AI Assisted Coding – Lab 15.4

NAME:- B.yashwanth
HT NO:-2403A51266
BATCH:-12

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.
```