Graphs, and their applications.

Database Systems

Course Code: CS310

Teacher details

Dr. Sarah Ahmed

Qualification: PhD in Computer Science

Specialization: Artificial Intelligence

Email: sarah.ahmed@university.edu

Prof. Ali Raza

Qualification: MS Software Engineering

Specialization: Data Science

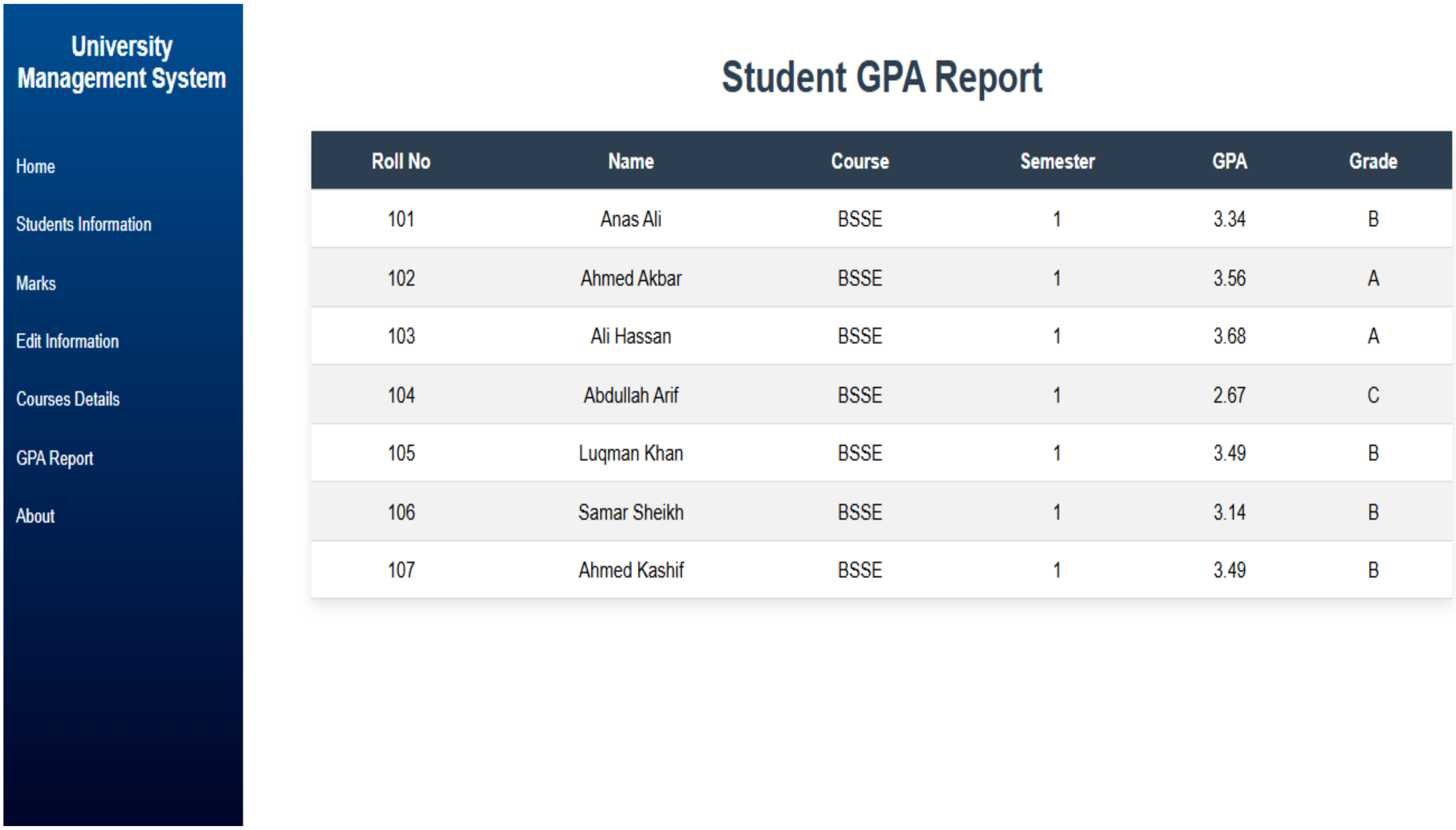
Email: ali.raza@university.edu

Ms. Hina Khan

Qualification: MS Information Technology

5.6 GPA Report Page

- The GPA Report page is used to display the calculated Grade Point Average of students.
- It retrieves student marks from the backend and computes GPA based on predefined criteria.
- The page shows student details such as roll number, name, course, and semester along with GPA.
- Letter grades are assigned according to the calculated GPA values.
- The GPA data is displayed in a tabular format for clarity and readability.
- The page updates GPA information dynamically based on stored student marks.
- It allows academic performance evaluation at a glance.
- The page helps students and administrators monitor academic progress effectively.
- A consistent sidebar navigation menu is provided for easy access to other system modules.
- The GPA Report page demonstrates backend calculation and frontend data presentation.
- This page serves as the final academic performance summary in the system.

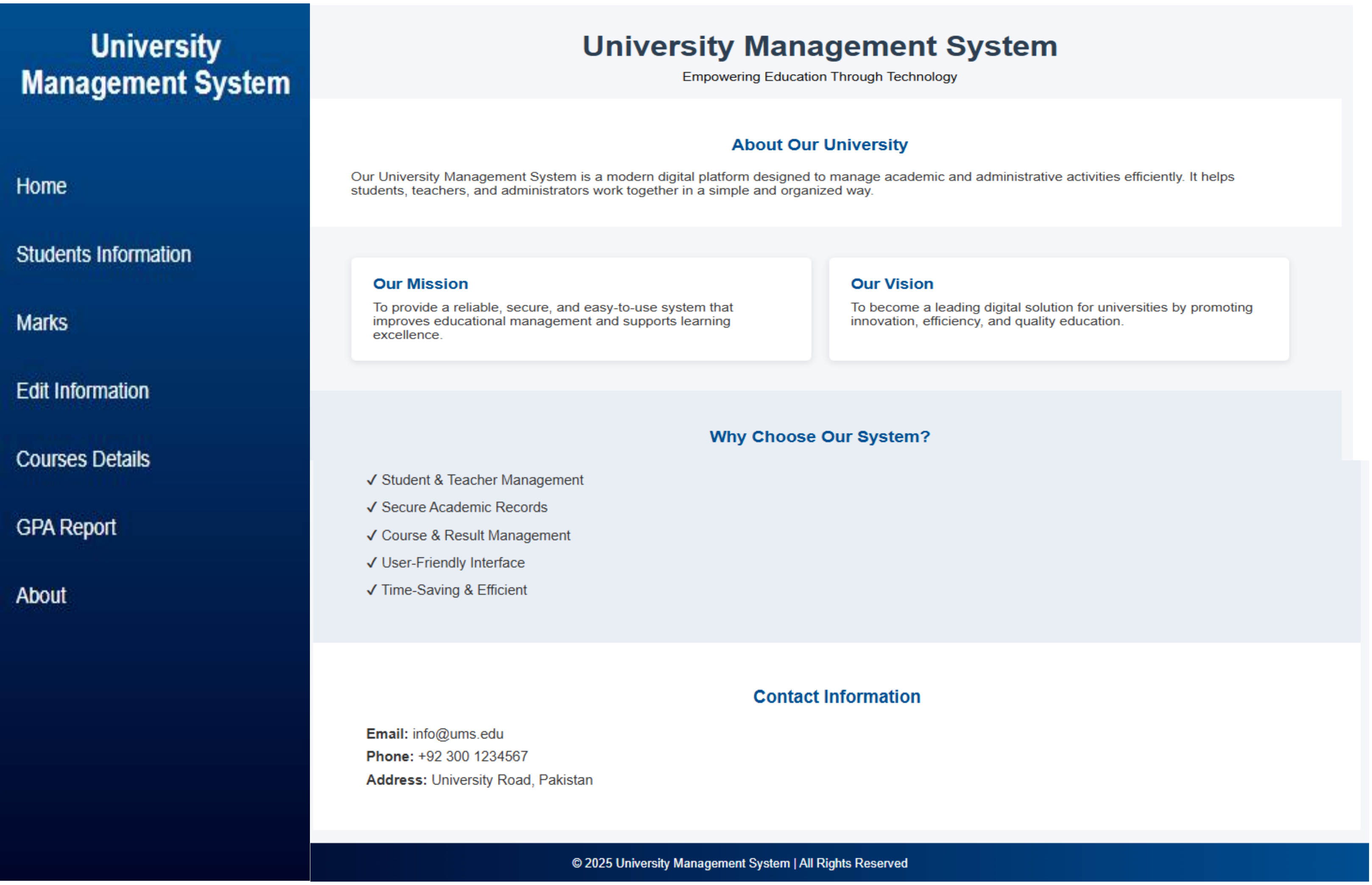


Student GPA Report

Roll No	Name	Course	Semester	GPA	Grade
101	Anas Ali	BSSE	1	3.34	B
102	Ahmed Akbar	BSSE	1	3.56	A
103	Ali Hassan	BSSE	1	3.68	A
104	Abdullah Arif	BSSE	1	2.67	C
105	Luqman Khan	BSSE	1	3.49	B
106	Samar Sheikh	BSSE	1	3.14	B
107	Ahmed Kashif	BSSE	1	3.49	B

5.7 About Page

- The About page provides an overview of the University Management System and its purpose.
- It explains the objectives and scope of the system in a clear manner.
- The page describes how the system helps manage students, marks, and academic records.
- It highlights the technologies used to develop the system, such as HTML, CSS, JavaScript, and Python.
- The page helps users understand the motivation behind creating the system.
- It serves as an informational section rather than a functional module.
- The About page improves transparency and clarity for users and evaluators.
- A consistent sidebar navigation menu is maintained for uniformity across all pages.
- The page enhances the professional presentation of the project.
- The About page acts as a conclusion and explanation of the system’s design and functionality.



University Management System

Empowering Education Through Technology

About Our University

Our University Management System is a modern digital platform designed to manage academic and administrative activities efficiently. It helps students, teachers, and administrators work together in a simple and organized way.

Our Mission

To provide a reliable, secure, and easy-to-use system that improves educational management and supports learning excellence.

Our Vision

To become a leading digital solution for universities by promoting innovation, efficiency, and quality education.

Why Choose Our System?

✓ Student & Teacher Management
✓ Secure Academic Records
✓ Course & Result Management
✓ User-Friendly Interface
✓ Time-Saving & Efficient

Contact Information

Email: info@ums.edu
Phone: +92 300 1234567
Address: University Road, Pakistan

© 2025 University Management System | All Rights Reserved

Python as backend:

Python is used as the backend of the University Management System to handle all server-side operations and application logic. The backend is developed using the Flask framework, which allows the system to receive and respond to requests from the frontend pages. Python processes student information and marks sent from the frontend in JSON format and stores them temporarily in server memory without using a database. It manages the flow of data between different modules such as student management, marks entry, record editing, and GPA calculation. The backend also performs important calculations, including computing GPA based on students' marks, and sends the results back to the frontend for display. By separating backend logic from the user interface, Python ensures smooth communication, dynamic updates without page reloads, and a well-structured and maintainable system design.

Objectives:

- To develop a menu-driven console-based application using Python.
- To understand and apply core Python programming concepts.
- To enable users to enter, view, update, and delete records through the console.
- To manage data efficiently using Python data structures.
- To improve logical thinking and problem-solving skills.
- To demonstrate basic CRUD operations in a text-based environment.
- To provide a simple and user-friendly console interaction.

Project Scope:

- The project focuses on developing a console-based application using Python.
- The system operates entirely through a text-based user interface.
- It allows users to enter, view, and manage data using menu-driven options.
- The application stores data temporarily in Python data structures such as lists or dictionaries.
- The project supports basic CRUD operations (Create, Read, Update, Delete).
- User interaction is handled through keyboard input and console output.
- The system demonstrates core Python programming concepts.
- The application is designed for learning and academic purposes.
- Advanced features such as databases, networking, and authentication are outside the project scope.

4. Key Technologies

4.1 Core Technologies:

- **Python** is used as the main programming language for backend logic and data processing.
- **Flask Framework** is used to build the web backend and handle routing and APIs.
- **Python Data Structures** are used to store data temporarily without a database.

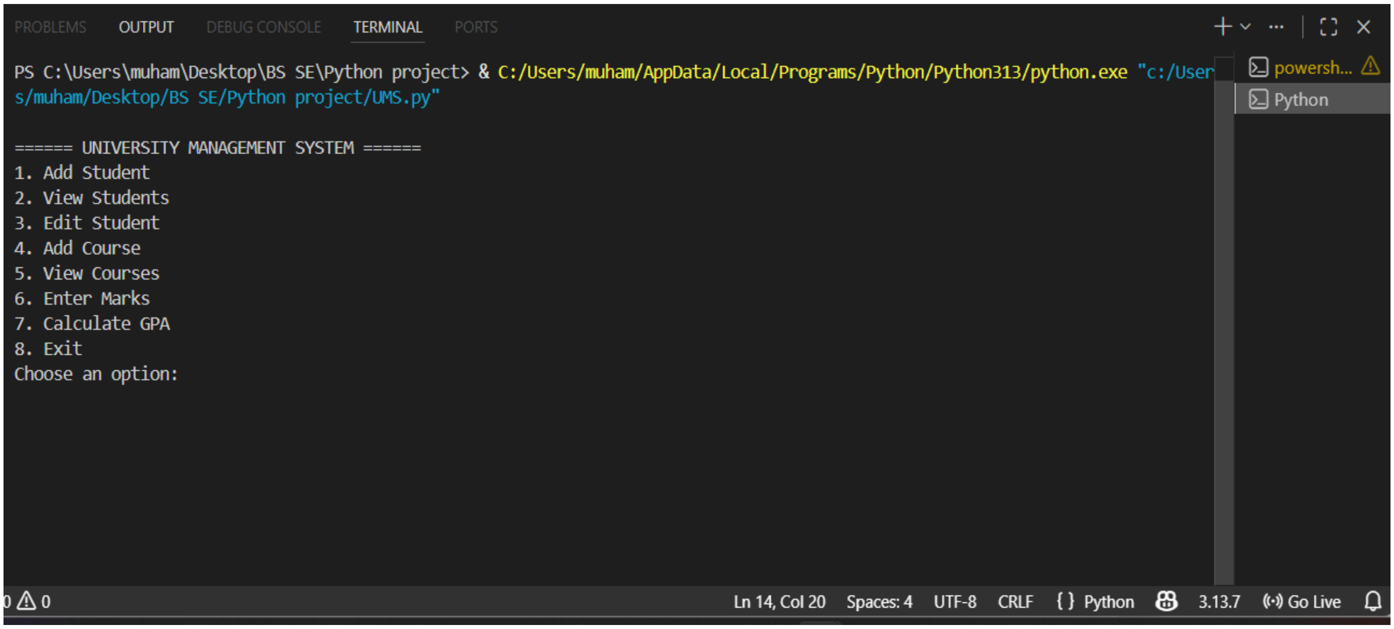
4.2 Development Tools

- IDE: VS Code
- Testing: Manual console testing
- Documentation: Inline comments and this report

5. Module-Wise Implementation Details:

5.1 Main Module:

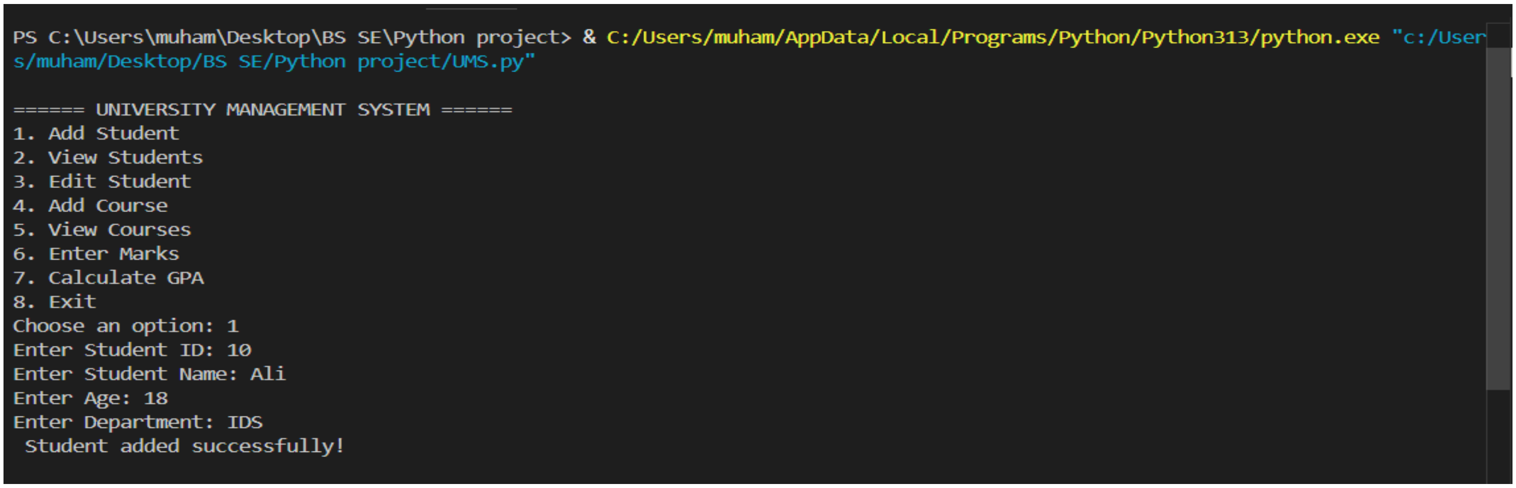
- The main module acts as the entry point of the Python application.
- It displays a menu-driven interface to the user through the console.
- It controls program flow by calling different functional modules based on user input.



```
PS C:\Users\muham\Desktop\BS SE\Python project> & C:/Users/muham/AppData/Local/Programs/Python/Python313/python.exe "c:/User
s/muham/Desktop/BS SE/Python project/UMS.py"

====== UNIVERSITY MANAGEMENT SYSTEM ======
1. Add Student
2. View Students
3. Edit Student
4. Add Course
5. View Courses
6. Enter Marks
7. Calculate GPA
8. Exit
Choose an option:
```

5.2 Add Student Module:

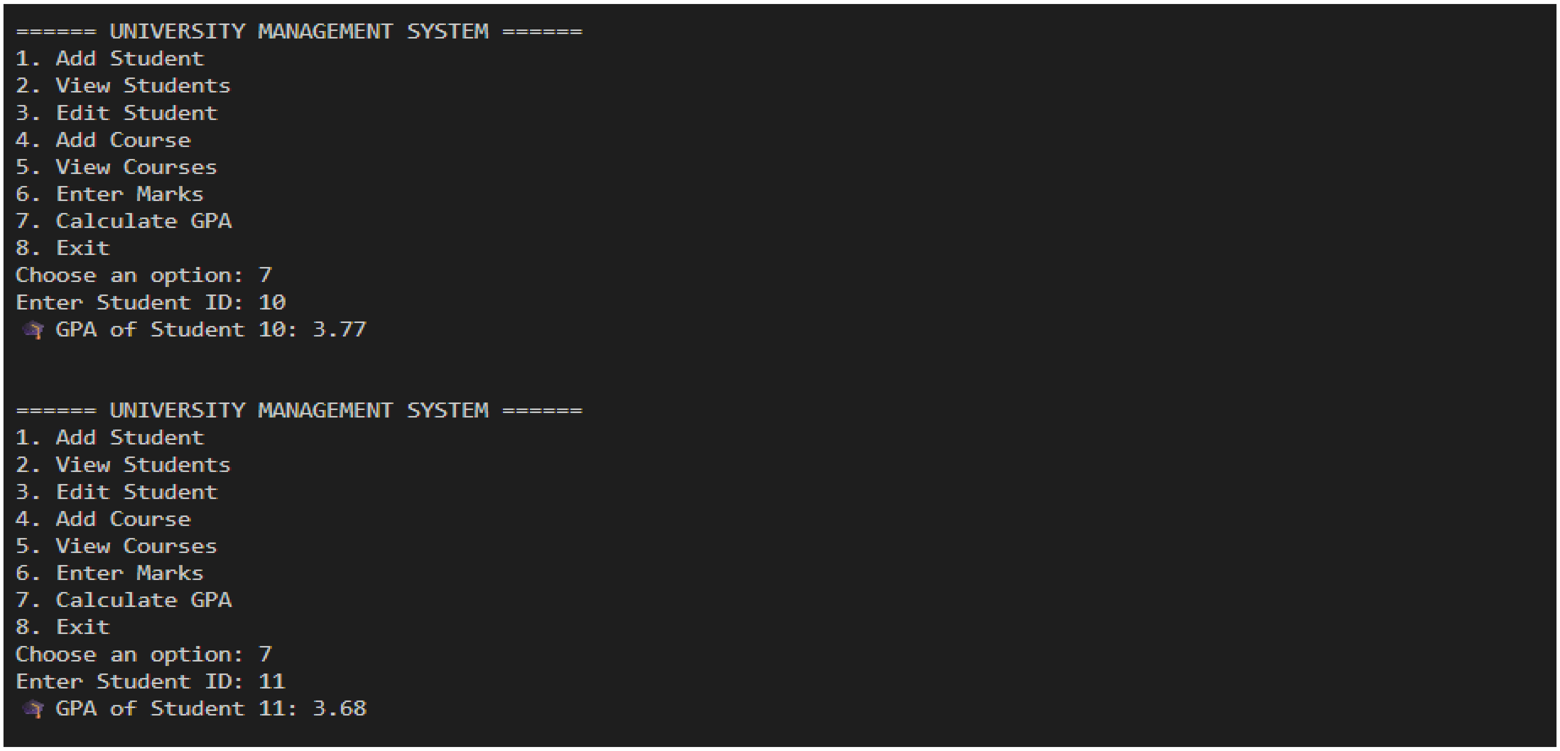


```
PS C:\Users\muham\Desktop\BS SE\Python project> & C:/Users/muham/AppData/Local/Programs/Python/Python313/python.exe "c:/User
s/muham/Desktop/BS SE/Python project/UMS.py"

====== UNIVERSITY MANAGEMENT SYSTEM ======
1. Add Student
2. View Students
3. Edit Student
4. Add Course
5. View Courses
6. Enter Marks
7. Calculate GPA
8. Exit
Choose an option: 1
Enter Student ID: 10
Enter Student Name: Ali
Enter Age: 18
Enter Department: IDS
 Student added successfully!
```

Console Based Output:

This is the console based output for the university management system in which we can find students GPA by adding their marks



```
====== UNIVERSITY MANAGEMENT SYSTEM ======
1. Add Student
2. View Students
3. Edit Student
4. Add Course
5. View Courses
6. Enter Marks
7. Calculate GPA
8. Exit
Choose an option: 7
Enter Student ID: 10
🎓 GPA of Student 10: 3.77


====== UNIVERSITY MANAGEMENT SYSTEM ======
1. Add Student
2. View Students
3. Edit Student
4. Add Course
5. View Courses
6. Enter Marks
7. Calculate GPA
8. Exit
Choose an option: 7
Enter Student ID: 11
🎓 GPA of Student 11: 3.68
```

Key Functions:

- **main menu()**

Displays the main menu and takes user input to navigate through different options of the system.

- **add student()**

Collects student details such as name, roll number, and course, and stores them in memory.

- **view students()**

Displays all stored student records in a readable format on the console.

- **enter marks()**

Allows the user to enter subject marks for a specific student using the roll number.

- **edit student()**

Updates existing student information or marks based on user input.

- **calculate_gpa()**

Calculates GPA for students using stored marks and predefined grading criteria.

- **display_gpa()**

Displays calculated GPA along with student details.

- **search student()**

Finds and displays student records using a unique roll number.

- **validate input()**

Ensures that user inputs are correct and prevents invalid data entry.

- **exit program()**

Safely terminates the application after user confirmation.

How is the Backend Working?

The backend of the University Management System is implemented using Python with the Flask framework and works as the core controller of the application. It starts by running a Flask server that listens for requests from the frontend web pages. Each frontend page communicates with the backend using HTTP requests through the JavaScript Fetch API**.**

When a user submits data, such as adding a student or entering marks, the frontend sends the information to the backend in JSON format. The backend receives this data through predefined Flask routes and processes it accordingly. Student records and marks are stored temporarily in Python data structures like lists and dictionaries, since no database is used in this project.

For operations like searching a student, editing records, or calculating GPA, the backend retrieves the relevant data from memory using the student's roll number as a unique identifier. The backend then performs the required logic, such as updating records or calculating GPA based on marks, and sends the results back to the frontend as JSON responses.

Finally, the frontend dynamically updates the user interface using the data returned from the backend without reloading the page. In this way, the backend manages data processing, storage, and calculations, while ensuring smooth communication and real-time updates across all modules of the system.

JavaScript Functionality:

JavaScript plays a crucial role in adding interactivity and dynamic behavior to the University Management System. It is used to handle user actions, communicate with the backend, and update the web pages without reloading.

- JavaScript captures user input from forms such as student information and marks entry.
- It sends data to the backend using the Fetch API in JSON format.
- JavaScript receives responses from the Python backend and processes the returned data.
- It dynamically updates HTML tables to display student records, marks, and GPA reports.
- It enables real-time page updates without refreshing the browser.
- JavaScript handles search operations such as finding students using roll numbers.
- It clears form fields after successful submission to improve user experience.
- JavaScript controls page behavior such as button actions and input validation.
- It ensures smooth communication between frontend pages and backend APIs.
- JavaScript improves the responsiveness and usability of the system.

System Architecture

The system architecture of the University Management System follows a three-tier structure consisting of the presentation layer**,** application layer, and data layer. The presentation layer represents the frontend of the system and is developed using HTML, CSS, and JavaScript. It provides user interfaces such as forms, tables, and navigation menus that allow users to interact with the system through a web browser.

- The application layer is implemented using Python with the Flask framework.
- The backend handles request processing and business logic.
- Frontend and backend communicate using HTTP requests and JSON format.
- The backend processes student data, marks entry, record editing, and GPA calculation.
- The data layer uses Python data structures for temporary storage.
- No database is used to store information permanently.
- Data is cleared when the backend server is restarted.
- The architecture ensures separation of concerns between frontend and backend.
- The system is lightweight and suitable for academic and learning purposes.

Future Enhancements

- A database can be integrated to provide permanent data storage and persistence.
- User authentication and login functionality can be added for security.
- Role-based access control can be implemented for admin, faculty, and students.
- The system can be enhanced to support multiple users simultaneously.
- Advanced GPA calculation with grading policies can be introduced.
- Report generation features such as PDF or Excel export can be added.
- Search and filter options can be improved for faster data access.
- A responsive design can be implemented for mobile and tablet devices.
- Cloud deployment can be introduced for remote access and scalability.
- Backup and recovery mechanisms can be added to prevent data loss.
- Notification features such as email or SMS alerts can be included.
- Performance optimization can be implemented for handling large datasets.

Conclusion:

The University Management System successfully achieves its objective of managing student information, academic marks, and GPA calculation in an organized and efficient manner. The project demonstrates effective integration of frontend technologies with a Python-based backend to handle data processing and application logic. By using in-memory data storage instead of a database, the system remains simple and lightweight, making it suitable for academic and learning purposes. The modular design of the system improves maintainability and clarity, while dynamic data updates enhance user experience. Overall, the project provides a strong foundation for understanding web application development concepts and can be further extended with advanced features in the future.