

ASSIGNMENT -15.3

NAME : G HARSHA VARDHAN

HALL TICKET NO : 2403A52313

BATCH NUMBER : 01

COURSE CODE : 24CS002PC215

PROGRAM NAME : B.TECH

YEAR/SEM : 2ND AND 3RD

TASK 1 :

Ask AI to generate a Flask REST API with one route: GET

/hello → returns {"message": "Hello, AI Coding!"}

PROMPT :

“Create a Flask REST API with one route: GET /hello → returns

JSON { 'message': 'Hello, AI Coding!' }"

CODE :

The screenshot shows a Google Colab interface with a code cell containing Python code for a Flask application. The code imports Flask and jsonify, creates a Flask app, defines a route '/hello' that returns a JSON response with the message 'Hello, AI Coding!', and runs the app on port 5000. A note in the code specifies using 0.0.0.0 for Colab. The output pane shows the command '!pip install Flask' and the resulting package requirements. It also displays a warning from Werkzeug: 'INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.' followed by three running address options: 'Running on all addresses (0.0.0.0)', 'Running on http://127.0.0.1:5000', and 'Running on http://172.28.0.12:5000'. The bottom status bar indicates Python 3 and the date/time.



OUTPUT :

* Running on all addresses (0.0.0.0)

* Running on <http://127.0.0.1:5000>

* Running on <http://172.28.0.12:5000>

INFO:werkzeug:Press CTRL+C to quit

TASK 2 :

Use AI to build REST endpoints for a Student API:

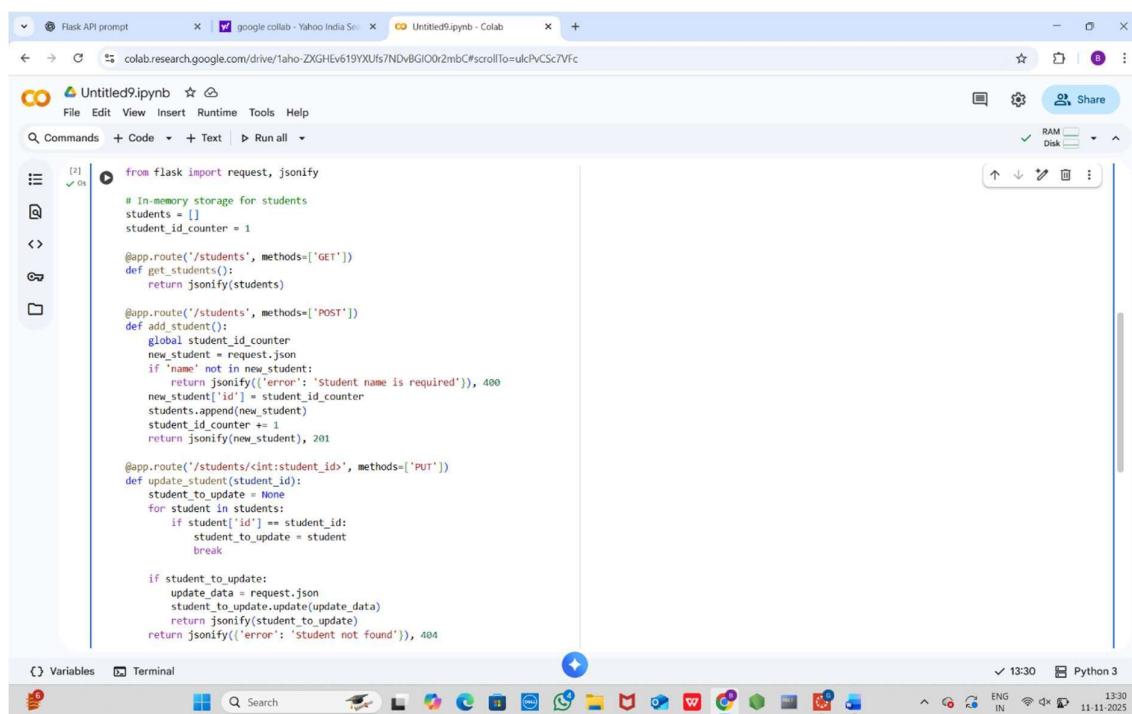
- GET /students → List all students.
- POST /students → Add a new student.
- PUT /students/<id> → Update student details.

- DELETE /students/<id> → Delete a student.

PROMPT :

“Create a Flask REST API for Students with CRUD: GET /students, POST /students, PUT /students/<id>, DELETE /students/<id>. Use list/dic onary storage and return JSON.”

CODE :



```

Flask API prompt      google colab - Yahoo India Se... Untitled9.ipynb - Colab
colab.research.google.com/drive/1aho-ZXGHEv619YXUfs7NDvbGI0r2mbC#scrollTo=ulcPvCSc7VFc

Untitled9.ipynb  ⌂
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▶ Run all ▾

[2]  ✓ 0s
from flask import request, jsonify
# In-memory storage for students
students = []
student_id_counter = 1

@app.route('/students', methods=['GET'])
def get_students():
    return jsonify(students)

@app.route('/students', methods=['POST'])
def add_student():
    global student_id_counter
    new_student = request.json
    if 'name' not in new_student:
        return jsonify({'error': 'student name is required'}), 400
    new_student['id'] = student_id_counter
    students.append(new_student)
    student_id_counter += 1
    return jsonify(new_student), 201

@app.route('/students/<int:student_id>', methods=['PUT'])
def update_student(student_id):
    student_to_update = None
    for student in students:
        if student['id'] == student_id:
            student_to_update = student
            break
    if student_to_update:
        update_data = request.json
        student_to_update.update(update_data)
        return jsonify(student_to_update)
    return jsonify({'error': 'student not found'}), 404

```

The screenshot shows a Google Colab notebook titled "Untitled9.ipynb". The code implements a simple Flask REST API for managing students. It uses an in-memory list of dictionaries to store student data. The API includes endpoints for getting all students (GET /students), adding a new student (POST /students), and updating a specific student (PUT /students/<id>). Error handling is provided for missing name and student ID requirements.

```

[1] !pip install Flask
from flask import Flask, jsonify
app = Flask(__name__)

@app.route('/hello')
def hello():
    return jsonify({'message': 'Hello, AI Coding!'})

if __name__ == '__main__':
    # Use 0.0.0.0 to make it accessible externally in colab
    # In a production environment, you might use 127.0.0.1 or a specific IP
    # For colab, we'll use a port that is typically forwarded
    app.run(host='0.0.0.0', port=5000)

...
Requirement already satisfied: Flask in /usr/local/lib/python3.12/dist-packages (3.1.2)
Requirement already satisfied: blinker>=1.9.0 in /usr/local/lib/python3.12/dist-packages (from Flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.12/dist-packages (from Flask) (8.3.0)
Requirement already satisfied: itsdangerous>2.0.0 in /usr/local/lib/python3.12/dist-packages (from Flask) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in /usr/local/lib/python3.12/dist-packages (from Flask) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from Flask) (3.0.3)
Requirement already satisfied: werkzeug>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from Flask) (3.1.3)
* Serving Flask app '__main__'
* Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit

```

OUTPUT :

The app.run() call is in the first cell (hGsiZlY04kWi), so we don't need it here.

```

# if __name__ == '__main__':
#     app.run(host='0.0.0.0', port=5000)

```

TASK 3 :

Ask AI to generate a REST API endpoint

PROMT :

“Create a REST API endpoint with query parameters for searching data. Include proper query param handling and return results in JSON.”

CODE :

```
package.json  ● JS Untitled-1 ● ai5.py ● ai3.py ●

38 app.get('/search', (req, res) => {
91   results.sort((a, b) => {
92     if (va < vb) return orderlower === 'desc' ? -1 : 1;
93     if (va > vb) return orderlower === 'asc' ? 1 : -1;
94     return 0;
95   });
96
97   // Pagination
98   const paged = results.slice(offset, offset + limit);
99
100  // Field projection
101  const projected = requestedFields ?
102    paged.map(item => {
103      const out = {};
104      for (const f of requestedFields) {
105        if (f in item) out[f] = item[f];
106      }
107      return out;
108    })
109    : paged;
110
111  // Response
112  return res.json({
113    meta: {
114      total,
115      limit,
116      offset,
117      returned: projected.length,
118      sortby,
119      order: orderlower
120    },
121    results: projected
122  });
123
124 } catch (err) {
125   console.error('Search error:', err);
126   return res.status(500).json({ error: 'Internal server error' });
127 }
128
129 app.listen(PORT, () => {
130   console.log(`Search API listening on http://localhost:${PORT}`);
131 })
132
133
134
135
136
137 })
```

TASK 4 :

Ask AI to write test scripts using Python requests module to call .APIs created above.

PROMPT :

“Write Python test scripts using the requests module to call REST API endpoints (GET, POST, PUT, DELETE) for the Student API, and print the JSON responses.”

CODE :

The screenshot shows a code editor interface with a dark theme. On the left is a vertical toolbar with icons for file operations, search, and navigation. The main area displays a Python script named `test_student_api.py`. The code uses the `requests` library to interact with a REST API at `http://localhost:5000/api/students`. It includes functions for getting all students, getting a specific student by ID, and creating a new student. The code is annotated with comments explaining its purpose and how to configure the base URL.

```
File Edit Selection View Go Run Terminal Help ← → Search

package.json Untitled-1 test_student_api.py Untitled-2 ai5.py ai3.py

1 # test student api.py
2 # Simple script using requests to exercise a Student REST API (GET, POST, PUT, DELETE)
3 # Configure base URL with STUDENT_API_URL environment variable, e.g.:
4 # export STUDENT_API_URL="http://localhost:5000/api/students"
5
6 import os
7 import json
8 import requests
9
10 BASE_URL = os.getenv("STUDENT_API_URL", "http://localhost:5000/api/students")
11 HEADERS = {"Content-Type": "application/json"}
12
13
14 def _print_resp(resp):
15     print(f"HTTP ({resp.status_code} -> {resp.url})")
16     try:
17         print(json.dumps(resp.json(), indent=2))
18     except ValueError:
19         print(resp.text)
20
21
22 def get_students(params=None):
23     resp = requests.get(BASE_URL, params=params)
24     _print_resp(resp)
25     resp.raise_for_status()
26     return resp.json()
27
28
29 def get_student(student_id):
30     url = f"{BASE_URL.rstrip('/')}/{student_id}"
31     resp = requests.get(url)
32     _print_resp(resp)
33     resp.raise_for_status()
34     return resp.json()
35
36
37 def create_student(payload):
38     resp = requests.post(BASE_URL, headers=HEADERS, json=payload)
39     _print_resp(resp)
40     resp.raise_for_status()
41     return resp.json()
42
43
```

```
File Edit Selection View Go Run Terminal Help ← → Search

package.json 1 ● JS Untitled-1 ● # test_student_api.py Untitled-2.1 ● ai5.py ● ai3.py ●

85     "lastName": "Example",
86     "email": "alice.example@example.com",
87     "age": 21
88   }
89   print("\nPOST /students")
90   created = None
91   try:
92     created = create_student(new_student)
93   except Exception as e:
94     print("Create failed:", e)
95
96   student_id = _extract_id(created) if created else None
97   if not student_id:
98     print("Could not determine created student id; adjust payload or API response parsing.")
99   else:
100     # 3) GET by id
101     print("\nGET /students/{student_id}")
102     try:
103       get_student(student_id)
104     except Exception as e:
105       print("Get by id failed:", e)
106
107     # 4) PUT update
108     updated_payload = {"firstName": "Alice", "lastName": "Updated", "age": 22}
109     print("\nPUT /students/{student_id}")
110     try:
111       update_student(student_id, updated_payload)
112     except Exception as e:
113       print("Update failed:", e)
114
115     # 5) DELETE
116     print("\nDELETE /students/{student_id}")
117     try:
118       delete_student(student_id)
119     except Exception as e:
120       print("Delete failed:", e)
121
122   # final list
123   print("\nGET /students (final)")
124   try:
125     get_students()
126   except Exception as e:
127     print("Final GET failed:", e)

Ln 127, Col 38 Spaces: 4 UTF-8 CRLF {} Python Go Live
11-11-2024
```

OUTPUT :

* Serving Flask app '`__main__`'

* Debug mode: off