AI ASSIGNMENT:15

HTNO:2403A51284

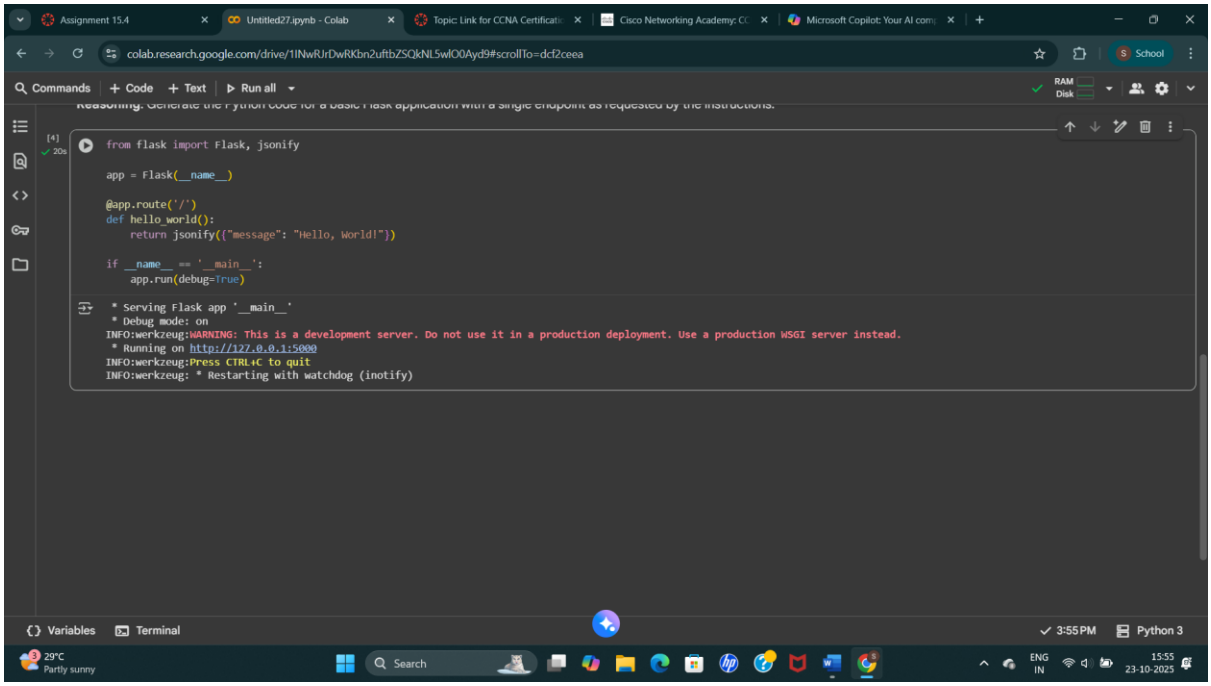
BATCH:12

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.
Legacy/Starter Code:

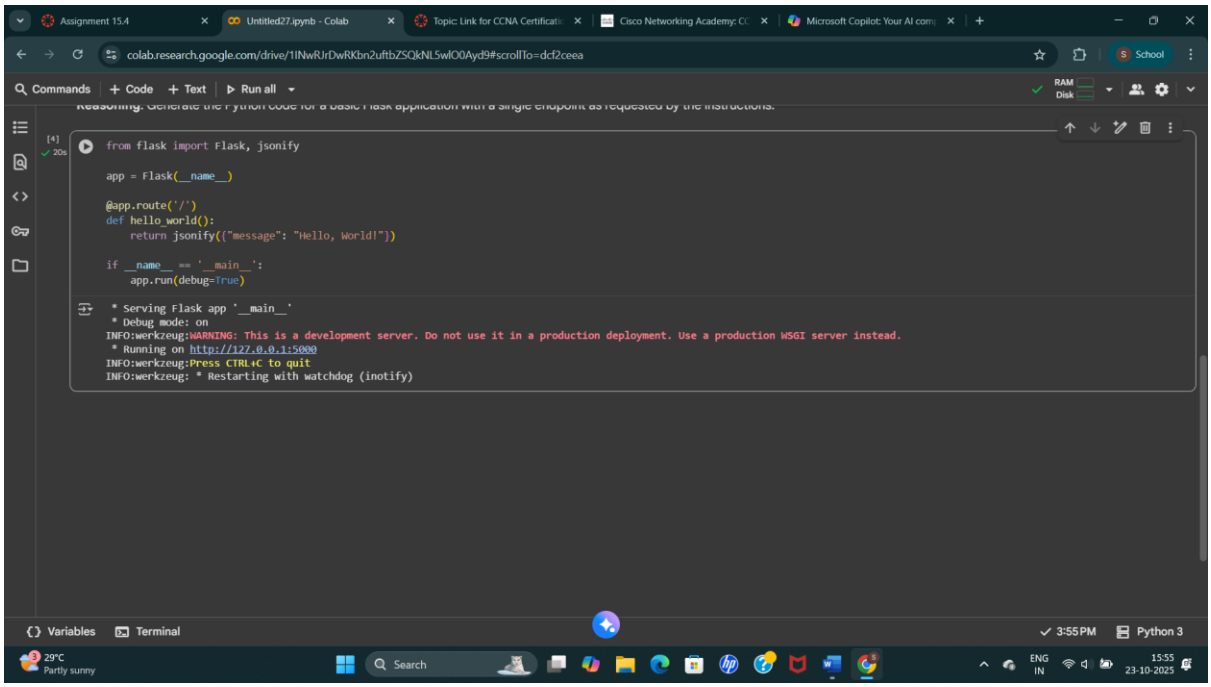
```
from flask import Flask, jsonify
app = Flask(__name__)
@app.route('/')
def home():
return jsonify({"message": "Welcome to AI-assisted API"})
if __name__ == "__main__":
app.run(debug=True)
```

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 coding}

CODE:



OUTPUT:



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.

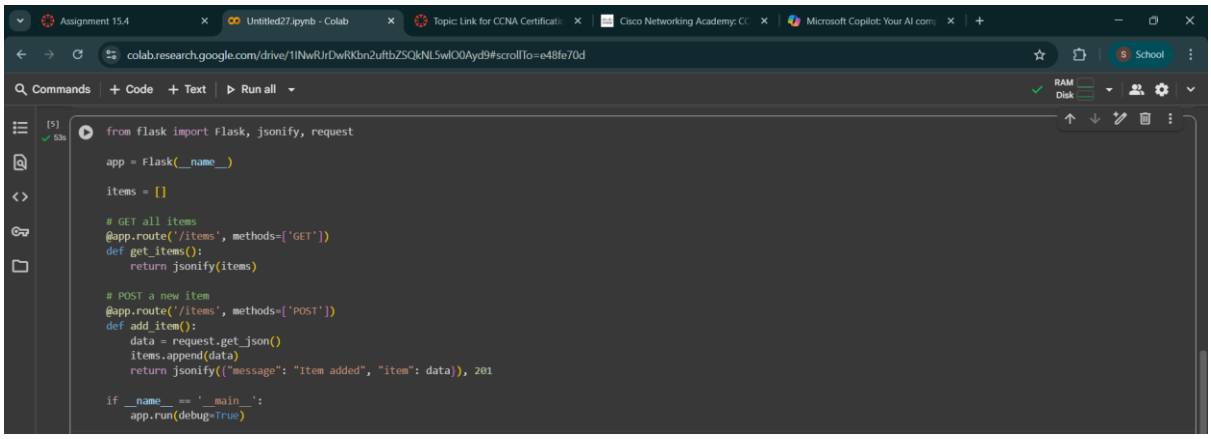
Starter Code:

from flask import Flask, jsonify, request

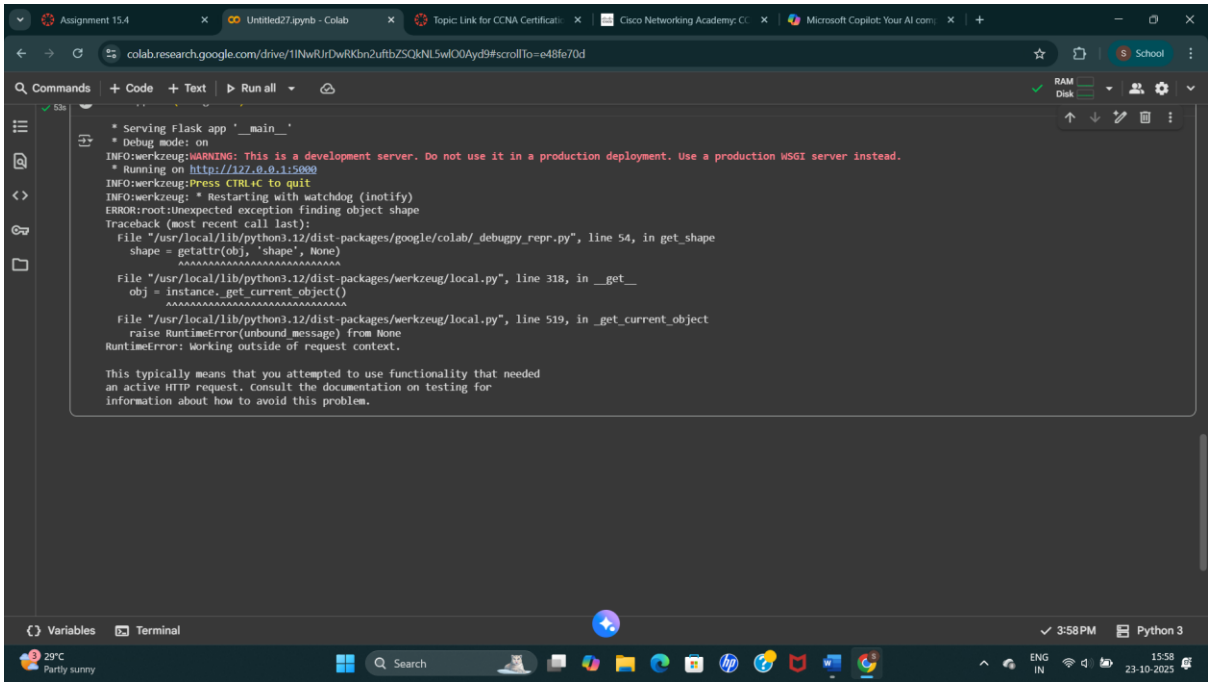
app = Flask(__name__)
items = []
# GET all items
@app.route('/items', methods=['GET'])
def get_items():
return jsonify(items)
POST a new item
@app.route('/items', methods=['POST'])
def add_item():
data = request.get_json()
items.append(data)

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

CODE:



OUTPUT:



Task 3: Update Item

Instructions:

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

Starter Code:

```
# PUT /items/<int:index>
@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})
```

Expected Output:

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

CODE:

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

app = Flask(__name__)

items = []

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

# POST a new item
@app.route('/items', methods=['POST'])
def add_item():
    data = request.get_json()
    items.append(data)
    return jsonify({"message": "item added", "item": data}), 201

# PUT /items/<int:index>
@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})

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

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
```

OUTPUT:

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug: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
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with watchdog (inotify)
ERROR:root:Unexpected exception finding object shape
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/google/colab/_debugpy_repr.py", line 54, in get_shape
    shape = getattr(obj, 'shape', None)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 318, in __get__
    obj = instance._get_current_object()
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 519, in _get_current_object
    raise RuntimeError(unbound_message) from None
RuntimeError: Working outside of request context.

This typically means that you attempted to use functionality that needed
an active HTTP request. Consult the documentation on testing for
information about how to avoid this problem.
```

Task 4: Delete Item

Instructions:

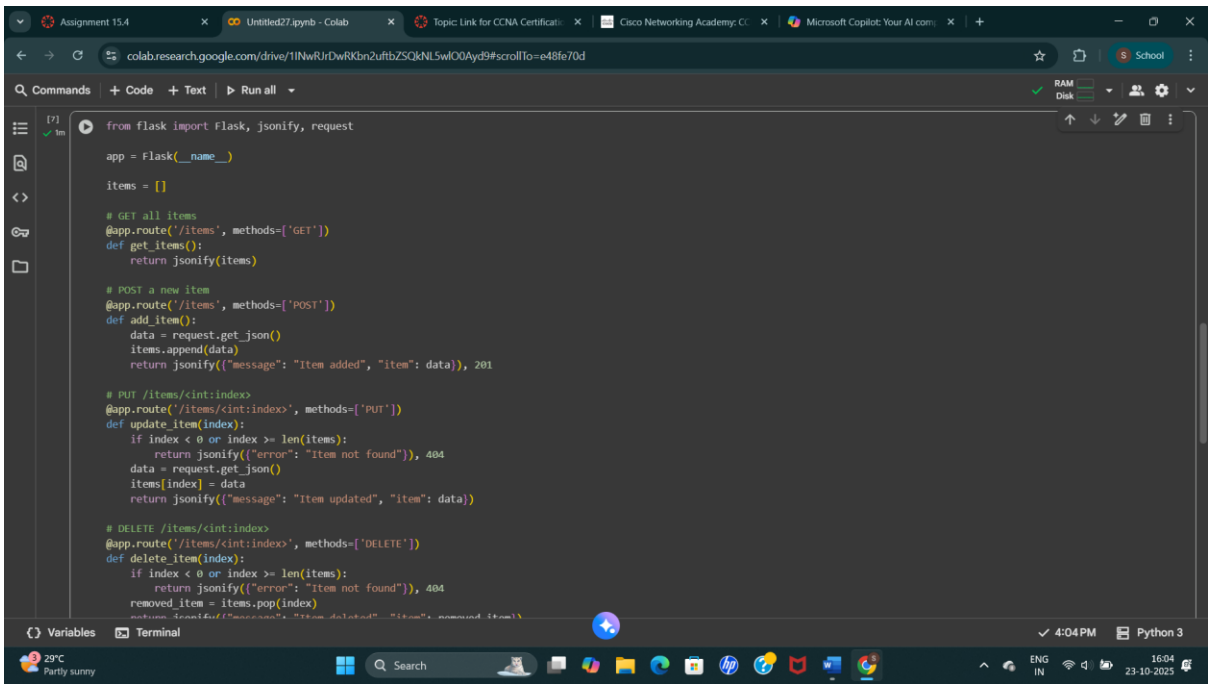
• Use AI to create a DELETE endpoint to remove an item by index or ID.

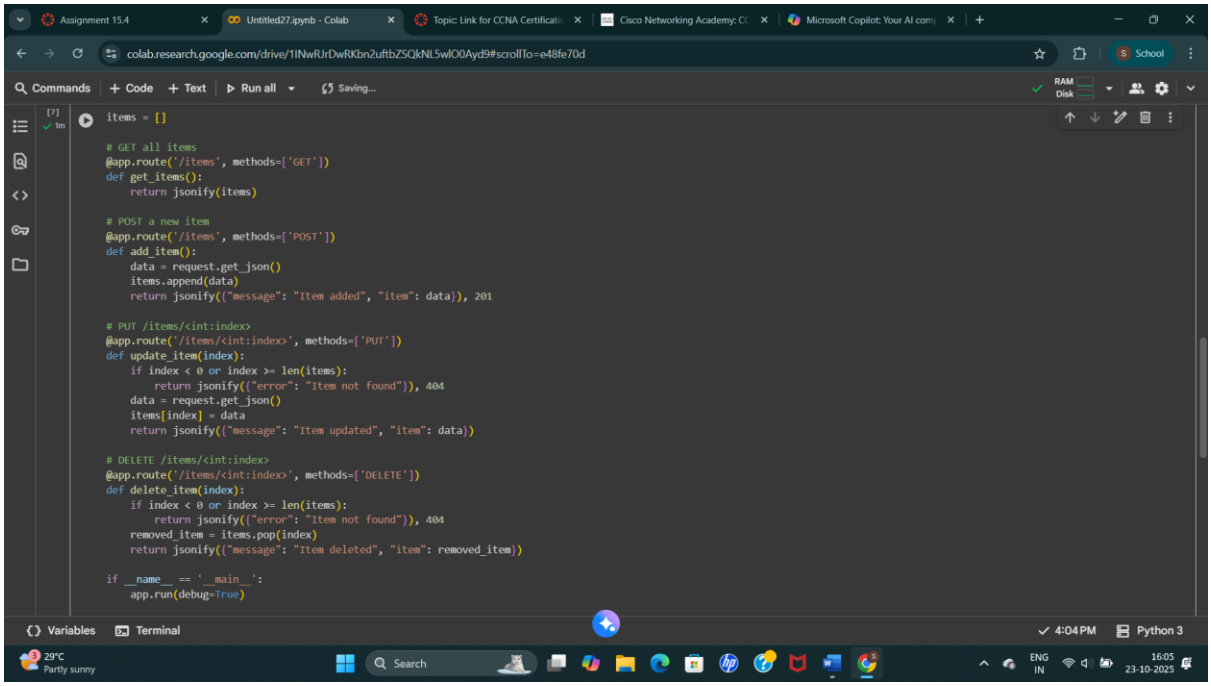
Starter Code:

```
# DELETE /items/<int: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})
Expected Output:
DELETE /items/0 →
{"message": "Item deleted", "item": {"name":"Notebook","price":250}}
• GET /items → []

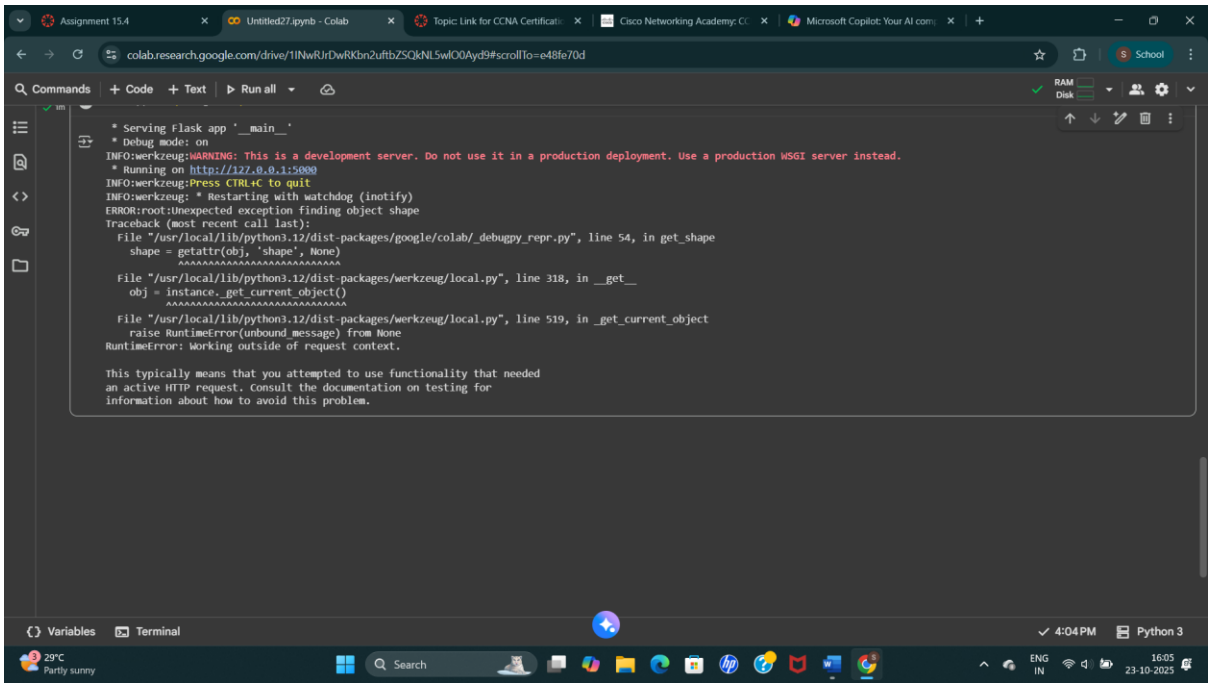
CODE:





```
items = []

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

# POST a new item
@app.route('/items', methods=['POST'])
def add_item():
    data = request.get_json()
    items.append(data)
    return jsonify({"message": "Item added", "item": data}), 201

# PUT /items/<int:index>
@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})

# DELETE /items/<int: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})

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

OUTPUT:



```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug: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
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with watchdog (inotify)
ERROR:root:Unexpected exception finding object shape
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/google/colab/_debugpy_repr.py", line 54, in get_shape
    shape = getattr(obj, 'shape', None)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 318, in __get__
    obj = instance._get_current_object()
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 519, in _get_current_object
    raise RuntimeError(unbound_message) from None
RuntimeError: Working outside of request context.

This typically means that you attempted to use functionality that needed
an active HTTP request. Consult the documentation on testing for
information about how to avoid this problem.
```

Task 5: Add Auto-Generated Documentation

Instructions:

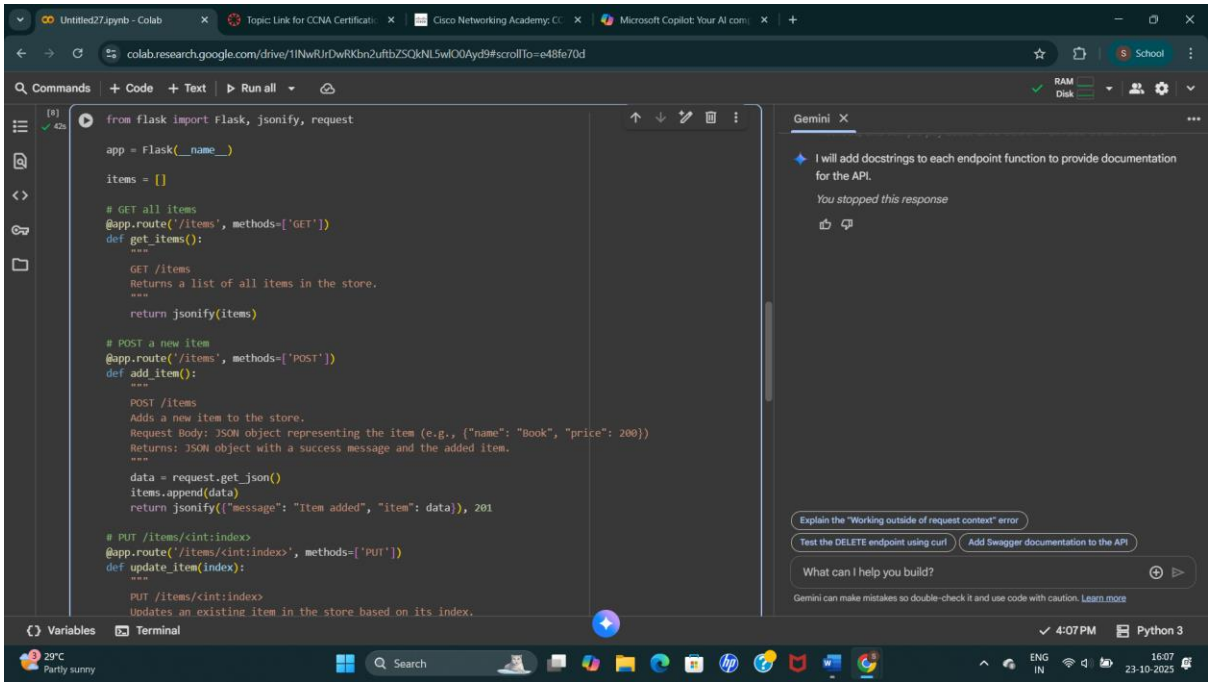
- Use AI to add inline comments and docstrings for all endpoints.
- Optionally, integrate Swagger / Flask-RESTX to auto-generate API documentation.

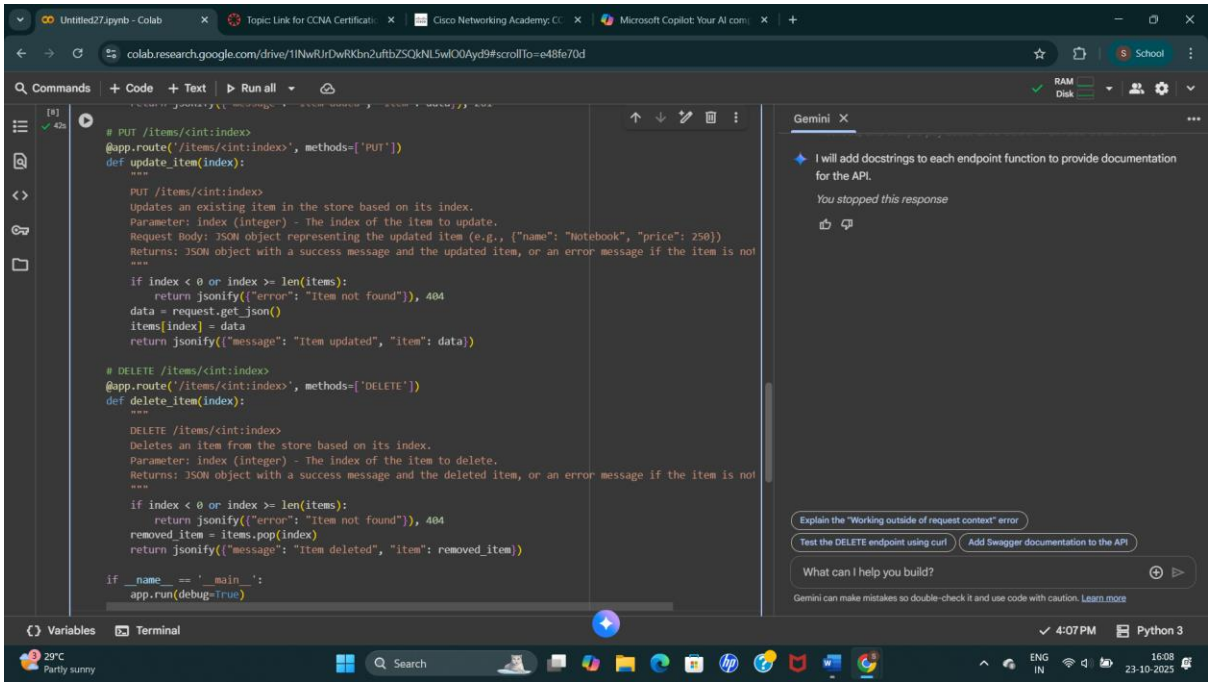
Starter Comment Example:

@app.route('/items', methods=['GET'])

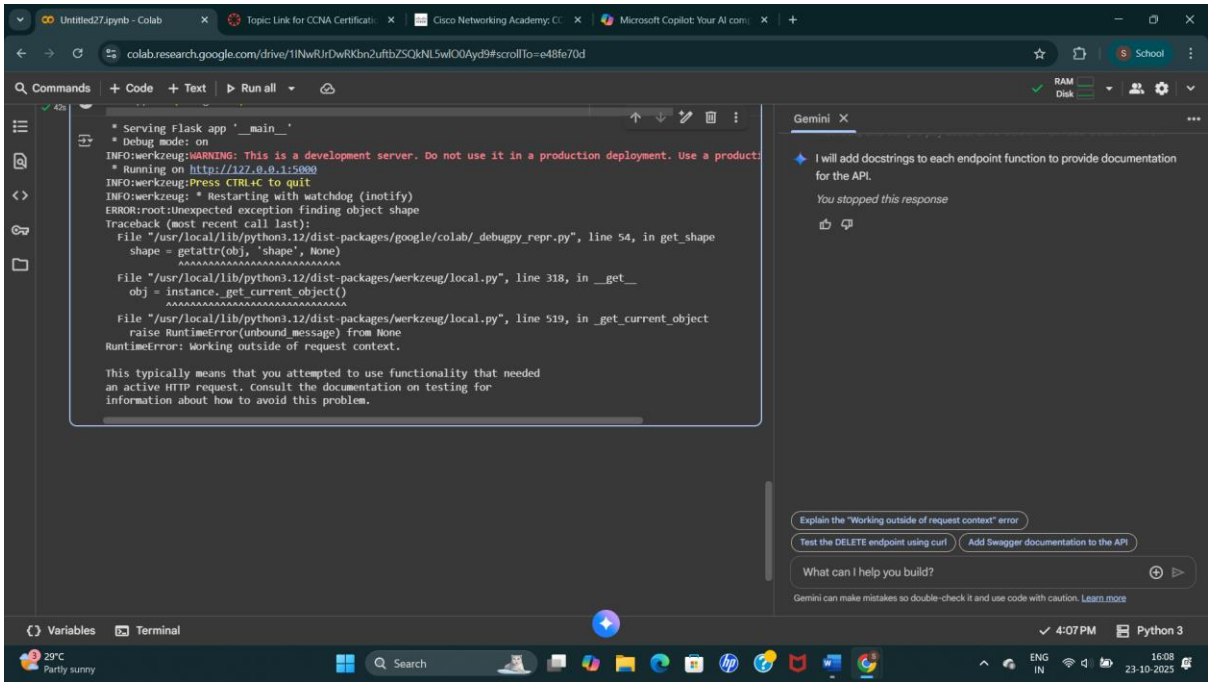
def get_items():
"""
GET /items
Returns a list of all items in the store.
"""
return jsonify(items)
Expected Output:
• Clear documentation for all endpoints, e.g., via /docs if using Swagger.
• Students should be able to see endpoint description, methods, and sample payloads.

CODE:




Untitled27.ipynb - Colab
colab.research.google.com/drive/1INwRJrDwRKbn2uftbZSQkNL5wlO0Ayd9#scrollTo=e48fe70d
Commands
+ Code
+ Text
Run all
PUT /items/<int:index>
@app.route('/items/<int:index>', methods=['PUT'])
def update_item(index):
"""
PUT /items/<int:index>
Updates an existing item in the store based on its index.
Parameter: index (integer) - The index of the item to update.
Request Body: JSON object representing the updated item (e.g., {"name": "Notebook", "price": 250})
Returns: JSON object with a success message and the updated item, or an error message if the item is not
"""
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})
DELETE /items/<int:index>
@app.route('/items/<int:index>', methods=['DELETE'])
def delete_item(index):
"""
DELETE /items/<int:index>
Deletes an item from the store based on its index.
Parameter: index (integer) - The index of the item to delete.
Returns: JSON object with a success message and the deleted item, or an error message if the item is not
"""
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})
if __name__ == '__main__':
app.run(debug=True)
Gemini
I will add docstrings to each endpoint function to provide documentation for the API.
You stopped this response
Explain the "Working outside of request context" error
Test the DELETE endpoint using curl
Add Swagger documentation to the API
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
4:07 PM
Python 3

OUTPUT:


Untitled27.ipynb - Colab
colab.research.google.com/drive/1INwRJrDwRKbn2uftbZSQkNL5wlO0Ayd9#scrollTo=e48fe70d
Commands
+ Code
+ Text
Run all
* Serving Flask app '__main__'
* Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a product
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with watchdog (inotify)
ERROR:root:Unexpected exception finding object shape
Traceback (most recent call last):
File "/usr/local/lib/python3.12/dist-packages/google/colab/_debugpy_repr.py", line 54, in get_shape
shape = getattr(obj, 'shape', None)
File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 318, in __get__
obj = instance._get_current_object()
File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 519, in _get_current_object
raise RuntimeError(unbound_message) from None
RuntimeError: Working outside of request context.
This typically means that you attempted to use functionality that needed
an active HTTP request. Consult the documentation on testing for
information about how to avoid this problem.
Gemini
I will add docstrings to each endpoint function to provide documentation for the API.
You stopped this response
Explain the "Working outside of request context" error
Test the DELETE endpoint using curl
Add Swagger documentation to the API
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
4:07 PM
Python 3