Project Documentation: Student Information Management

1. Overview

This project is designed to manage student information using a **Flask web application**. It retrieves data from a MySQL database and displays it in the browser. The project contains three main components:

- **HTML Files**: sample.html for the form, and view_all_students.html for displaying student data.
- **Python Backend**: app.py, which connects to the MySQL database, retrieves data, and renders the templates.
- **Database**: students table in a MySQL database to store the student information.

2. File Structure

```
StudentInfoProject/
│
├── app.py                  # Python backend file
├── templates/
│   ├── sample.html             # Form for entering student data
│   └── view_all_students.html    # Display all student records
├── static/
│   └── style.css               # Optional: CSS file for styling (if needed)
└── students.sql                # MySQL database table creation script
```

3. HTML Files

3.1. sample.html

This file provides a **form** for entering student information like ID, name, age, and grade.

Purpose: This HTML file is used to create a form where users can input student information.

Features:

- **Form Elements**: Contains fields for entering a student's ID, name, age, and grade.

- **Styling**: Uses basic CSS to style the form with a white background, padding, and a subtle shadow.
- **Submission**: The form sends data to the /add_student endpoint on submission, which is handled by the Flask backend.

User Experience:

- **Layout**: The form is centered and styled to be user-friendly, with clearly labeled input fields.
- **Button**: Includes a submit button that adds the student information to the database when clicked.

Functionality: Users enter student details into the form. When the "Add Student" button is clicked, the form data is sent to the Flask backend (/add_student) via POST request.

3.2. view_all_students.html

This file displays all student information retrieved from the **students** database in a tabular format.

Purpose: This HTML file is designed to display a list of all students in a table format.

Features:

- **Table Display**: Shows student information in a tabular format with columns for ID, name, age, and grade.
- **Dynamic Content**: Uses Jinja templating to dynamically insert student data from the Flask backend.
- **Styling**: Includes CSS for a clean, readable table with alternating row colors and hover effects.

User Experience:

- **Layout**: Presents data in a full-width table with a fixed background image.
- **Design**: The table is styled to be clear and easy to read, with headers highlighted and alternating row colors for better visual separation.

Functionality: This table is dynamically populated with data from the students database. The students variable is passed from the Flask backend.

4. Python Backend (app.py)

This file is the core backend of the application. It handles the routes, connects to the MySQL database, and manages data flow.

Purpose: This Python script sets up a Flask web application to handle student information. It connects to a MySQL database to add and retrieve student records.

Key Features:

- **Database Connection**: Connects to a MySQL database named yukesh to store and retrieve student data.
- **Routes**:
  - /: Displays a form for adding new student information.
  - /add_student: Handles form submission, adding student data to the database.
  - /view_students: Retrieves all student records from the database and displays them in a table.

Functionality:

- **Form Handling**: Collects and processes student data submitted via a form.
- **Database Operations**: Inserts new student records and queries existing records.
- **Template Rendering**: Uses Jinja2 templates to render HTML pages with dynamic data.

Execution:

- The application runs with debugging enabled, which provides detailed error messages and auto-reloads on code changes.

Key Functionalities:

1. / Route: Displays the **student form** (sample.html).
2. /add_student Route: Accepts form data and inserts the student into the MySQL **students** table.
3. /view_students Route: Queries the **students** table and displays all records on the view_all_students.html page.

5. Database (MySQL)

This section covers the creation of the **students** table in your MySQL database.

5.1. Creating the Database

```
CREATE DATABASE table_name;
USE table_name;
```

5.2. Creating the Students Table

```
CREATE TABLE table_name (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    age INT,
    grade VARCHAR(50)
);
```

- **Columns**:
  - id: Unique ID for each student.
  - name: Student's name.
  - age: Student's age.
  - grade: Student's grade.

5.3. Sample Data (Optional)

To insert sample data for testing:

```
INSERT INTO table_name (id, name, age, grade) VALUES (1, 'John Doe', 15, '10th Grade');
INSERT INTO table_name (id, name, age, grade) VALUES (2, 'Jane Smith', 14, '9th Grade');
```

6. Conclusion

This project allows users to add and view student information, with the data stored in a MySQL database and served to the user through Flask.