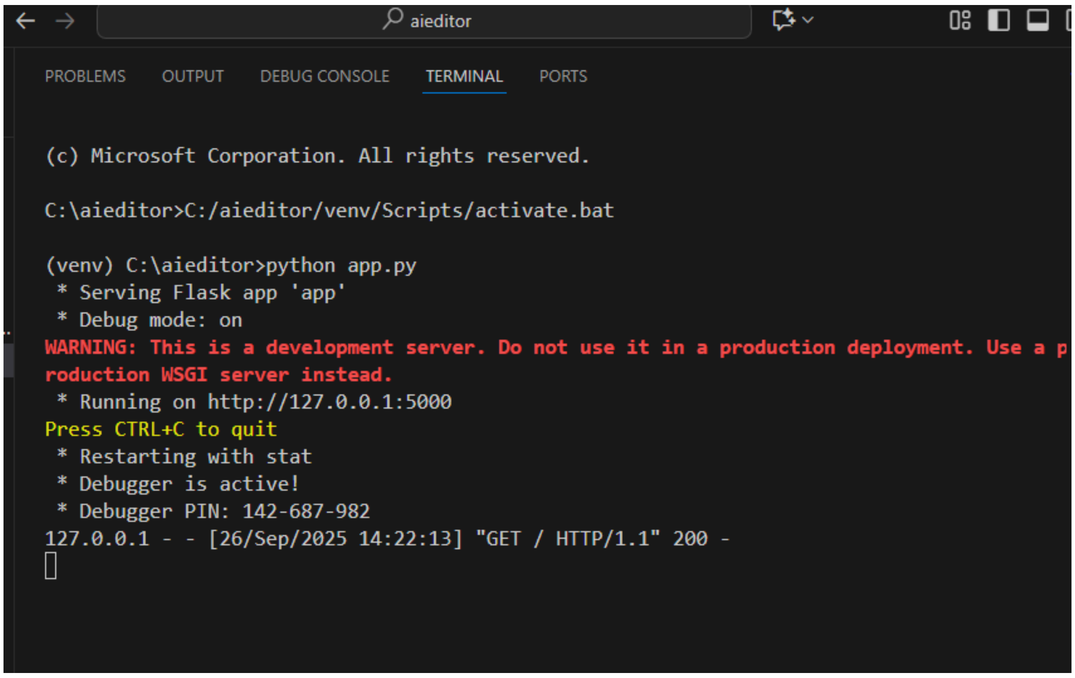
Flask REST API with Code and Output Screenshots

Terminal Output:

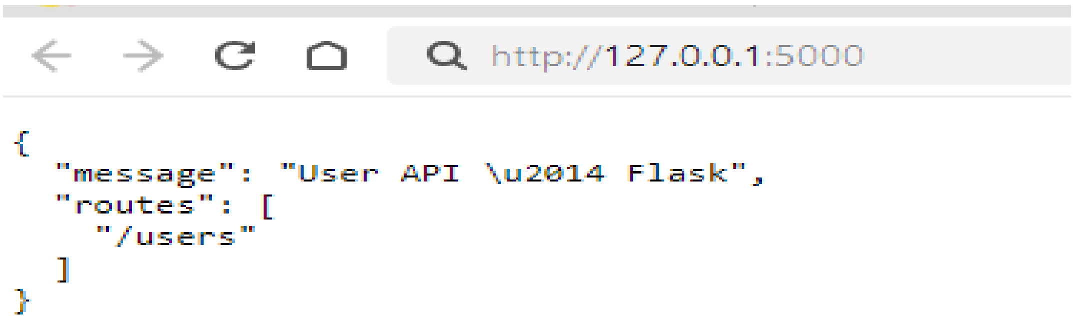


```
(c) Microsoft Corporation. All rights reserved.

C:\aieditor>C:/aieditor/venv/Scripts/activate.bat

(venv) C:\aieditor>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a p
roduction WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 142-687-982
127.0.0.1 - - [26/Sep/2025 14:22:13] "GET / HTTP/1.1" 200 -
```

Browser Output:



```
{
  "message": "User API \u2014 Flask",
  "routes": [
    "/users"
  ]
}
```

Flask API Source Code:

```
from flask import Flask, request, jsonify, make_response, url_for
```

```python
app = Flask(__name__)

# In-memory store (simple)
users = []
_next_id = 1

def get_next_id():
    global _next_id
    nid = _next_id
    _next_id += 1
    return nid

def find_user(user_id):
    return next((u for u in users if u["id"] == user_id), None)

# Root
@app.route("/", methods=["GET"])
def index():
    return jsonify({"message": "User API — Flask", "routes": ["/users"]})

# List users
@app.route("/users", methods=["GET"])
def list_users():
    return jsonify(users), 200

# Get single user
@app.route("/users/<int:user_id>", methods=["GET"])
def get_user(user_id):
    user = find_user(user_id)
    if not user:
        return jsonify({"error": "User not found"}), 404
    return jsonify(user), 200

# Create user
@app.route("/users", methods=["POST"])
def create_user():
    if not request.is_json:
        return jsonify({"error": "Request body must be JSON"}), 400

    data = request.get_json()
    # Basic validation
    if not data.get("name") or not data.get("email"):
        return jsonify({"error": "Both 'name' and 'email' are required"}), 400

    user = {
        "id": get_next_id(),
        "name": data["name"],
        "email": data["email"],
        # extra optional fields
        "age": data.get("age"),
        "bio": data.get("bio")
    }
    users.append(user)
    # Location header with new resource url
    location = url_for("get_user", user_id=user["id"], _external=False)
    return make_response(jsonify(user), 201, {"Location": location})

# Update user (replace/modify)
@app.route("/users/<int:user_id>", methods=["PUT"])
def update_user(user_id):
    user = find_user(user_id)
    if not user:
        return jsonify({"error": "User not found"}), 404

    if not request.is_json:
        return jsonify({"error": "Request body must be JSON"}), 400

    data = request.get_json()
    # Validate at least one field present
    if not any(k in data for k in ("name", "email", "age", "bio")):
        return jsonify({"error": "No valid fields to update"}), 400

    # Update allowed fields
    for field in ("name", "email", "age", "bio"):
        if field in data:
            user[field] = data[field]
    return jsonify(user), 200
```

```
# Delete user
@app.route("/users/<int:user_id>", methods=["DELETE"])
def delete_user(user_id):
    user = find_user(user_id)
    if not user:
        return jsonify({"error": "User not found"}), 404
    users.remove(user)
    # 204 No Content (successful delete)
    return "", 204

# Simple error handler example
@app.errorhandler(404)
def not_found(e):
    return jsonify({"error": "Not found"}), 404

if __name__ == "__main__":
    # Run with: python app.py
    app.run(debug=True, host="127.0.0.1", port=5000)
```