Flask project using:
✓ Flask API
✓ JSON
✓ HTML + CSS
✓ Jinja2 templates
✓ Fetching JSON data from API
✓ Displaying student database on a webpage

APP DESCRIPTION
A simple **Student Database Web App** that:
- Uses Flask API to store and retrieve students (JSON format)
- A webpage fetches this API data and displays it in a **HTML table**
- Uses Flask templates (templates/ folder)
- Uses Bootstrap for clean styling
- No database — uses a simple Python list

Below is a **simple, clean, beginner-friendly Flask project** that uses:
✓ Flask API
✓ JSON
✓ HTML + CSS
✓ Jinja2 templates
✓ Fetching JSON data from API
✓ Displaying student database on a webpage
This is an ideal classroom example.

APP DESCRIPTION
A simple **Student Database Web App** that:
- **Uses Flask API to store and retrieve students (JSON format)**
- **A webpage fetches this API data and displays it in a HTML table**
- **Uses Flask templates (templates/ folder)**
- **Uses Bootstrap for clean styling**
- **No database — uses a simple Python list**

FINAL OUTPUT
- **A webpage that shows:**
- **Student ID**
- **Student Name**
- **Age**
- **Course**

STEP 1 — PROJECT STRUCTURE
Create folders like this:
flask_student_app/

```
|
├── app.py
|
└── templates/
    └── students.html
```

STEP 2 — Create Virtual Environment (Recommended)
Windows:

```
>>python -m venv venv
>>venv\Scripts\activate
```

Mac/Linux:

```
>>python3 -m venv venv
>>source venv/bin/activate
```

STEP 3 — Install Flask

```
>>pip install flask
```

STEP 4 — Write Flask API (app.py)
File: app.py

```
from flask import Flask, jsonify, render_template

app = Flask(__name__)

# Temporary student list (as database)
students = [
   {"id": 1, "name": "Alice", "age": 20, "course": "Computer Science"},
   {"id": 2, "name": "Bob", "age": 22, "course": "Electronics"},
   {"id": 3, "name": "Charlie", "age": 21, "course": "Mechanical"}
]

# --------------------------------------
# API to return JSON student data
# --------------------------------------
@app.route('/api/students')
def api_students():
   return jsonify(students)

# --------------------------------------
# Webpage to display student table
# --------------------------------------
@app.route('/')
def home():
   return render_template("students.html")
```

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

STEP 5 — Create HTML Template

File: templates/students.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Student Database</title>

    <!-- Simple styling using Bootstrap CDN -->
    <link rel="stylesheet"
          href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

    <style>
        body {
            margin: 30px;
        }
        h1 {
            color: #4a148c;
            margin-bottom: 20px;
        }
    </style>
</head>
<body>

    <h1>Student Database</h1>

    <table class="table table-bordered table-striped" id="studentTable">
        <thead class="table-dark">
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Age</th>
                <th>Course</th>
            </tr>
        </thead>
        <tbody>
            <!-- Data will be added here by JavaScript -->
        </tbody>
    </table>

    <!-- JavaScript to fetch and display JSON data -->
    <script>
        fetch("/api/students")
        .then(response => response.json())
        .then(data => {
```

```
        const tableBody = document.querySelector("#studentTable tbody");

        data.forEach(student => {
          const row = `<tr>
                        <td>${student.id}</td>
                        <td>${student.name}</td>
                        <td>${student.age}</td>
                        <td>${student.course}</td>
                      </tr>`;
          tableBody.innerHTML += row;
        });
      });
    </script>

</body>
</html>
```

STEP 6 — Run the App
Inside project folder:
>>python app.py
You will see:
Running on http://127.0.0.1:5000/
Open a browser:
http://127.0.0.1:5000/
You will see the student table displayed!

WHAT YOU LEARN FROM THIS EXAMPLE
Python + Flask

- Routing
- JSON APIs
- Serving templates

HTML Concepts Used
✓ <table>
✓ <thead>, <tbody>
✓ Linking Bootstrap CSS
✓ Fetch API (JavaScript)
JSON Concepts
✓ JSON serialization
✓ Fetching JSON from Flask
✓ Rendering dynamic content

NEXT LEVEL (If you want)
I can add:

✓ Add Student Page (HTML form → POST API)
✓ Edit/Delete Student
✓ SQLite database
✓ Flask Blueprint structure
✓ Bootstrap UI improvements
✓ API documentation