STUDENT MANAGEMENT SYSTEM

A Mini Project Report

Submitted by

MUTHAMIL S - 23CDR104

NIRANJANA Y - 23CDR110

in partial fulfillment of the requirements

for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND DESIGN

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638060

MAY 2025

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638060

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN

BONAFIDE CERTIFICATE

Name : MUTHAMIL S - 23CDR104
NIRANJANA Y - 23CDR110

Course Code : 22CDL41

Course Name : DATABASE MANAGEMENT SYSTEM LABORATORY

Semester IV

Certified that this is a bonafide record of work for application projects done by the above students for **22CDL41 – DATABASE MANAGEMENT SYSTEM LABORATORY** during the academic year **2024 - 2025.**

Submitted for the Viva Voce Examination held on _____

Faculty In-Charge **Head of the Department**

STUDENT MANAGEMENT SYSTEM

- **ABSTRACT**
- **INTRODUCTION**
- **SYSTEM REQUIREMENT SPECIFICATION**
- **DATA FLOW DIAGRAM**
- **ER - DIAGRAM**
- **IMPLEMENTATION**
- **SAMPLE INPUT AND OUTPUT**
- **CONCLUSION AND FUTURE ENHANCEMENT**

ABSTRACT

The Student Management System (SMS) utilizes Structured Query Language (SQL) and the Flask framework to manage and streamline academic and administrative operations in educational institutions. By automating essential tasks such as student registration, course enrollment, attendance tracking, academic performance management, and feedback collection, the SMS enhances data accuracy and operational efficiency. A robust SQL-based database ensures secure, centralized storage and retrieval of critical student-related data, maintaining data integrity and consistency across all modules.

This system reduces manual workload, minimizes human errors, and facilitates fast and reliable access to student information. It supports role-based access for students, faculty, and administrators, ensuring that each user interacts only with relevant modules and data. The SMS also includes built-in analytics and reporting features, aiding in academic planning and institutional decision-making.

This implementation of a Student Management System using Python (Flask) and MySQL includes the creation of essential tables: Students, Courses, Enrollments, Attendance, Academic Performance, Feedback, Staff, Departments, and Users. It supports full CRUD functionality, secure user authentication, and scalable data management. The system is designed to simplify institutional processes and provide a comprehensive, efficient, and user-friendly platform for managing educational data

1. INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

The project titled “Student Management System” is developed to enhance academic and administrative operations in educational institutions. It uses Python (Flask) for the backend and MySQL as the database. The system captures and processes student data for registration, course enrollment, attendance, academic performance, and feedback.

By providing role-based access to students, faculty, and administrators, it ensures secure and efficient data management. The system automates repetitive tasks, supports real-time data tracking, and reduces administrative workload, thereby improving communication and decision-making within the institution.

1.2 ABOUT THE EXISTING SYSTEM - MANUAL SYSTEM:

Most institutions still depend on manual or semi-digital methods like spreadsheets and documents to manage student information. These methods are inefficient , error-prone and not scalable for large datasets.

Processes like registration , attendance tracking and performance monitoring take significant
Time and often result in inconsistent records. A digital Student Management System is needed to streamline these processes , maintain accurate data and support institutional growth.

1.3 DRAWBACKS OF THE EXISTING SYSTEM

- Tedious and inefficient handling of large volumes of student data
- High time consumption for basic academic tasks
- Difficulty in maintaining and retrieving historical data
- No centralized or integrated system for academic records
- Lack of role-based access and data security
- Delays in generating reports and making informed decisions**.**

2. SYSTEM REQUIREMENT SPECIFICATION

2.1 HARDWARE SPECIFICATION

- Processor : Intel Core i3 or higher
- RAM : 4 GB or above
- Hard Disk : 5 GB
- Monitor : 15'' HD Color Monitor
- Keyboard : Standard 104 keys
- Pointing Device : Optical Mouse

2.2 SOFTWARE SPECIFICATION

- Operating System : Windows 10
- Programming Language : Python (Flask Framework)
- Database : MySQL
- Database Management Tool : phpMyAdmin
- Web Server : Apache (via XAMPP/WAMP)
- IDE : Visual Studio Code
- Browser : Chrome / Firefox (for testing web interface

3. DATA FLOW DIAGRAM

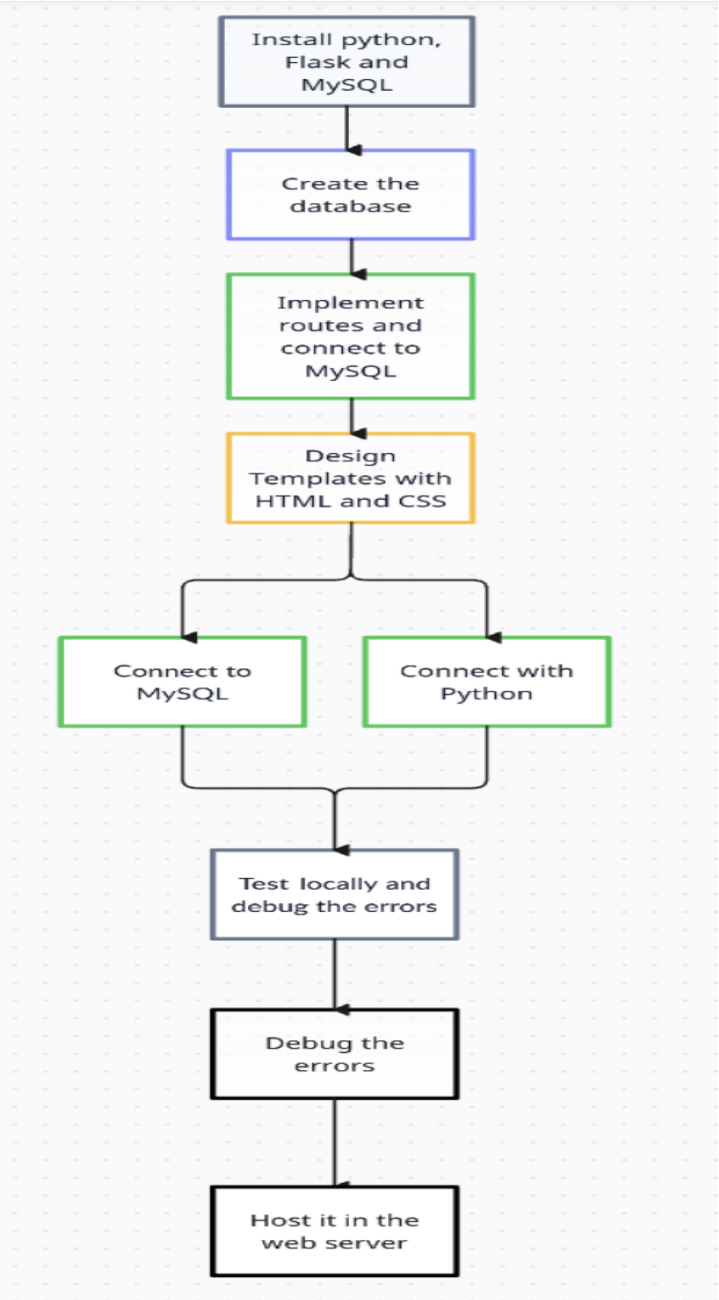

Fig 3.1.Data Flow diagram for Student Management System

4. ER - DIAGRAM

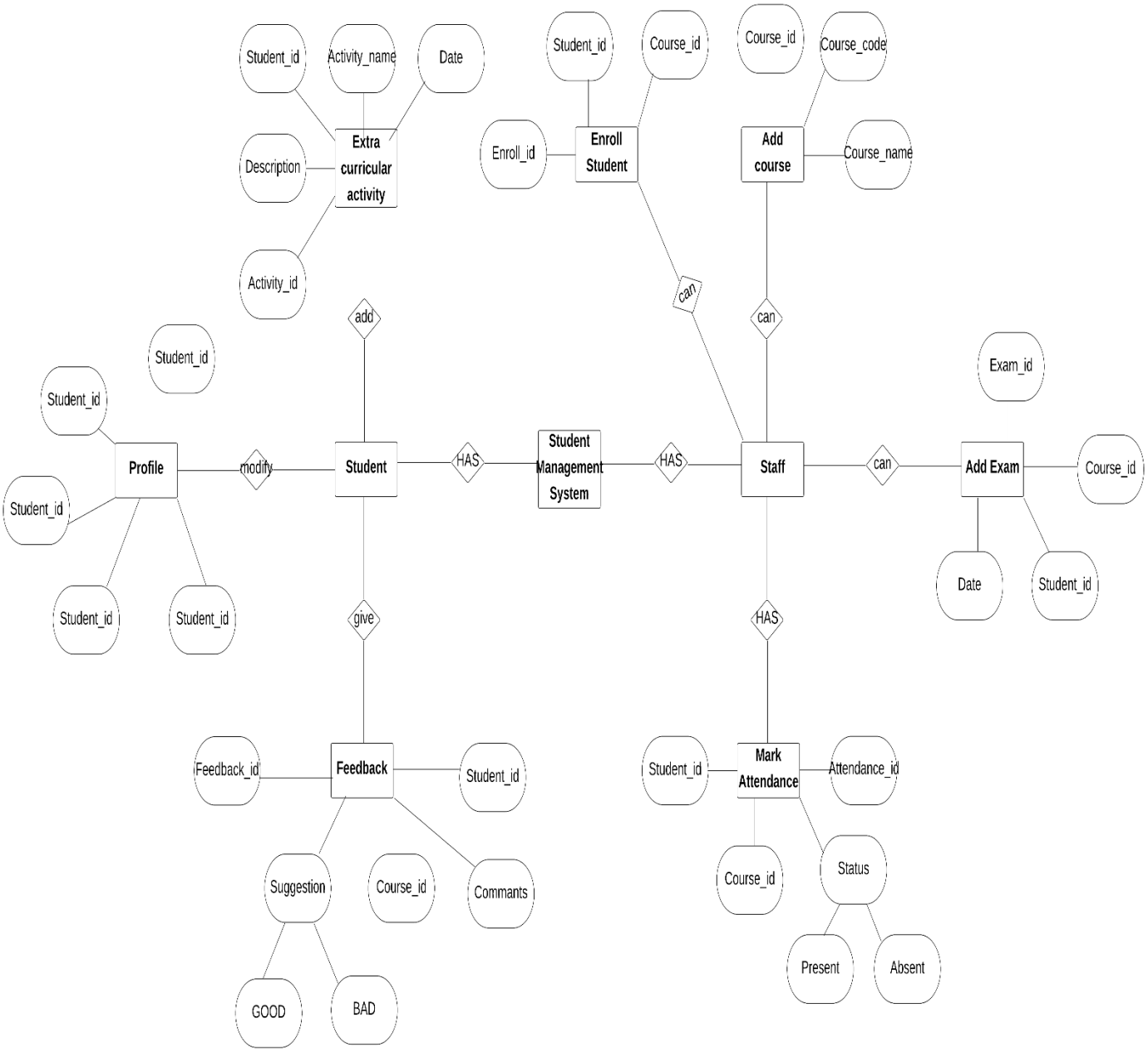

Fig.4.1. ER Diagram for Student Management System

5. IMPLEMENTATION

Implementing a Student Management System using Python and MySQL is a comprehensive process that begins with setting up the development environment. First, download and install MySQL and ensure that the MySQL service is running. Create a project directory for your Python scripts, naming it appropriately, such as student_management_system.

Next, design the database schema to support the core functionalities of the system. Access the MySQL command line or use a tool like phpMyAdmin to create a new database named student_management. Define tables for students, courses, enrollments, grades, attendance, departments, users, and feedback with appropriate fields and relationships. For instance, the Student table includes fields like student_id, name, dob, email, phone, department, etc., while the Course table includes fields such as course_id, course_name, course_code, and more. Additionally, the Enrollments table includes fields such as enrollment_id, student_id, course_id, and enrollment_date, and the Grades table includes fields like grade_id, student_id, course_id, marks, and grade.

Develop the backend using Python to manage database interactions. Start by creating a db_connection.py file to establish a connection to the MySQL database. This file includes connection parameters and error handling to ensure reliable database access. Implement functionality for managing students by creating Python functions in scripts like add_student.py, view_students.py, edit_student.py, and delete_student.py. These scripts handle user inputs, execute SQL queries, and provide feedback to users. For example, add_student.py includes a function for inputting student details and inserting the data into the database upon submission.

Similarly, develop scripts to manage courses, enrollments, grades, and attendance. For instance, add_course.py adds a new course, while view_enrollments.py displays students enrolled in specific courses. Each functionality should be divided into manageable scripts for clear structure and ease of maintenance.

Next, create a user interface for the system. You can either create a command-line interface (CLI) for simplicity or a graphical user interface (GUI) using libraries like Tkinter or Flask for web-based development. Develop a main.py file that serves as the main entry point, including a menu for different functionalities such as adding students, enrolling in courses, marking attendance, assigning grades, and providing feedback. Enhance the user experience by ensuring the interface is intuitive and user-friendly.

Once the basic functionalities are in place, test the system by running main.py and verifying that all operations work correctly. Ensure that functionalities such as adding, viewing, editing, and deleting student records work as expected. Verify that students can enroll in courses, attendance can be marked accurately, grades are assigned properly, and feedback is submitted and retrieved correctly.

After the basic system is fully functional, consider adding advanced features such as:

Reporting and Analytics: Generate reports for student attendance, performance, and grades.

Automated Notifications: Notify students about important events, exam dates, or deadlines.

Real-Time Updates: Allow for real-time updates of student records, attendance, and grades.

Additionally, consider implementing user authentication to ensure secure access to the system. Only authorized personnel, such as faculty members and administrators, should be able to modify sensitive data like grades and attendance records.

This structured approach ensures a robust, scalable, and user-friendly student management system that can efficiently handle student data, courses, enrollments, attendance, grades, and feedback. By continuously enhancing the system with new features and functionalities, you can provide an effective, comprehensive solution for managing student information in any educational institution

SAMPLE TABLES

Table 1: STUDENTS

- **Table Name: students**
- **Primary Key: student_id**
- **Description: Stores student details like personal information and department.**

Field Name	Data Type	Description
student_id	INT	Primary key, unique student identifier
Name	VARCHAR(100)	Name of the student
Dob	DATE	Date of birth
Email	VARCHAR(100)	Student email ID
Phone	VARCHAR(15)	Student phone number
department	VARCHAR(50)	Department the student belongs to

Table 2: COURSES

- **Table Name: courses**
- **Primary Key: course_id**
- **Description: Stores information about courses offered to students.**

Field Name	Data Type	Description
course_id	INT	Primary key, unique course identifier
course_name	VARCHAR(100)	Name of the course
course_code	VARCHAR(10)	Unique course code

Table 3: ENROLLEMENTS

- **Table Name: enrollments**
- **Primary Key: enrollment_id**
- **Description: Manages the relationship between students and the courses they are enrolled in.**

Field Name	Data Type	Description
enrollment_id	INT	Primary key, unique enrollment identifier
student_id	INT	Foreign key referencing students(student_id)
course_id	INT	Foreign key referencing courses(course_id)

Table 4: ATTENDANCE

- **Table Name: attendance**
- **Primary Key: attendance_id**
- **Description: Tracks the attendance of students in each course.**

Field Name	Data Type	Description
attendance_id	INT	Primary key, unique attendance identifier
student_id	INT	Foreign key referencing students(student_id)
course_id	INT	Foreign key referencing courses(course_id)
attendance_date	DATE	Date of attendance
status	ENUM('Present', 'Absent')	Attendance status (Present or Absent)

Table 5: ACADEMIC PERFORMANCE

- **Table Name: academic_performance**
- **Primary Key: performance_id**
- **Description: Stores the grades for students in each course.**

Field Name	Data Type	Description
performance_id	INT	Primary key, unique performance record identifier
student_id	INT	Foreign key referencing students(student_id)
course_id	INT	Foreign key referencing courses(course_id)
Grade	VARCHAR(2)	The grade the student received (e.g., 'A', 'B+')

Table 6: FEEDBACK

- **Table Name: feedback**
- **Primary Key: feedback_id**
- **Description: Stores feedback given by students for each course.**

Field Name	Data Type	Description
feedback_id	INT	Primary key, unique feedback record identifier
student_id	INT	Foreign key referencing students(student_id)
course_id	INT	Foreign key referencing courses(course_id)
feedback_text	TEXT	The feedback provided by the student
submission_date	DATE	The date the feedback was submitted

Table 7: DEPARTMENTS

- **Table Name: departments**
- **Primary Key: department_id**
- **Description: Stores information about the departments in the institution.**

Field Name	Data Type	Description
department_id	INT	Primary key, unique department identifier
department_name	VARCHAR(100)	Name of the department

Table 8: USERS

- **Table Name: users**
- **Primary Key: user_id**
- **Description: Stores user login details, supporting authenticati**

Field Name	Data Type	Description
user_id	INT	Primary key, unique user identifier
username	VARCHAR(50)	Username for login
password	VARCHAR(100)	Password for login
email	VARCHAR(100)	Email ID for the user
user_type	ENUM('admin', 'student', 'faculty')	Role of the user (admin, student, faculty)

Table 9: STUDENT DEPARTMENT ALLOCATION

- **Table Name: student_department**
- **Primary Key: allocation_id**
- **Description: Stores data about which student belongs to which department and the date of joining.**

Field Name	Data Type	Description
allocation_id	INT	Primary key, unique allocation identifier
student_id	INT	Foreign key referencing students(student_id)
department_id	INT	Foreign key referencing departments(department_id)
join_date	DATE	The date the student joined the department

6. SAMPLE INPUT AND OUTPUT

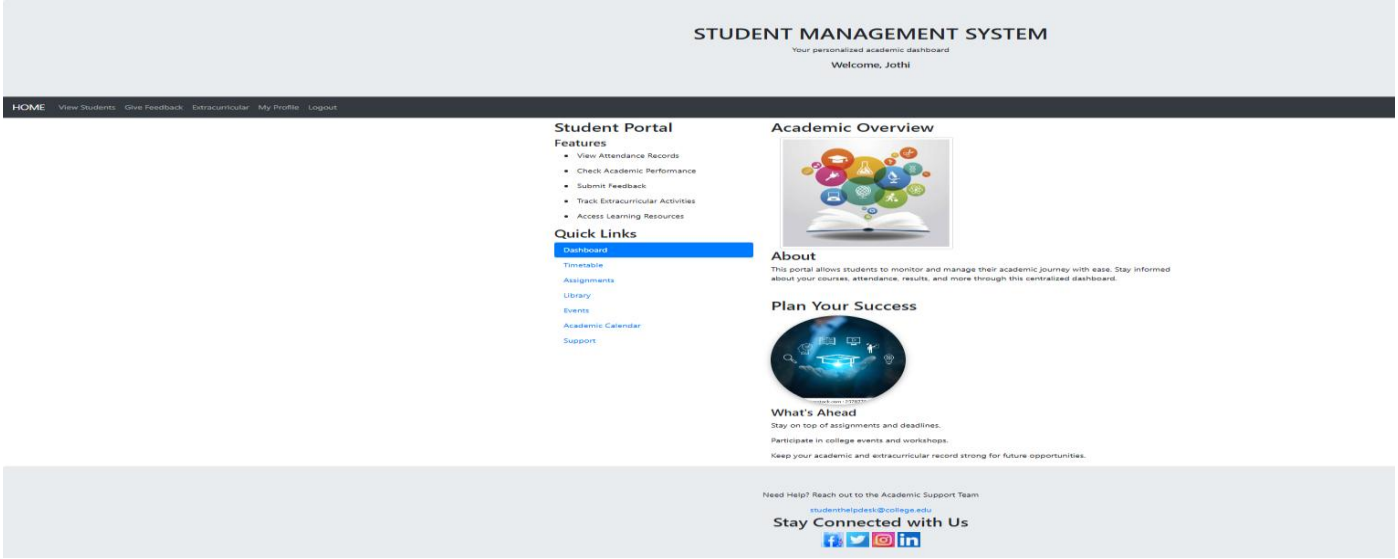

Fig 6.1. Home Page (For Students)

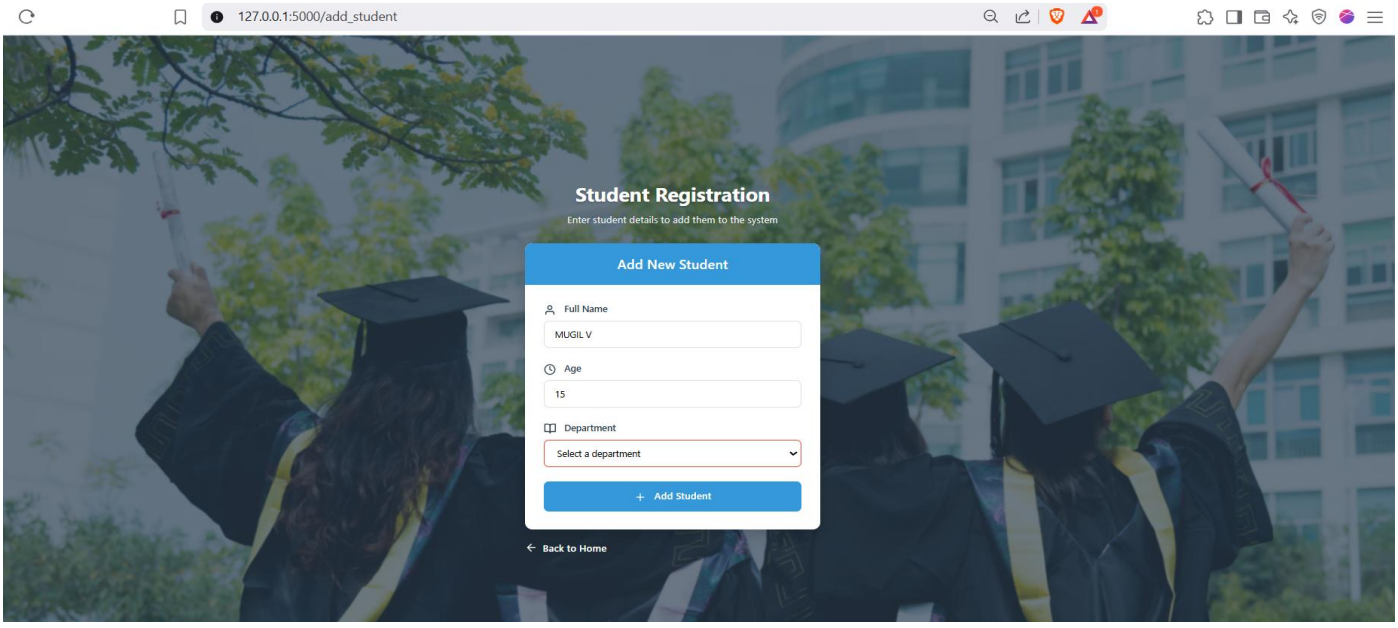

Fig 6.2. Student Registration Page

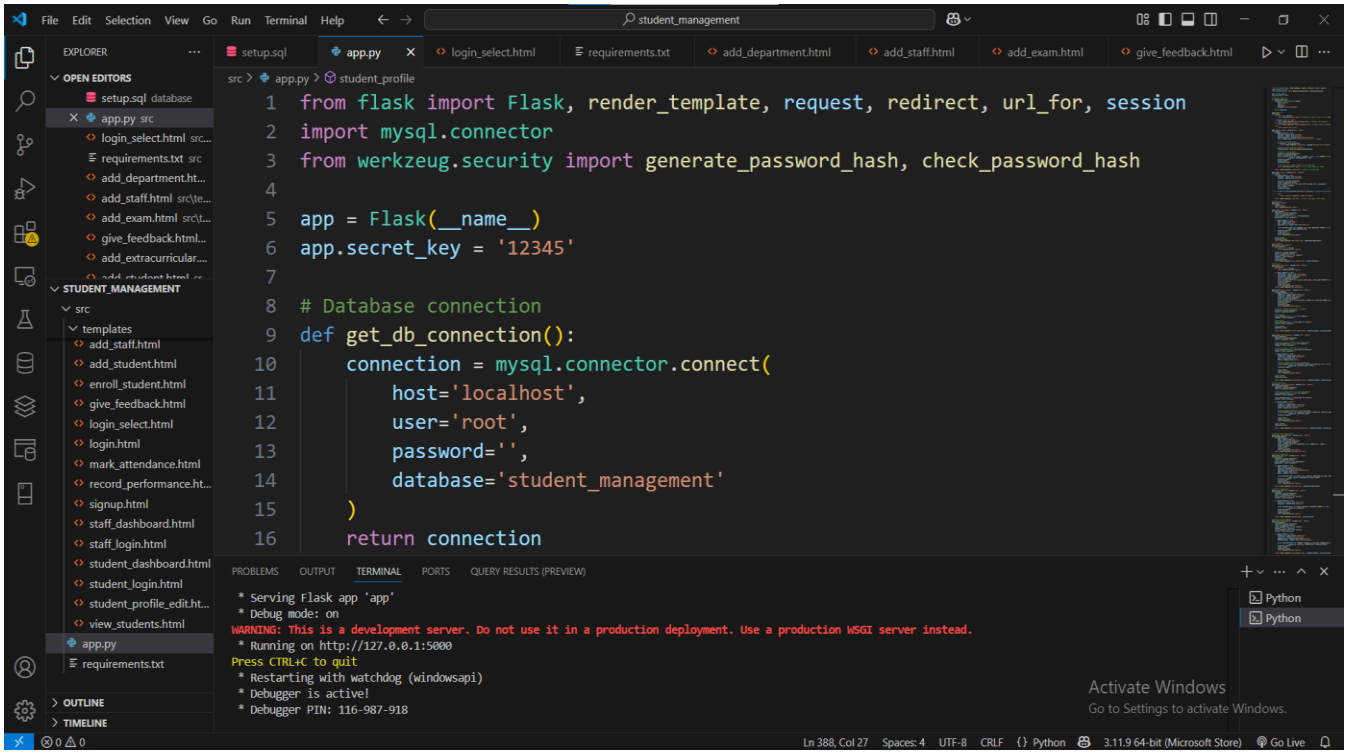

Fig 6.3. Main Code for Student Management System

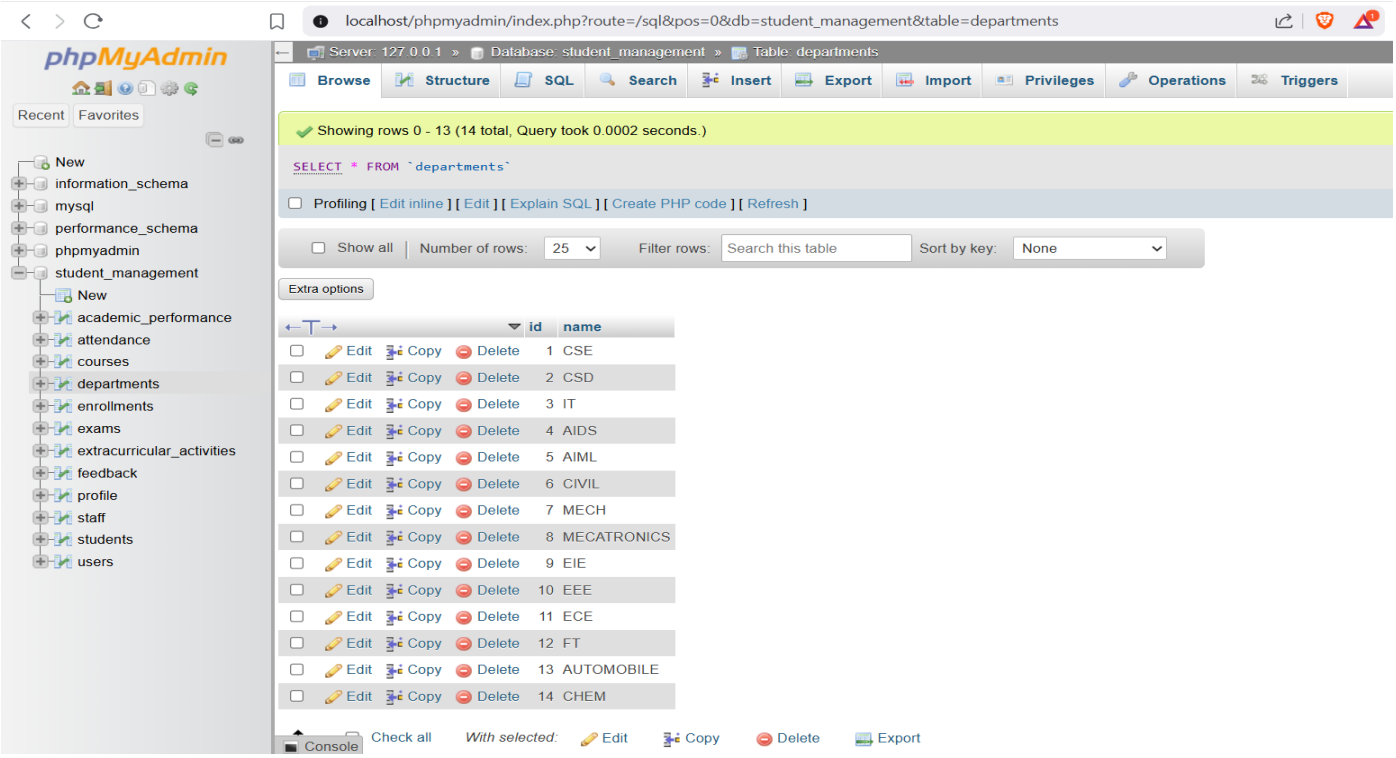

Fig 6.4. Data Stored in Database for Student Management System

7. CONCLUSION

The Student Management System (SMS) effectively addresses all core requirements for managing student records, courses, enrollments, grades, and attendance. The system has been tested with sample data, ensuring smooth functionality and optimal performance.

This user-friendly system allows administrators to add, view, edit, and delete student details, assign courses, track attendance, and manage grades. It also supports comprehensive report generation, which aids in quick and informed decision-making. By automating processes like student enrollment and grade assignment, the system reduces manual errors and administrative workload, allowing educational institutions to focus more on academic excellence and student well-being. The system overcomes issues found in older manual systems, such as data inconsistency and inefficiency.

FUTURE ENHANCEMENT

- User Authentication: Implement role-based access to ensure data security.
- Advanced Reporting: Add features for detailed performance and attendance analytics.
- System Integration: Integrate with other campus systems (e.g., HR, library) for a unified platform.
- Automated Notifications: Send automated reminders for exams, assignments, and grade updates.
- Mobile App: Develop a mobile version for easy access to student data.
- Cloud Deployment: Migrate the system to the cloud for improved scalability and remote access.

REFERENCES

1. MySQL. (2023). MySQL Documentation. Retrieved from *https://dev.mysql.com/doc/*

2. Python Software Foundation. (2023). Python Documentation. Retrieved from *https://docs.python.org/3/*

3. MySQL Connector/Python Developer Guide. (2023). Retrieved from *https://dev.mysql.com/doc/connector-python/en/*

4. GitHub. (2023). GitHub Documentation. Retrieved from *https://docs.github.com*

5. W3Schools. (2023). SQL Tutorial. Retrieved from *https://www.w3schools.com/sql/*