OpenAI Assistant: Create a Code to UML-diagram generator

Key Takeaways

Task 1

Title: Create an assistant using playground

- Prompts should include context and specific instructions. If they are likely to be repeated, you can pre-fill these instructions.
- OpenAI Assistant API use might not always be free

Task 2

Title: Fine-tune the prompt

- Instructions are user-prompt-independent instructions
- Threads are a set of messages. Like ChatGPT conversations
- Instructions are user input helpers which reduce the prompt amount introduced by the user
- Clear a thread to reset the context from previous interactions

Task 3

Title: Activate the code interpreter

- This feature provides a clear separation of the user prompt, system prompt and file input
- Code interpreter feature not only interpretes input files, it creates a standalone environment, where code will be written and run
- Files can be feed in from the user thread, as an internal asset within the assistant, or generated within the session
- This feature is not for free. At this moment is priced by at $0.03 / session

Task 4

Title: Execute the assistant programmatically

- Install Open AI Python SDK using
  - `pip install openai`

- Create an authenticated client with your API_KEY SDK using
    - `client = OpenAI(api_key=”API_KEY”)`
- Create an assistant using
    - `my_assistant = client.beta.assistants.create(instructions=”INSTRUCTIONS”, model="gpt-4-turbo")`
- Create a thread using
    - `empty_thread = client.beta.threads.create()`
- Create a thread using
    - `message = client.beta.threads.messages.create(THREAD_ID, role="user", content="MESSAGE")`
- Run the thread
    - `run = client.beta.threads.runs.create(thread_id=THREAD_ID, assistant_id=ASSISTANT_ID )`

Task 5

Title: Cancel a deployment with an announcing pop-up

- FastAPI chosen to facilitates the interaction through their automatic generated interactive docs
- Add parameter `stream = True` to obtain a server side events which notifies you when to the thread has finish to process

Task 6

Title: Display the output with MermaidJS

- String manipulation with python is more robust than prompt engineering
- Mermaid code extraction and clean up is done within the frontend, but could be a feature of the backend system
- Frontend is modular separated, so you can build your frontend with the framework or programming language of your choice
- You can test this Code 2 Diagram with any programming language models file input.

Additional Resources:

- OpenAI API Docs
- Frontend Git repository
- Backend Git repository