**Lab 15 – Backend API Development: Creating RESTful services with AI**

**Task 1: Setup Flask Backend**

**Instructions:**

- Install Flask and set up a basic Python server.
- Use AI to generate starter code for a simple backend with a single endpoint.

**Prompt:** Generate a basic Flask backend with one endpoint that returns a welcome message in JSON.

**Expected Output:**

- Running the server should show:

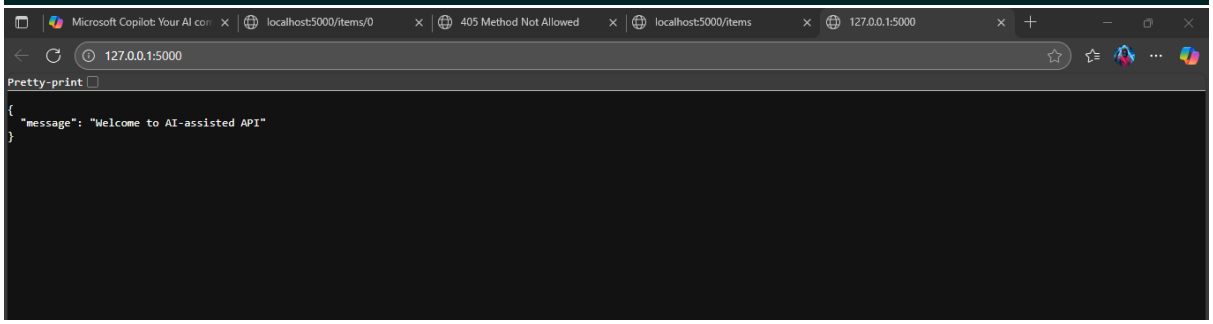
http://127.0.0.1:5000/

- Accessing it in the browser or Postman returns:

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

HTML files > ai_09 > basic.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)
```




**Task 2: Create a CRUD API – Read and Create**

**Instructions:**

- Use AI to implement endpoints to **list all items** and **add a new item**.

Use an in-memory Python list to store items

**Prompt:** Create a Flask API with two endpoints: one to list all items (GET /items) and one to add a new item (POST /items). Use an in-memory Python list to store items. Return JSON responses.

**Expected Output:**

- **GET /items** → [] initially
- **POST /items** with payload {"name":"Book","price":200} →

{"message": "Item added", "item": {"name":"Book","price":200}}

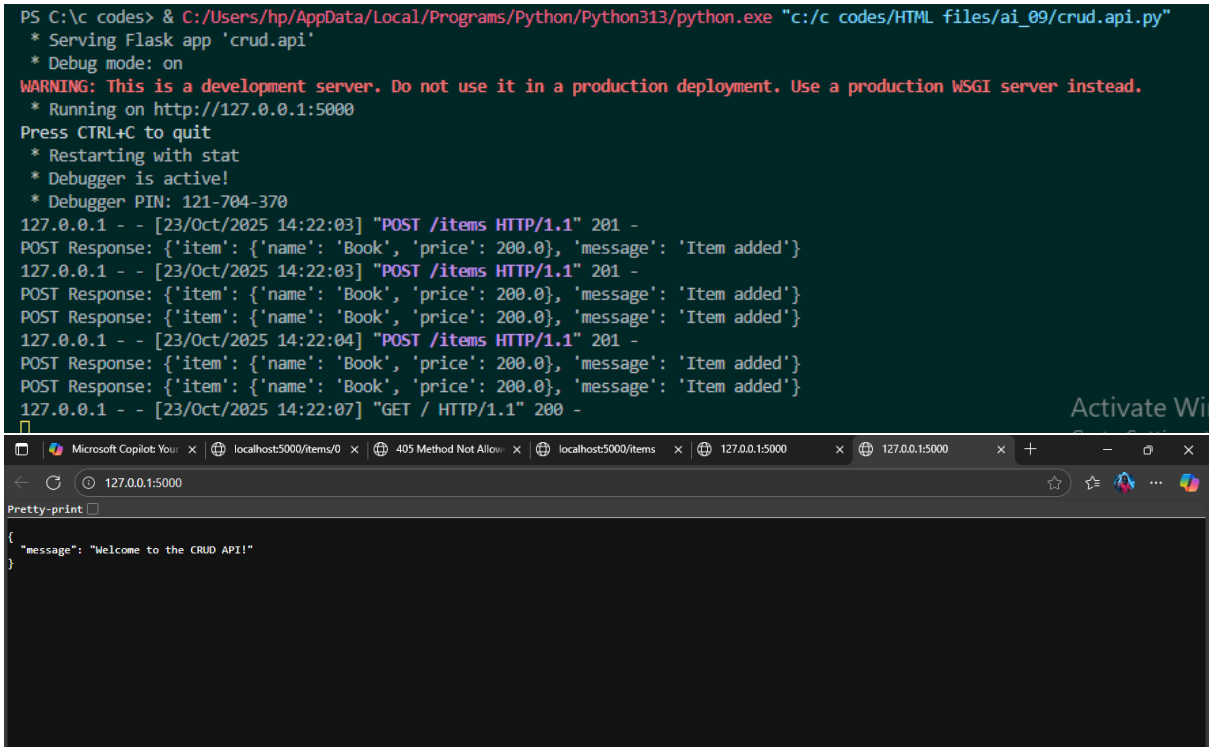
- **GET /items** now returns:

[{"name":"Book","price":200}]

HTML files > ai_09 > crud.api.py > ...

```python
from flask import Flask, jsonify, request
import threading
import time
import requests

app = Flask(__name__)

# In-memory list to store items
items = []

# GET /items → List all items
@app.route('/items', methods=['GET'])
def get_items():
    return jsonify(items), 200

# POST /items → Add a new item
@app.route('/items', methods=['POST'])
def add_item():
    data = request.get_json()

    # Validate input
    if not data or 'name' not in data or 'price' not in data:
        return jsonify({"error": "Missing 'name' or 'price'"}), 400

    try:
        data['price'] = float(data['price'])
    except (ValueError, TypeError):
        return jsonify({"error": "'price' must be a number"}), 400
```

```python
def add_item():
        return jsonify({"error": "'price' must be a number"}), 400

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

# Optional root route
@app.route('/')
def index():
    return jsonify({"message": "Welcome to the CRUD API!"}), 200

# Function to test POST request after server starts
def test_post():
    time.sleep(1)  # Wait for server to start
    url = "http://127.0.0.1:5000/items"
    data = {"name": "Book", "price": 200}
    response = requests.post(url, json=data)
    print("POST Response:", response.json())

if __name__ == "__main__":
    threading.Thread(target=test_post).start()
    app.run(debug=True)
```




## Task 3: Update Item

**Instructions:**

- Use AI to create a **PUT endpoint** to update an existing item based on its index or ID.

**Prompt:** Build a Flask API with two endpoints: • `GET /items` to return all items from an in-memory list • `POST /items` to add a new item to the list using JSON payload Return responses in JSON format.

**Expected Output:**

- **PUT /items/0** with payload {"name":"Notebook","price":250} → {"message": "Item updated", "item": {"name":"Notebook","price":250}}

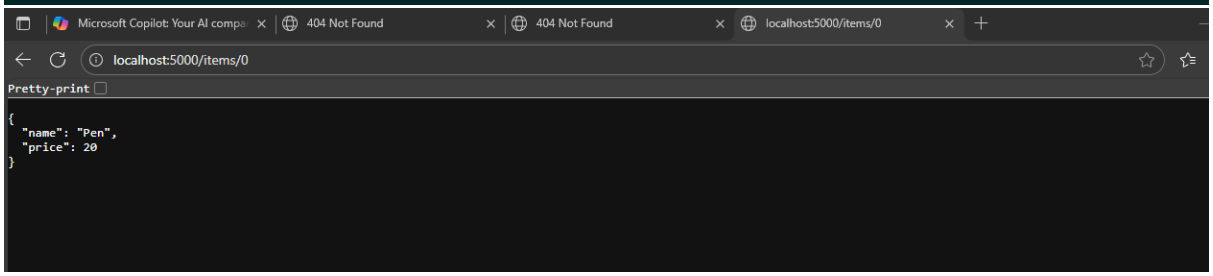
HTML files > ai_09 > 3_ai.py > ...

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

# Sample items list
items = [
    {"name": "Pen", "price": 20},
    {"name": "Book", "price": 100}
]

# PUT /items/<int:index> → Update 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 or not isinstance(data, dict):
        return jsonify({"error": "Invalid input"}), 400

    items[index].update(data)
    return jsonify({"message": "Item updated", "item": items[index]})

# GET /items/<int:index> → View item (for browser testing)
@app.route('/items/<int:index>', methods=['GET'])
def get_item(index):
    if index < 0 or index >= len(items):
        return jsonify({"error": "Item not found"}), 404
```


```python
@app.route('/items/<int:index>', methods=['GET'])
def get_item(index):
    if index < 0 or index >= len(items):
        return jsonify({"error": "Item not found"}), 404
```


```python
    return jsonify(items[index])

if __name__ == '__main__':
    app.run(debug=True)
```

```
PS C:\c codes> & C:/Users/hp/AppData/Local/Programs/Python/Python313/python.exe "c:/c codes/HTML files/ai_09/3_ai.py"
 * Serving Flask app '3_ai'
 * 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: 121-704-370
127.0.0.1 - - [23/Oct/2025 14:24:52] "GET / HTTP/1.1" 404 -
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: 121-704-370
127.0.0.1 - - [23/Oct/2025 14:24:52] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 14:25:16] "GET /items HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 14:26:07] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 14:26:19] "GET /items/0 HTTP/1.1" 200 -
```




**Task 4: Delete Item**

**Instructions:**

- Use AI to create a **DELETE endpoint** to remove an item by index or ID.

**Prompt :** Add a DELETE endpoint to a Flask API that removes an item from an in-memory list by index. If the index is invalid, return a 404 error. Respond with JSON.

**Expected Output:**

- **DELETE /items/0 →**

{"message": "Item deleted", "item": {"name":"Notebook","price":250}}

- **GET /items →** []

HTML files > ai_09 > 4_AI.py > ...

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

# Sample items list
items = [
    {"name": "Notebook", "price": 250}
]

# DELETE /items/<int:index> → Delete item by index
@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})

# GET /items → View all items
@app.route('/items', methods=['GET'])
def get_items():
    return jsonify(items)

if __name__ == '__main__':
    app.run(debug=True)
```

```
PS C:\c codes> & C:/Users/hp/AppData/Local/Programs/Python/Python313/python.exe "c:/c codes/HTML files/ai_09/4_AI.py"
 * Serving Flask app '4_AI'
 * 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!
 * Restarting with stat
 * Debugger is active!
 * Debugger is active!
 * Debugger PIN: 121-704-370
127.0.0.1 - - [23/Oct/2025 14:17:05] "DELETE /items/0 HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2025 14:19:07] "DELETE /items/0 HTTP/1.1" 404 -
```

```
PS C:\c codes> curl.exe -X DELETE http://localhost:5000/items/0
>>
{
  "item": {
    "name": "Notebook",
    "price": 250
  },
    "price": 250
  },
  },
  "message": "Item deleted"
}
PS C:\c codes> curl.exe -X DELETE http://localhost:5000/items/0
{
  "error": "Item not found"
}
PS C:\c codes>
```

**Task 5: Add Auto-Generated Documentation**

**Instructions:**

- Use AI to add inline comments and docstrings for all endpoints.
- Optionally, integrate Swagger / Flask-RESTX to auto-generate API documentation.

Starter Comment Example:

```
@app.route('/items', methods=['GET'])
def get_items():
    """
    GET /items
    Returns a list of all items in the store.
    """
    return jsonify(items)
```

**Expected Output:**

- Clear documentation for all endpoints, e.g., via /docs if using Swagger.
- Students should be able to see endpoint description, methods, and sample payloads.

**Prompt :**

Add inline comments and docstrings to all endpoints in a Flask API that manages an in-memory list of items. Each endpoint should include a clear description, HTTP method, and expected input/output. Optionally, integrate Swagger or Flask-RESTX to auto-generate API documentation accessible via `/docs`.

HTML files > ai_09 > 5_ai.py > ...

```python
from flask import Flask, request
from flask_restx import Api, Resource, fields

# Initialize Flask and Flask-RESTX
app = Flask(__name__)
api = Api(
    app,
    version='1.0',
    title='Item Store API',
    description='Manage items with CRUD operations',
    doc='/docs'  # Swagger UI available at /docs
)

# In-memory item list
items = [
    {"name": "Pen", "price": 20},
    {"name": "Book", "price": 100}
]

# Define the item model for Swagger documentation
item_model = api.model('Item', {
    'name': fields.String(required=True, description='Name of the item', example='Notebook'),
    'price': fields.Integer(required=True, description='Price of the item', example=250)
})

@api.route('/items')
class ItemList(Resource):
    def get(self):
```

HTML files > ai_09 > 5_ai.py > ...

```python
class ItemList(Resource):
    def get(self):
        """
        GET /items
        Returns a list of all items.
        """
        return items

    @api.expect(item_model)
    def post(self):
        """
        POST /items
        Adds a new item to the list.
        Payload: { "name": "Notebook", "price": 250 }
        """
        data = request.json
        items.append(data)
        return {"message": "Item added", "item": data}, 201

@api.route('/items/<int:index>')
@api.doc(params={'index': 'Index of the item in the list'})
class Item(Resource):
    def get(self, index):
        """
        GET /items/<index>
        Returns a single item by index.
        """
```

HTML files > ai_09 > 5_ai.py > ItemList > get

```python
class Item(Resource):
    def get(self, index):
        if index < 0 or index >= len(items):
            api.abort(404, "Item not found")
        return items[index]

    @api.expect(item_model)
    def put(self, index):
        """
        PUT /items/<index>
        Updates an existing item by index.
        Payload: { "name": "Notebook", "price": 250 }
        """
        if index < 0 or index >= len(items):
            api.abort(404, "Item not found")
        data = request.json
        items[index].update(data)
        return {"message": "Item updated", "item": items[index]}

    def delete(self, index):
        """
        DELETE /items/<index>
        Deletes an item by index.
        """
        if index < 0 or index >= len(items):
            api.abort(404, "Item not found")
        removed_item = items.pop(index)
        return {"message": "Item deleted", "item": removed item}
```

```python
        DELETE /items/<index>
        Deletes an item by index.
        """
        if index < 0 or index >= len(items):
            api.abort(404, "Item not found")
        removed_item = items.pop(index)
        return {"message": "Item deleted", "item": removed_item}

if __name__ == '__main__':
    app.run(debug=True)
```

```
PS C:\c codes> & C:/Users/hp/AppData/Local/Programs/Python/Python313/python.exe "c:/c codes/HTML files/ai_09/5_ai.py"
 * Serving Flask app '5_ai'
 * 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: 121-704-370
127.0.0.1 - - [23/Oct/2025 15:10:20] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 15:10:20] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 15:10:36] "GET /doc HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 15:10:39] "GET /doc HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 15:11:05] "GET /docs HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2025 15:11:05] "GET /swaggerui/swagger-ui-standalone-preset.js HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2025 15:11:05] "GET /swaggerui/droid-sans.css HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2025 15:11:05] "GET /swaggerui/swagger-ui.css HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2025 15:11:05] "GET /swaggerui/swagger-ui-bundle.js HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2025 15:11:06] "GET /swagger.json HTTP/1.1" 200 -
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 121-704-370
127.0.0.1 - - [23/Oct/2025 15:10:20] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 15:10:20] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 15:10:36] "GET /doc HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 15:10:39] "GET /doc HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2025 15:11:05] "GET /docs HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2025 15:11:05] "GET /swaggerui/swagger-ui-standalone-preset.js HTTP/1.1" 200 -
```