AI-ASSISTED CODING

ASSIGNMENT-15.3

Backend API Development: Creating RESTful Services with AI

**NAME: Adepu Goutham Sri Bhargav**

**HALL-TICKET: 2403A510C4**

**BATCH NO: 05**

**TASK-1**:

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:**

from flask import Flask, jsonify

app = Flask(\_\_name\_\_)

@app.route('/hello', methods=['GET'])

def hello():

    return jsonify({"message": "Hello, AI Coding!"})

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**OUTPUT(terminal and website):**

**A screen shot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**OBSERVATION :**

The Flask REST API runs successfully and responds correctly when accessed at /hello. It returns a JSON message {"message": "Hello, AI Coding!"}, confirming that the route and server setup work as expected.

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:

Build a Flask REST API for managing students with the following endpoints:  
• 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:

from flask import Flask, jsonify, request

app = Flask(\_\_name\_\_)

# In-memory storage

students = [

    {"id": 1, "name": "Alice", "age": 20},

    {"id": 2, "name": "Bob", "age": 22}

]

# GET /students → List all students

@app.route('/students', methods=['GET'])

def get\_students():

    return jsonify(students), 200

# POST /students → Add a new student

@app.route('/students', methods=['POST'])

def add\_student():

    data = request.get\_json()

    new\_id = max(s["id"] for s in students) + 1 if students else 1

    new\_student = {

        "id": new\_id,

        "name": data.get("name"),

        "age": data.get("age")

    }

    students.append(new\_student)

    return jsonify({"message": "Student added successfully!", "student": new\_student}), 201

# PUT /students/<id> → Update student details

@app.route('/students/<int:student\_id>', methods=['PUT'])

def update\_student(student\_id):

    data = request.get\_json()

    for student in students:

        if student["id"] == student\_id:

            student["name"] = data.get("name", student["name"])

            student["age"] = data.get("age", student["age"])

            return jsonify({"message": "Student updated successfully!", "student": student}), 200

    return jsonify({"error": "Student not found"}), 404

# DELETE /students/<id> → Delete a student

@app.route('/students/<int:student\_id>', methods=['DELETE'])

def delete\_student(student\_id):

    for student in students:

        if student["id"] == student\_id:

            students.remove(student)

            return jsonify({"message": "Student deleted successfully!"}), 200

    return jsonify({"error": "Student not found"}), 404

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

OUTPUT:  
A screenshot of a computer program

AI-generated content may be incorrect.

OBSERVATION:

The Student API works successfully with CRUD operations using Flask. Each endpoint returns appropriate JSON responses, and the in-memory list acts as temporary storage for student data.

TASK – 3:

Ask AI to generate a REST API endpoint

PROMPT:

Generate a Flask REST API with a /search endpoint that takes a query parameter name, searches a list of students, and returns matching results in JSON format. Include code that handles when no query or no matches are found.

CODE GENERATED:

from flask import Flask, request, jsonify

app = Flask(\_\_name\_\_)

# Sample data

students = [

    {"id": 1, "name": "Alice"},

    {"id": 2, "name": "Bob"},

    {"id": 3, "name": "Charlie"},

    {"id": 4, "name": "Alicia"}

]

# GET /search?name=<query>

@app.route('/search', methods=['GET'])

def search\_students():

    query = request.args.get('name', '').lower()

    if not query:

        return jsonify({"error": "Please provide a 'name' query parameter."}), 400

    results = [s for s in students if query in s['name'].lower()]

    if not results:

        return jsonify({"message": "No matching students found."}), 404

    return jsonify({"results": results}), 200

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

from flask import Flask, request, jsonify

app = Flask(\_\_name\_\_)

# Sample data

students = [

    {"id": 1, "name": "Alice"},

    {"id": 2, "name": "Bob"},

    {"id": 3, "name": "Charlie"},

    {"id": 4, "name": "Alicia"}

]

# GET /search?name=<query>

@app.route('/search', methods=['GET'])

def search\_students():

    query = request.args.get('name', '').lower()

    if not query:

        return jsonify({"error": "Please provide a 'name' query parameter."}), 400

    results = [s for s in students if query in s['name'].lower()]

    if not results:

        return jsonify({"message": "No matching students found."}), 404

    return jsonify({"results": results}), 200

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

OBSERVATION:

The /search endpoint works correctly by taking a query parameter, filtering matching student names, and returning results in JSON format. It also handles cases with no query or no matches gracefully.

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 the Flask APIs created earlier.

CODE GENERATED:

import urllib.request

import urllib.parse

import json

BASE\_URL = "http://127.0.0.1:5000"

def print\_response(resp):

    data = resp.read().decode()

    try:

        print(json.dumps(json.loads(data), indent=2))

    except:

        print(data)

    print("-" \* 40)

# 1. GET /students

print("GET /students")

resp = urllib.request.urlopen(f"{BASE\_URL}/students")

print\_response(resp)

# 2. POST /students

print("POST /students")

student = {"name": "David", "age": 23}

data = json.dumps(student).encode("utf-8")

req = urllib.request.Request(f"{BASE\_URL}/students", data=data, headers={"Content-Type": "application/json"})

resp = urllib.request.urlopen(req)

print\_response(resp)

# 3. PUT /students/1

print("PUT /students/1")

update\_data = {"name": "Alice Updated", "age": 21}

data = json.dumps(update\_data).encode("utf-8")

req = urllib.request.Request(f"{BASE\_URL}/students/1", data=data, headers={"Content-Type": "application/json"}, method="PUT")

resp = urllib.request.urlopen(req)

print\_response(resp)

# 4. DELETE /students/2

print("DELETE /students/2")

req = urllib.request.Request(f"{BASE\_URL}/students/2", method="DELETE")

resp = urllib.request.urlopen(req)

print\_response(resp)

# 5. GET /search?name=ali

print("GET /search?name=ali")

query = urllib.parse.urlencode({"name": "ali"})

resp = urllib.request.urlopen(f"{BASE\_URL}/search?{query}")

print\_response(resp)

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

OBSERVATION:

The test script successfully calls all Flask API endpoints using Python’s built-in modules. It handles GET, POST, PUT, DELETE, and query parameters, returning clear JSON responses without requiring any external libraries.