ASSIGNMENT – 15.4

NAME: G. Bala Varshitha Reddy

HT.NO: 2403A52050

BATCH NO: AIB03

TASK 1:

PROMPT:

Generate a simple Flask backend with one endpoint that returns a JSON message “Welcome to AI-assisted API”.

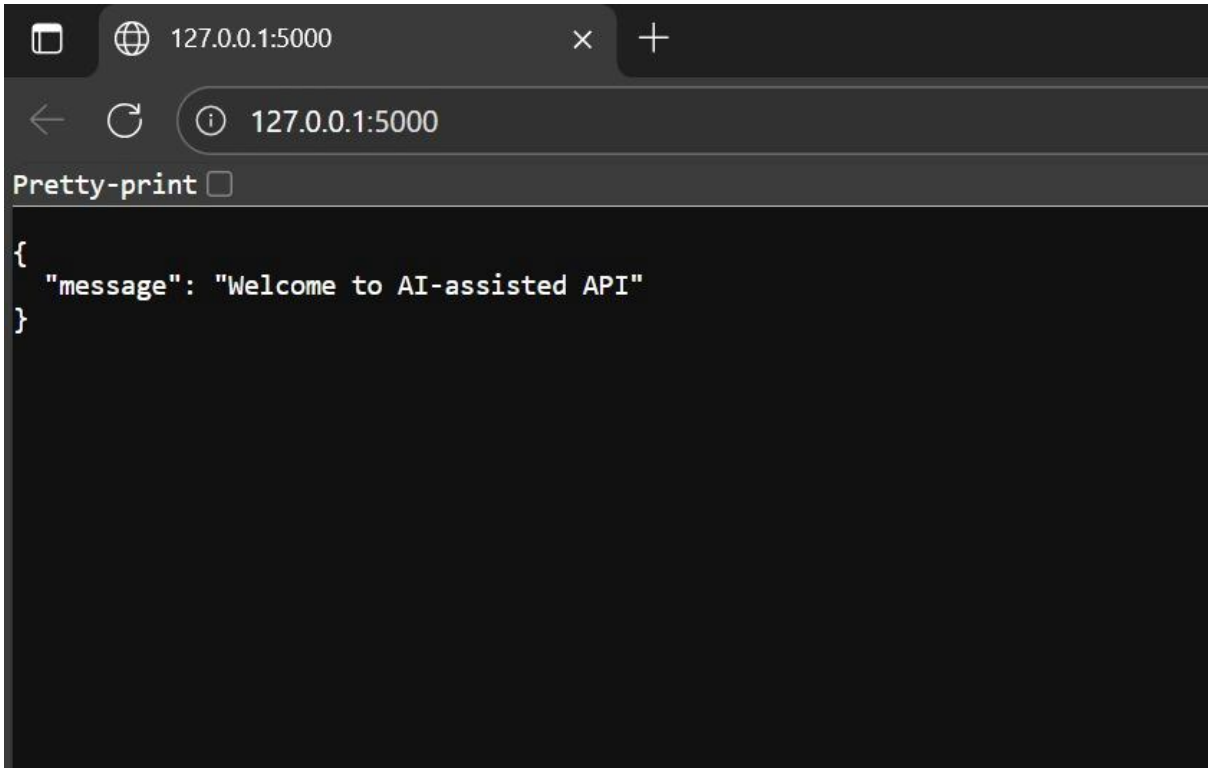
CODE:

lab-15 > task1.py > ...

```python
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:

```
in\OneDrive\Desktop\AI Assisted coding assignments\AI Assisted Coding\lab-15\task1.py'
 * Serving Flask app 'task1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server in
stead.
 * Running on http://127.0.0.1:5000
 * Running on http://127.0.0.1:5000   arting with stat
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
```



OBSERVATION:

In python we used flask library. Code was simple and short. In normal python code we use print but here it used jsonif. When we run the code. In terminal it is showing a link in which the text is deployed. When we click on that we can see the message.

TASK 2:

PROMPT:

Create a simple Flask CRUD API with Read (GET) and Create (POST) operations using an in-memory list.

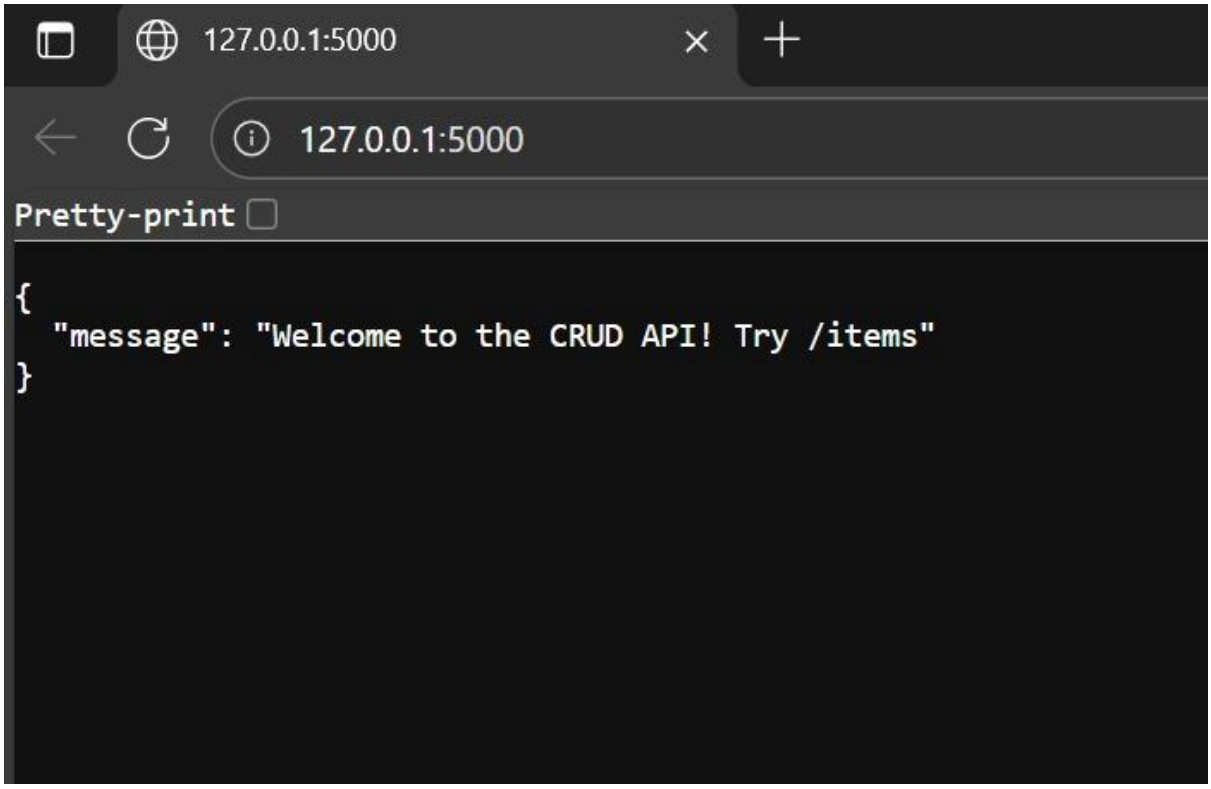
CODE:

lab-15 > task2.py > ...

```python
from flask import Flask, jsonify, request
app = Flask(__name__)
items = []
@app.route('/')
def home():
    return jsonify({"message": "Welcome to the CRUD API! Try /items"}), 200
# GET all items
@app.route('/items', methods=['GET'])
def get_items():
    return jsonify(items), 200
# POST a new item
@app.route('/items', methods=['POST'])
def add_item():
    data = request.get_json()
    if not data or "name" not in data:
        return jsonify({"error": "Invalid item data"}), 400
    item = {
        "id": len(items) + 1,
        "name": data["name"]
    }
    items.append(item)
    return jsonify({"message": "Item added", "item": item}), 201
if __name__ == "__main__":
    app.run(debug=True)

```

OUTPUT:

```
 * Serving Flask app 'task2'
 * 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 stat
 * Debugger is active!
 * Debugger PIN: 747-951-939
127.0.0.1 - - [23/Oct/2025 12:24:11] "GET / HTTP/1.1" 200 -
```



OBSERVATION:

Here it used Flask to create two API endpoints for reading and adding items. The /items route with the GET method returns all items stored in the in-memory list, while the POST method adds a new item received as JSON data. The added item is appended to the list and a success message is returned. Since the data is stored in memory, it will reset when the server restarts. This provides a simple demonstration of basic CRUD operations.

TASK 3:

PROMPT:

Create a PUT endpoint in Flask to update an existing item by its index.

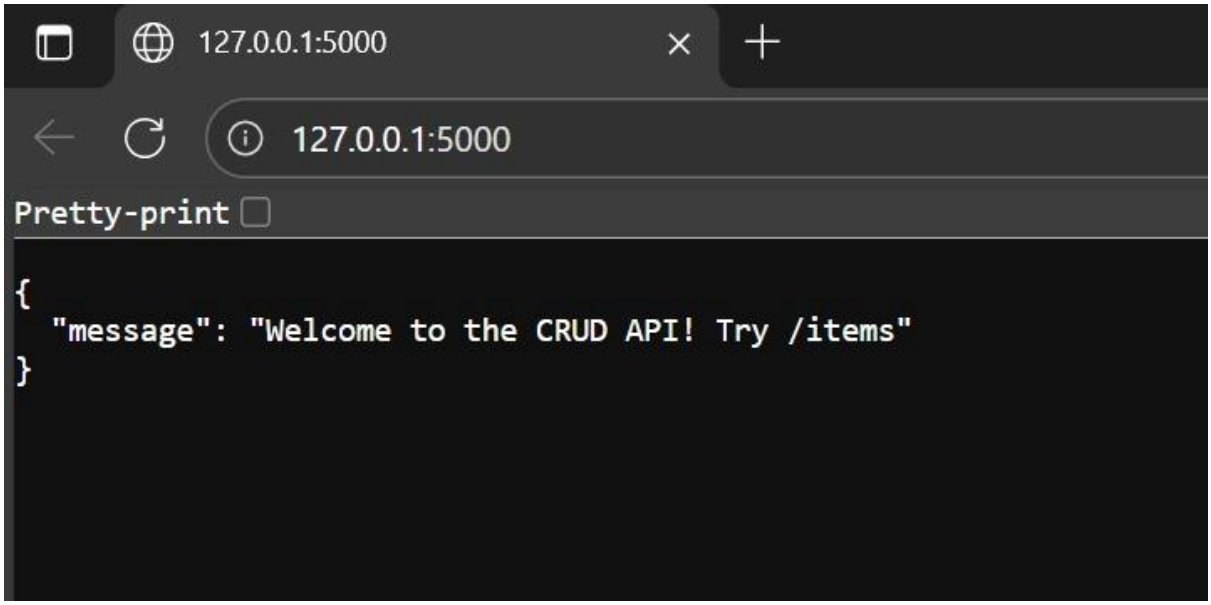
CODE:

lab-15 > task3.py > ...

```python
from flask import Flask, jsonify, request
app = Flask(__name__)
# In-memory list
items = []
# Home route
@app.route('/')
def home():
    return jsonify({"message": "Welcome to the CRUD API! Try /items"}), 200
# GET all items
@app.route('/items', methods=['GET'])
def get_items():
    return jsonify(items), 200
# POST a new item
@app.route('/items', methods=['POST'])
def add_item():
    data = request.get_json()
    if not data:
        return jsonify({"error": "Invalid item data"}), 400
    items.append(data)
    return jsonify({"message": "Item added", "item": data}), 201
# PUT /items/<int:index> - update an existing item
@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()
    if not data:
        return jsonify({"error": "Invalid item data"}), 400
    items[index] = data
    return jsonify({"message": "Item updated", "item": data}), 200
if __name__ == "__main__":
    app.run(debug=True)
```

OUTPUT:

```
 * Serving Flask app 'task3'
 * 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 stat
 * Debugger is active!
 * Debugger PIN: 747-951-939
127.0.0.1 - - [23/Oct/2025 12:30:05] "GET / HTTP/1.1" 200 -
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 747-951-939
127.0.0.1 - - [23/Oct/2025 12:30:05] "GET / HTTP/1.1" 200 -
```



OBSERVATION:

This Flask code defines a PUT endpoint allows updating an existing item in the list based on its index. It first checks if the given index is valid to avoid errors when accessing non-existent items. The new data for the update is received as a JSON payload from the client request and replaces the old item in the list. Finally, the API returns a JSON response containing a success message along with the updated item details, confirming that the update was successful.