AI Assisted Coding – Lab 15.4

NAME:- B.ShivaShankar

HT NO:-2403A51252

BATCH:-11

# Task 1 – Setup Flask Backend

## **Aim: Create a basic Flask server with a single welcome endpoint.**

## **CODE:-**

from flask import Flask, jsonify app = Flask( name )

@app.route('/') def home():

return jsonify({"message": "Welcome to AI-assisted API"})

if name == " main ": app.run(debug=True)

**Output:**

{"message": "Welcome to AI-assisted API"}

# Task 2 – Create a CRUD API (Read and Create)

## **Aim: Implement endpoints to list and add items.**

## **CODE:-**

from flask import Flask, jsonify, request app = Flask( name )

items = []

@app.route('/items', methods=['GET']) def get\_items():

return jsonify(items)

@app.route('/items', methods=['POST']) def add\_item():

data = request.get\_json() items.append(data)

return jsonify({"message": "Item added", "item": data}), 201

**Output:**

GET /items  []

POST /items {"name":"Book","price":200}  {"message": "Item added","item":{"name":"Book","price":200 GET /items  [{"name":"Book","price":200}]

# Task 3 – Update Item

## **Aim: Implement PUT endpoint to update an existing item.**

## **CODE:-**

@app.route('/items/<int:index>', methods=['PUT']) def update\_item(index):

if index < 0 or index >= len(items):

return jsonify({"error": "Item not found"}), 404 data = request.get\_json()

items[index] = data

return jsonify({"message": "Item updated", "item": data})

**Output:**

PUT /items/0 {"name":"Notebook","price":250}  {"message":"Item updated","item":{"name":"Notebook","

# Task 4 – Delete Item

## **Aim:** Implement DELETE endpoint to remove an item.

## **CODE:-**

@app.route('/items/<int:index>', methods=['DELETE']) def delete\_item(index):

if index < 0 or index >= len(items):

return jsonify({"error": "Item not found"}), 404 removed\_item = items.pop(index)

return jsonify({"message": "Item deleted", "item": removed\_item})

**Output:**

DELETE /items/0  {"message":"Item deleted","item":{"name":"Notebook","price":250}} GET /items  []

# Task 5 – Add Auto-Generated Documentation

## **Aim: Add inline docstrings and Swagger integration for API documentation.**

## **CODE:-**

@app.route('/items', methods=['GET']) def get\_items():

"""

GET /items

Returns a list of all items in the store. """

return jsonify(items)

# Swagger setup (optional)

# from flask\_restx import Api

# api = Api(app, doc='/docs', title='AI-Assisted API', description='CRUD Operations Demo')

**Output:-**

API documentation visible at /docs (if Flask-RESTX is installed). Inline docstrings available for all endpoints.