UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+
Accredited University

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Mini Project Report

On

Manage student information efficiently using Flask, MySQL, and Bootstrap.

SUBMITTED BY:

Vikash

UID:25MCI10384

CLASS: MCA (AI&ML)

SECTION: 25MAM6-A

SUBMITTED TO:

Ms. Sweta

Table of Content

1.Acknowledgement

I would like to express my heartfelt gratitude to all those who contributed to the successful development of Manage student information efficiently using Flask, MySQL, and Bootstrap.

First and foremost, I am thankful to my mentors and instructors for their continuous guidance and encouragement throughout this project. Their insights into web development and system design have been invaluable in shaping the technical direction of this work.

I would also like to acknowledge my peers and friends for their constructive feedback and support during the testing and debugging phases. Their suggestions helped improve the user experience and functionality of the platform.

Lastly, I am grateful for the open-source community and the developers behind the technologies used in this project. Tools like HTML, MySQL, Bootstrap, and CSS have made it possible to bring this idea to life with both functionality and creativity.

2. Introduction

The **Student Record Management System (SRMS)** is a web-based application developed to manage and maintain student data efficiently.
In educational institutions, keeping track of student details such as name, age, and course manually can be time-consuming and prone to errors.

This project aims to automate that process through a simple yet powerful web-based platform. It allows the user to perform essential operations like adding, viewing, updating, and deleting student information.

The system integrates **Python Flask framework** for backend functionality, **MySQL** for database management, and **HTML, CSS, and Bootstrap** for the user interface. This makes it lightweight, reliable, and user-friendly.

3. Background

Before this project, most student data management processes in schools or colleges were manual. Teachers or administrators often relied on handwritten registers or Excel sheets, which could easily become disorganized or lost.

With the advancement of web technologies, there is now a need for a centralized system that provides:

- Easy access to student data
- A way to update records quickly
- Secure and reliable data storage

The **Student Record Management System** was designed to solve these real-world challenges by using programming and database management technologies effectively.

4. Objective of the Project

The main objective of the Student Record Management System is to **create a structured, efficient, and user-friendly web platform** to manage student records digitally.

Specific Objectives:

- To automate the manual record-keeping process.
- To provide CRUD (Create, Read, Update, Delete) functionalities.
- To design a web interface that allows easy access to all features.
- To use Python and MySQL for handling data storage and retrieval.
- To demonstrate full-stack development concepts practically.

5. Purpose of the Project

The purpose of developing this project is to help educational institutions maintain accurate student information in a systematic manner.

It is designed for teachers, school administrators, and data entry operators who need to manage a large number of student details efficiently.

The system can replace paper-based record systems and reduce human error while improving data accuracy and accessibility.

6. Technologies Used

Technology	Description
Python (Flask Framework)	Flask is a micro web framework used for building web applications easily in Python.
MySQL Database	Used to store and manage all student-related information.
HTML	Used to design the structure of the web pages.
CSS (Bootstrap)	Used to create responsive and attractive layouts.
Jinja Templates	Used for connecting backend data dynamically with frontend HTML pages.

7. Requirements

Hardware Requirements

- Processor: Intel i3 or above
- RAM: Minimum 4 GB
- Hard Disk: 250 GB or higher
- Display: HD Resolution

Software Requirements

- Operating System: Windows 10 / 11 or Linux
- Python Version: 3.11 or above
- Flask Framework
- MySQL Server / XAMPP
- Visual Studio Code or PyCharm IDE
- Web Browser: Chrome, Edge, or Firefox

8. System Design

Architecture of the System:

User Interface (HTML, CSS, Bootstrap)

↓

Application Layer (Flask - Python)

↓

Database Layer (MySQL)

- **Frontend Layer:** Handles user interaction through web pages and forms.
- **Backend Layer:** Processes requests and interacts with the database.
- **Database Layer:** Stores all student data securely in structured tables.

Database Table Design

Table Name: student

Field	Data Type	Description
id	INT (Auto Increment)	Unique ID of each student
name	VARCHAR(100)	Student Name
age	INT	Student Age
course	VARCHAR(100)	Student Course Name

9. Source Code (Main File – app.py)

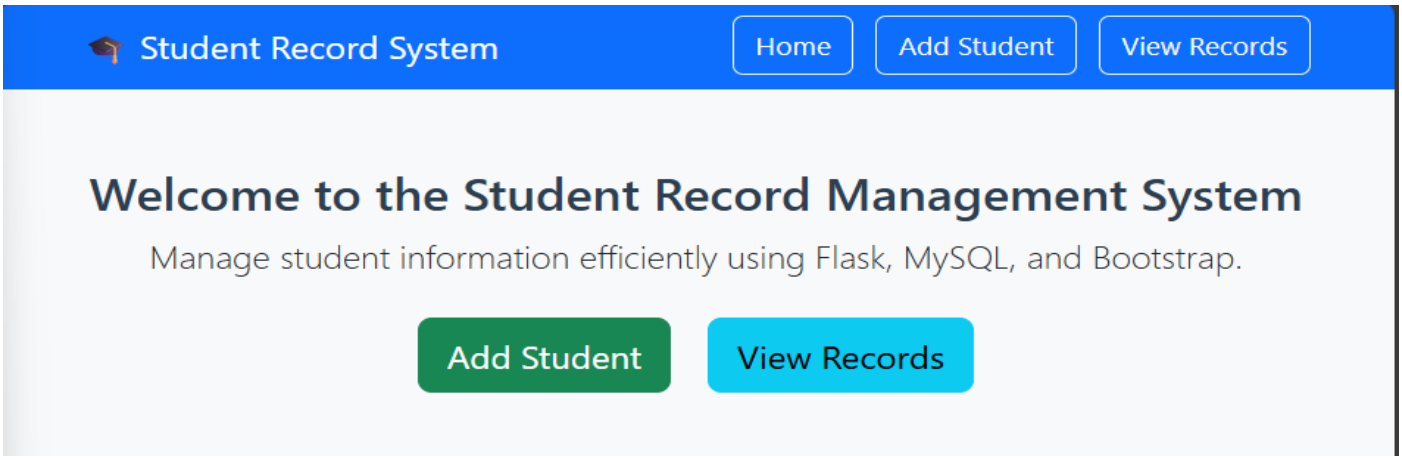
```
from flask import Flask, render_template, request, redirect
import mysql.connector

app = Flask(__name__)

# Database connection function
def get_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="",  # Add your MySQL password if any
        database="studentdb"
    )

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/add', methods=['GET', 'POST'])
```

```
def add_student():
    if request.method == 'POST':
        name = request.form['name']
        age = request.form['age']
        course = request.form['course']
        conn = get_connection()
        cur = conn.cursor()
        cur.execute("INSERT INTO student (name, age, course) VALUES (%s, %s, %s)", (name, age, course))
        conn.commit()
        conn.close()
        return redirect('/view')
    return render_template('add.html')

@app.route('/view')
def view_students():
    conn = get_connection()
    cur = conn.cursor()
    cur.execute("SELECT * FROM student")
    students = cur.fetchall()
    conn.close()
    return render_template('view.html', students=students)

@app.route('/update/<int:id>', methods=['GET', 'POST'])
def update_student(id):
    conn = get_connection()
    cur = conn.cursor()
```

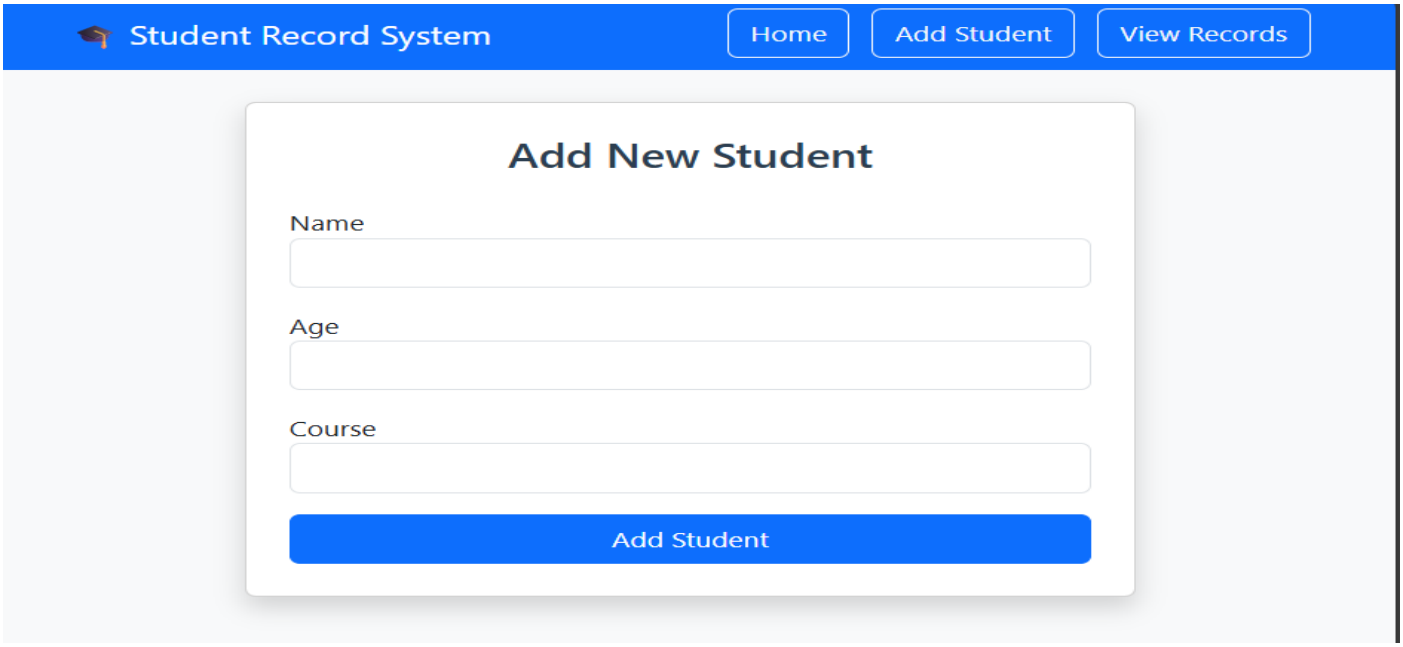
```
    if request.method == 'POST':
        name = request.form['name']
        age = request.form['age']
        course = request.form['course']
        cur.execute("UPDATE student SET name=%s, age=%s, course=%s
WHERE id=%s", (name, age, course, id))
        conn.commit()
        conn.close()
        return redirect('/view')
    else:
        cur.execute("SELECT * FROM student WHERE id=%s", (id,))
        student = cur.fetchone()
        conn.close()
        return render_template('update.html', student=student)

@app.route('/delete/<int:id>')
def delete_student(id):
    conn = get_connection()
    cur = conn.cursor()
    cur.execute("DELETE FROM student WHERE id=%s", (id,))
    conn.commit()
    conn.close()
    return redirect('/view')

if __name__ == '__main__':
    app.run(debug=True)
```

10. Output Screens

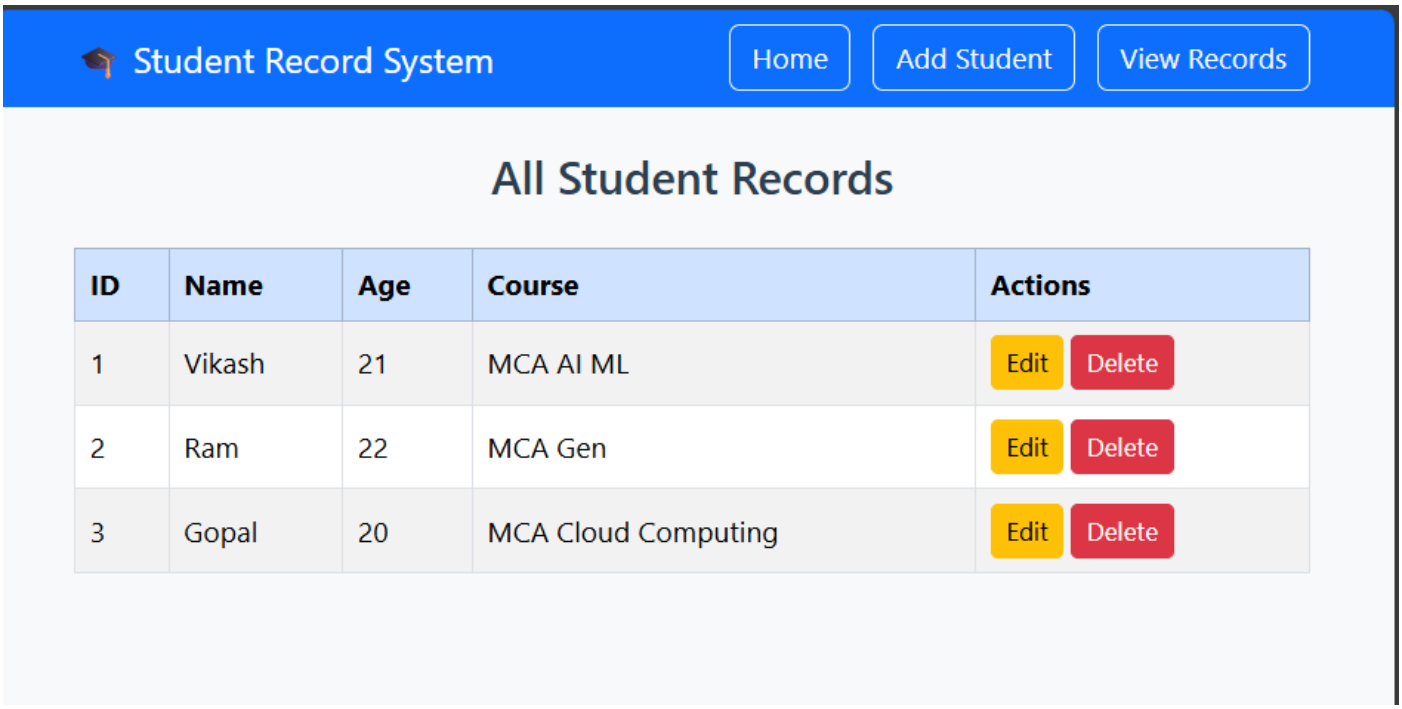
A. Main Window



B. Add New Student



C. View Student



11. Functionalities

Function	Description
Add Student	Inserts a new student record into the MySQL database.
View Students	Displays all student records in a tabular format.
Update Student	Edits existing student information.
Delete Student	Removes the student record permanently.
Responsive UI	Works properly on both mobile and desktop browsers.

12. Conclusion

The **Student Record Management System** project successfully simplifies student data handling.
It allows users to perform all necessary operations through a web interface efficiently and securely.
The system proves how Python Flask and MySQL can be combined to create dynamic web applications with minimal resources.

13. Future Work

In the future, the system can be enhanced with:

- Admin login and authentication system
- Student profile photo upload
- Search and filter features
- Report generation in PDF or Excel format
- Cloud deployment for online access

14. Learning Outcomes

During the development of this project, the following skills and knowledge were gained:

- Understanding of **web development using Flask**
- Working with **MySQL database connectivity**
- Creation of **CRUD applications**
- Use of **Bootstrap and CSS** for UI design
- Hands-on experience in **Python full-stack development**