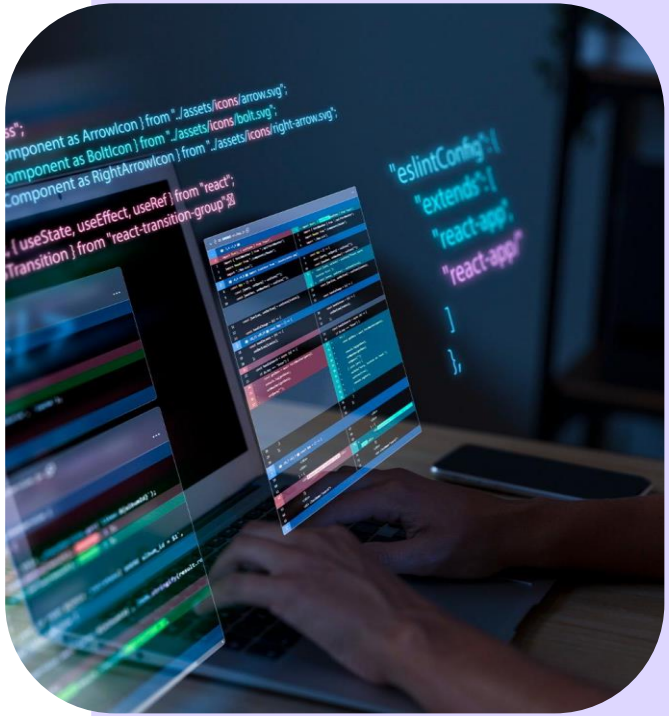
ASEDS Technical Test



Made by : lambarki aymane

Context :

We're creating a system to interact with ChatGPT, storing questions and their corresponding answers in a simple database-like microservice. Duplicate entries are acceptable for our needs.

Objective :

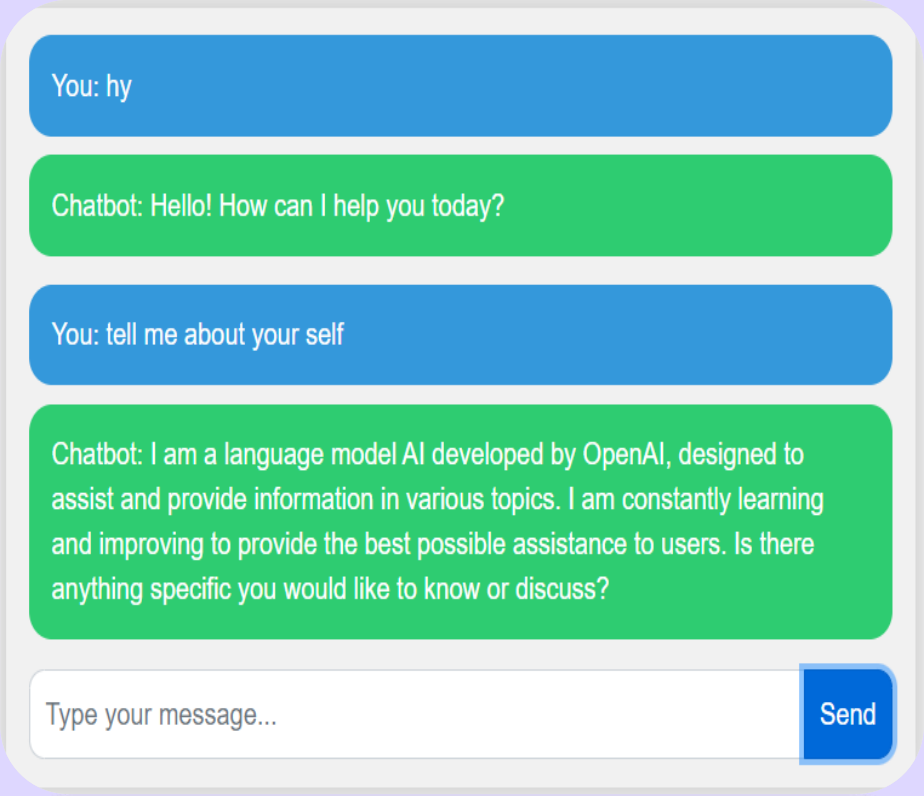
1-Create a microservice to interact with the ChatGPT API, accepting user-inputted questions.

2-Package the microservice in a Docker container for easy deployment.

3-Upon invocation, the microservice should query the ChatGPT API, extract the answer for the provided question, and store this information locally.(**Memorizing**)

4-Utilize a CSV file stored in a volume, establishing a mapping between a location on the host and a location in the container system for efficient data storage.

1 The microservice :



- **Generate an API key**
- **Develop an endpoint**
- **Handle requests**
- **Manage responses**

```python
api_key = [redacted]
client = OpenAI(api_key=api_key)
```

```python
@app.route('/api', methods=['POST'])
def chat_with_gpt():
```

```python
def chat_with_gpt():
    try:
        user_input = request.json.get('user_input', '')
        if not user_input:
            return jsonify({'error': 'Invalid input'})
```

```python
response = client.chat.completions.create(
    model="gpt-3.5-turbo-1106",
    messages=conversation_history
)

if response.choices and response.choices[0].message:
    content = response.choices[0].message.content.strip()
```

Generate an API key

Develop an endpoint

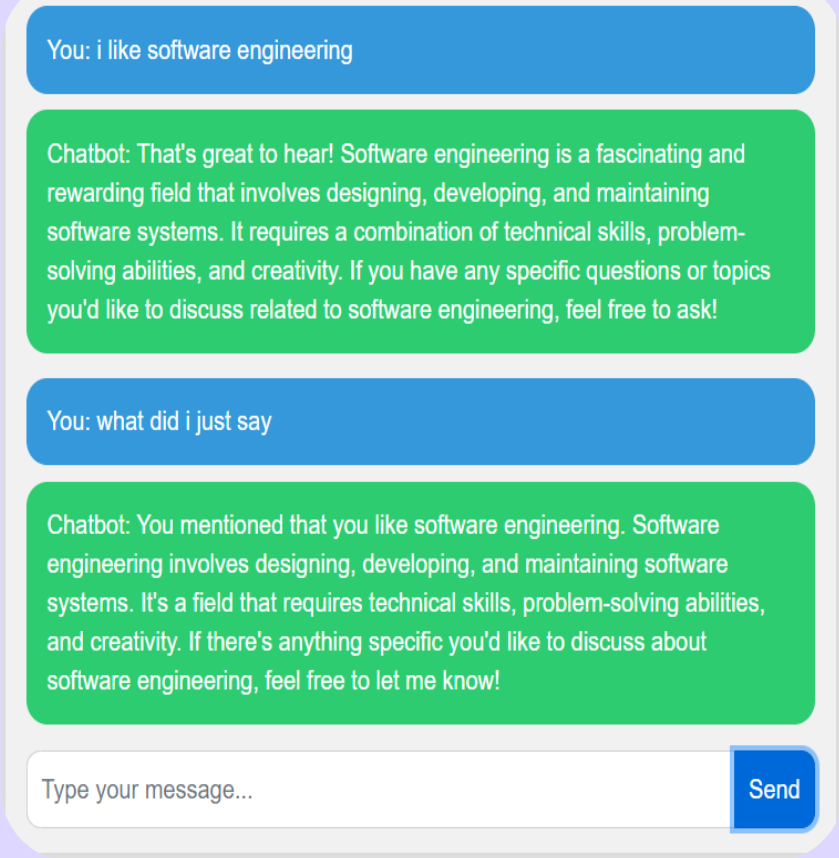
Handle requests

Manage responses

2 Memorizing :

i like software engineering	That's great to hear! Software engineering is a fas
what did i just say	You mentioned that you like software engineering.

The chatbot stores the conversation in a CSV file and adapts its next response



Initializes the file path and empty list

```
csv_file_path = 'chatbot_responses.csv'
conversation_history = []
```

Create and open the specified CSV file

```
with open(csv_file_path, 'w', newline='', encoding='utf-8') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['User Input', 'Chatbot Response'])
```

Stores the input and the response in the list

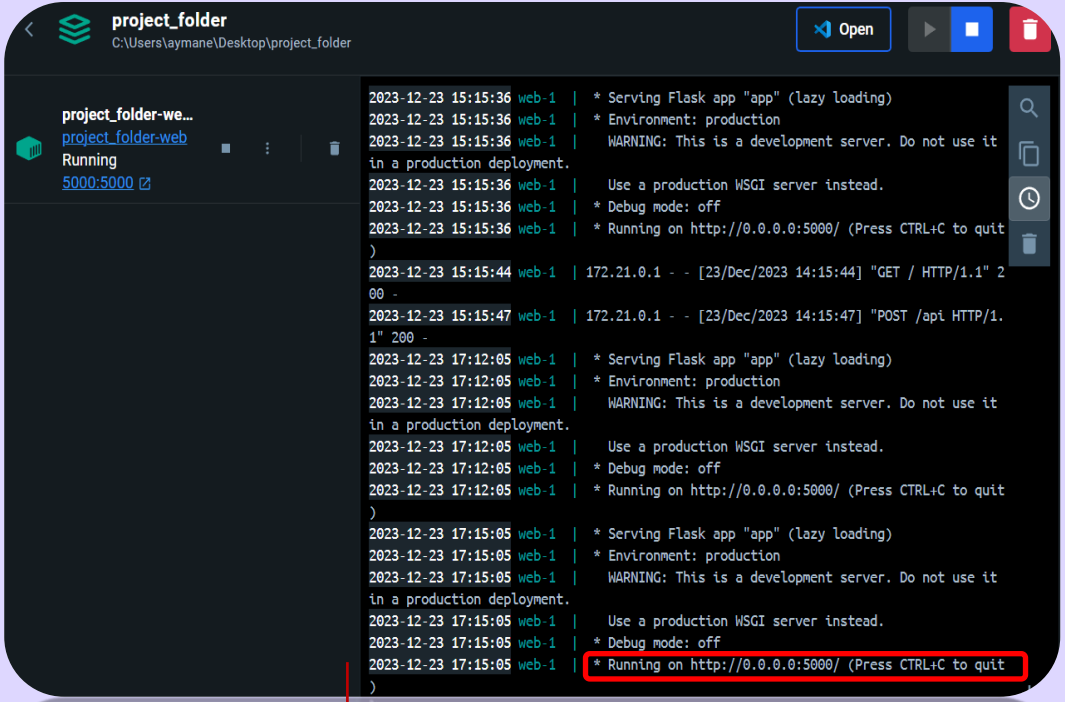
```
conversation_history.append({"role": "user", "content": user_input})
conversation_history.append({"role": "assistant", "content": content})
```

3 Package the work with docker :

Structure of the project :

```
project_folder/
|
├── app.py
├── requirements.txt
├── templates/
|   └── index.html
├── Dockerfile
└── docker-compose.yml
```

```
C:\Users\aymane\Desktop\project_folder>docker-compose up
[+] Running 1/1
 ⠿ Container project_folder-web-1  Recreated
Attaching to web-1
web-1  |  * Serving Flask app "app" (lazy loading)
web-1  |  * Environment: production
web-1  |    WARNING: This is a development server. Do not use it in a production deployment.
web-1  |    Use a production WSGI server instead.
web-1  |  * Debug mode: off
web-1  |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```



Running in http://localhost:5000/

How to use the Local Chatbot :

```
C:\Users\aymane>cd .\Desktop\project_folder
```

Open the terminal in the project folder

```
C:\Users\aymane\Desktop\project_folder>docker-compose build
[+] Building 52.7s (10/10) FINISHED
 => [web internal] load build definition from Dockerfile
 => => transferring dockerfile: 576B
 => [web internal] load .dockerignore
```

Run the (docker-compose build) command

```
C:\Users\aymane\Desktop\project_folder>docker-compose up
[+] Running 1/0
 ⠿ Container project_folder-web-1  Recreated
Attaching to web-1
web-1  |  * Serving Flask app "app" (lazy loading)
web-1  |  * Environment: production
web-1  |    WARNING: This is a development server. Do not use it in a production deployment.
web-1  |    Use a production WSGI server instead.
web-1  |  * Debug mode: off
web-1  |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Run the (docker-compose up) command you will find your chatbot in localhost:5000

THANKS!

Do you have any questions?

fscrinne@gmail.com

+212 663592697