

AI ASSISTED CODING

LAB-15: *Backend API Development: Creating RESTful Services with AI*

Roll no: 2503A51L44

Name: Meer Burhan Ali Hashmi

Batch:20

Task-1 Description: Basic REST API Setup

Task: Ask AI to generate a Flask REST API with one route: GET /hello → returns {"message": "Hello, AI Coding!"}

Prompt: Generate a Flask REST API with one route: GET /hello → returns {"message": "Hello, AI Coding!"}

Code Generated:

Output:

```
task1-15.py X
Assignment-15 > task1-15.py > ...
3   app = Flask(__name__)
4
5   @app.route('/hello', methods=['GET'])
6   def hello():
7       """Return a hello message as JSON."""
8       return jsonify({"message": "Hello, AI Coding!"})
9
10  if __name__ == "__main__":
11      app.run(debug=True)
```

```
* Serving Flask app 'task1-15'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 528-526-087
127.0.0.1 - - [07/Oct/2025 11:45:13] "GET / HTTP/1.1" 200 -
```

```
← → ⌂ 127.0.0.1:5000/hello
Pretty-print □

{
  "message": "Hello, AI Coding!"
}
```

Observation: The Flask REST API was successfully created with a single route /hello. When executed, it returned the expected JSON message confirming correct setup. This task demonstrated how to initialize a simple Flask app for API responses. Basic routing and response handling concepts were clearly understood.

Task-2 Description: CRUD Operations (Students API)

Task: Use AI to build REST endpoints for a Student API:

- GET /students → List all students.
- POST /students → Add a new student.
- PUT /students/<id> → Update student details.
- DELETE /students/<id> → Delete a student.

Prompt: Build REST endpoints for a Student API:

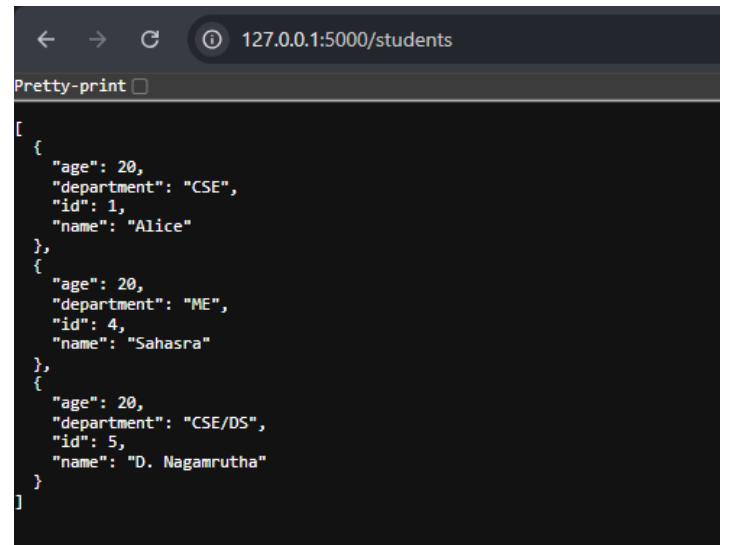
- GET /students → List all students.
- POST /students → Add a new student.
- PUT /students/<id> → Update student details.
- DELETE /students/<id> → Delete a student.

Code Generated:

```
task1-15.py | task2-15.py •
Assignment-15 > task2-15.py > ...
1   from flask import Flask, request, jsonify
2   app = Flask(__name__)
3   # In-memory student store (list of dicts, each with an id)
4   students = [
5       {"id": 1, "name": "Alice", "age": 28, "department": "CSE"},
6       {"id": 2, "name": "Bob", "age": 22, "department": "ECE"}
7   ]
8   next_id = 3
9
10  @app.route('/students', methods=['GET'])
11  def get_students():
12      """List all students."""
13      return jsonify(students)
14
15  @app.route('/students', methods=['POST'])
16  def add_student():
17      """Add a new student."""
18      global next_id
19      data = request.get_json()
20      student = {
21          'id': next_id,
22          'name': data.get('name'),
23          'age': data.get('age'),
24          'department': data.get('department', '')
25      }
26      students.append(student)
27      next_id += 1
28      return jsonify(student), 201
29
30  @app.route('/students<int:student_id>', methods=['PUT'])
31  def update_student(student_id):
32      """Update student details."""
33      for student in students:
34          if student['id'] == student_id:
35              data = request.get_json()
36              student['name'] = data.get('name', student['name'])
37              student['age'] = data.get('age', student['age'])
38              student['department'] = data.get('department', student['department'])
39              return jsonify(student)
40      return jsonify({'error': 'Student not found'}), 404
41
42  @app.route('/students<int:student_id>', methods=['DELETE'])
43  def delete_student(student_id):
44      """Delete a student."""
45      for i, student in enumerate(students):
46          if student['id'] == student_id:
47              deleted = students.pop(i)
48              return jsonify(deleted)
49      return jsonify({'error': 'Student not found'}), 404
50
51  if __name__ == "__main__":
52      app.run(debug=True)
```

Output:

```
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 528-526-087
* Detected change in 'c:\\\\Users\\\\Admin\\\\OneDrive\\\\Documents\\\\Custom Office Temp
g
* Detected change in 'c:\\\\Users\\\\Admin\\\\OneDrive\\\\Documents\\\\Custom Office Temp
g
* Detected change in 'c:\\\\Users\\\\Admin\\\\OneDrive\\\\Documents\\\\Custom Office Temp
t-15\\\\task2-15.py', reloading
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 528-526-087
127.0.0.1 - - [07/Oct/2025 12:16:45] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [07/Oct/2025 12:16:48] "GET /students HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:17:18] "POST /students HTTP/1.1" 400 -
127.0.0.1 - - [07/Oct/2025 12:17:37] "POST /students HTTP/1.1" 400 -
127.0.0.1 - - [07/Oct/2025 12:17:44] "POST /students HTTP/1.1" 400 -
127.0.0.1 - - [07/Oct/2025 12:18:27] "POST /students HTTP/1.1" 201 -
127.0.0.1 - - [07/Oct/2025 12:19:38] "PUT /students/3 HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:19:52] "GET /students HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:19:53] "GET /students HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:20:01] "POST /students HTTP/1.1" 201 -
127.0.0.1 - - [07/Oct/2025 12:20:08] "GET /students HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:21:11] "PUT /students/3 HTTP/1.1" 400 -
127.0.0.1 - - [07/Oct/2025 12:22:01] "PUT /students/3 HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:22:26] "PUT /students/3 HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:22:49] "PUT /students/3 HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:23:10] "GET /students HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:23:32] "DELETE /students/3 HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:23:46] "PUT /students/3 HTTP/1.1" 404 -
127.0.0.1 - - [07/Oct/2025 12:24:22] "POST /students HTTP/1.1" 201 -
127.0.0.1 - - [07/Oct/2025 12:24:28] "DELETE /students/2 HTTP/1.1" 200 -
```



```
C:\Users\Admin>curl -X POST http://127.0.0.1:5000/students -H "Content-Type: application/json" -d "{\"name\": \"Sahasra\", \"age\": 20, \"department\": \"ME\"}"
{
  "age": 20,
  "department": "ME",
  "id": 3,
  "name": "Sahasra"
}

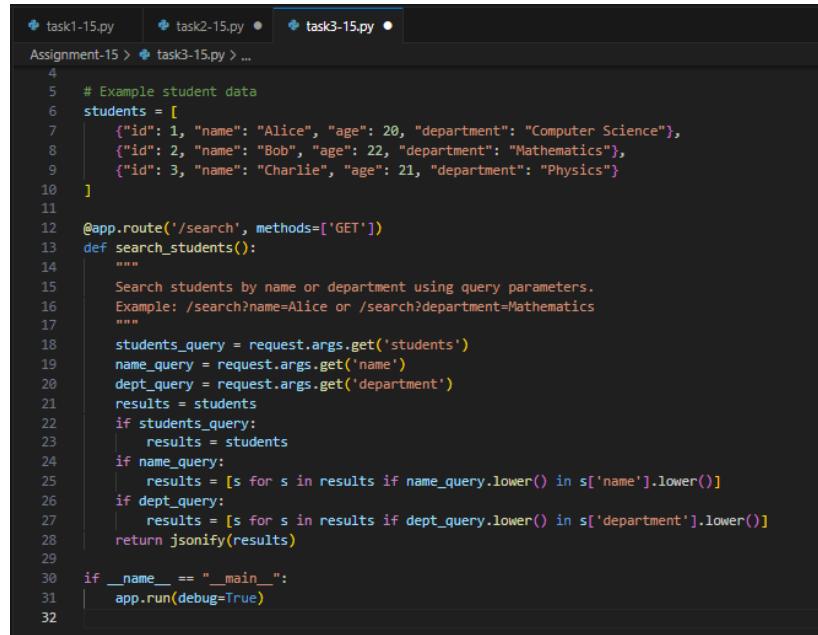
C:\Users\Admin>curl -X PUT http://localhost:5000/students/3 -H "Content-Type: application/json" -d "{\"name\": \"D. Nagamrutha\", \"age\": 20, \"department\": \"Data Science\"}"
{
  "age": 20,
  "department": "Data Science",
  "id": 3,
  "name": "D. Nagamrutha"
}
```

```
C:\Users\Admin>curl -X DELETE http://localhost:5000/students/2
{
  "age": 22,
  "department": "ECE",
  "id": 2,
  "name": "Bob"
}
```

Commands Used:

Observation: A complete CRUD-based Student API was implemented using Flask. Endpoints for GET, POST, PUT, and DELETE worked as intended. Data could be created, viewed, modified, and removed successfully. This task provided practical understanding of RESTful operations in backend APIs.

Task-3



```
task1-15.py task2-15.py task3-15.py
Assignment-15 > task3-15.py ...
4
5  # Example student data
6 students = [
7      {"id": 1, "name": "Alice", "age": 20, "department": "Computer Science"},
8      {"id": 2, "name": "Bob", "age": 22, "department": "Mathematics"},
9      {"id": 3, "name": "Charlie", "age": 21, "department": "Physics"}
10 ]
11
12 @app.route('/search', methods=['GET'])
13 def search_students():
14     """
15         Search students by name or department using query parameters.
16         Example: /search?name=Alice or /search?department=Mathematics
17     """
18     students_query = request.args.get('students')
19     name_query = request.args.get('name')
20     dept_query = request.args.get('department')
21     results = students
22     if students_query:
23         results = students
24     if name_query:
25         results = [s for s in results if name_query.lower() in s['name'].lower()]
26     if dept_query:
27         results = [s for s in results if dept_query.lower() in s['department'].lower()]
28     return jsonify(results)
29
30 if __name__ == "__main__":
31     app.run(debug=True)
```

Description: API with Query Parameters

Task: Ask AI to generate a REST API endpoint.

Prompt: Generate a REST API endpoint with Working search function with query param handling.

Code Generated:

Output:

Query all

```
C:\Users\Admin>curl "http://127.0.0.1:5000/search?name=Alice"
[{"id": 1, "name": "Alice", "age": 20, "department": "Computer Science"}]
```

to get

students:

```
C:\Users\Admin>curl "http://127.0.0.1:5000/search?department=Mathematics"
[{"id": 2, "name": "Bob", "age": 22, "department": "Mathematics"}]
```

```
* Serving Flask app 'task3-15'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 528-526-087
127.0.0.1 - - [07/Oct/2025 12:45:05] "GET /search?students HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:45:10] "GET /search?name=Alice HTTP/1.1" 200 -
127.0.0.1 - - [07/Oct/2025 12:45:27] "GET /search?department=Mathematics HTTP/1.1" 200 -
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding>
```

Query to get student by name:

```
C:\Users\Admin>curl "http://127.0.0.1:5000/search?students"
[{"id": 1, "name": "Alice", "age": 20, "department": "Computer Science"}, {"id": 2, "name": "Bob", "age": 22, "department": "Mathematics"}, {"id": 3, "name": "Charlie", "age": 21, "department": "Physics"}]
```

Query to get student by department:

```

task1-15.py task2-15.py task3-15.py task4-15.py
Assignment-15 > task4-15.py > update_student
1  from flask import Flask, jsonify, request
2
3  app = Flask(__name__)
4
5  students = [
6      {"id": 1, "name": "Alice", "age": 20, "department": "Computer Science"},
7      {"id": 2, "name": "Bob", "age": 22, "department": "Mathematics"},
8      {"id": 3, "name": "Charlie", "age": 21, "department": "Physics"}
9  ]
10 next_id = 4
11
12 # Get all students
13 @app.route('/students', methods=['GET'])
14 def get_students():
15     return jsonify(students)
16
17 # Add a student
18 @app.route('/students', methods=['POST'])
19 def add_student():
20     global next_id
21     data = request.get_json()
22     data['id'] = next_id
23     next_id += 1
24     students.append(data)
25     return jsonify(data)
26
27 # Update a student
28 @app.route('/students/<int:student_id>', methods=['PUT'])
29 def update_student(student_id):
30     data = request.get_json()
31     for student in students:
32         if student['id'] == student_id:
33             student.update(data)
34             return jsonify(student)
35     return jsonify({"error": "Student not found"}), 404
36
37 # Delete a student
38 @app.route('/students/<int:student_id>', methods=['DELETE'])
39 def delete_student(student_id):
40     for student in students:
41         if student['id'] == student_id:
42             students.remove(student)
43             return jsonify({"message": "Student deleted successfully"})
44     return jsonify({"error": "Student not found"}), 404
45
46 # Search students
47 @app.route('/search', methods=['GET'])
48 def search_students():
49     name = request.args.get('name')
50     department = request.args.get('department')
51     result = students
52
53     if name:
54         result = [s for s in result if s['name'] == name]
55     if department:
56         result = [s for s in result if s['department'] == department]
57
58     return jsonify(result)
59
60 if __name__ == '__main__':
61     app.run(debug=True)
62

```

Code Generated:

Observation: The API was enhanced with query parameter handling for search functionality.

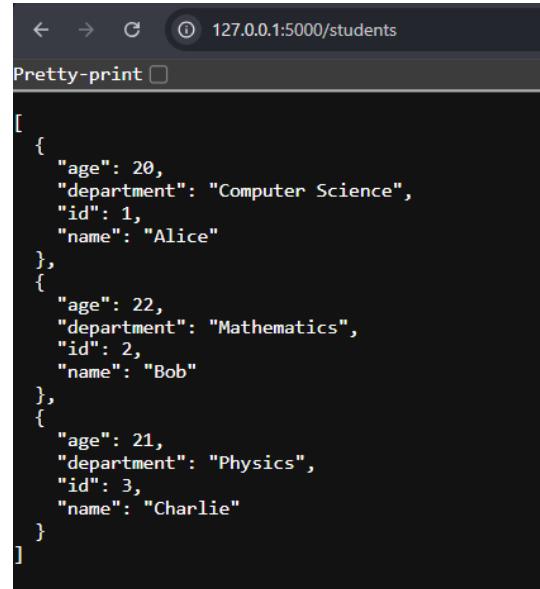
Users could retrieve students by name or department efficiently. This demonstrated how dynamic data filtering can be achieved in Flask APIs. Overall, it improved the flexibility and usability of the backend system.

Task-4 Description:— Integration & Testing
Task: Ask AI to write test scripts using Python requests module to call APIs created above.

Prompt: Write test scripts using python requests module to call APIs created before/above.

Output:

```
* Serving Flask app 'task4-15'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 528-526-087
127.0.0.1 - - [07/Oct/2025 12:57:28] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [07/Oct/2025 12:57:48] "GET /students HTTP/1.1" 200 -
```



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:5000/students`. Below the address bar, there is a "Pretty-print" checkbox which is checked. The main content area displays a JSON array with three elements, each representing a student with properties: id, name, department, and age.

```
[  
  {  
    "age": 20,  
    "department": "Computer Science",  
    "id": 1,  
    "name": "Alice"  
  },  
  {  
    "age": 22,  
    "department": "Mathematics",  
    "id": 2,  
    "name": "Bob"  
  },  
  {  
    "age": 21,  
    "department": "Physics",  
    "id": 3,  
    "name": "Charlie"  
  }]
```

Observation: Python requests module was used to test all the API endpoints. Each request returned appropriate JSON responses confirming functionality. The testing scripts validated both data integrity and route accuracy. This task reinforced API testing and integration verification skills.