UML images to PlantUML code

April 11, 2024

[19]:
```
pip install openai==0.28
```

```
Requirement already satisfied: openai==0.28 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages
(0.28.0)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: requests>=2.20 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
openai==0.28) (2.28.2)
Requirement already satisfied: tqdm in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
openai==0.28) (4.66.1)
Requirement already satisfied: aiohttp in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
openai==0.28) (3.9.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
requests>=2.20->openai==0.28) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
requests>=2.20->openai==0.28) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
requests>=2.20->openai==0.28) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
requests>=2.20->openai==0.28) (2022.12.7)
Requirement already satisfied: aiosignal>=1.1.2 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
aiohttp->openai==0.28) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
aiohttp->openai==0.28) (22.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
aiohttp->openai==0.28) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
aiohttp->openai==0.28) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
aiohttp->openai==0.28) (1.9.4)
Requirement already satisfied: colorama in
c:\users\hp\appdata\local\programs\python\python311\lib\site-packages (from
tqdm->openai==0.28) (0.4.6)
```

0.0.1 Here, I imported all the libraries

[20]:
```
import base64
import json
import os
import openai
from PIL import Image
from IPython.display import display, Markdown
```

0.0.2 This is the function used to extract PlantUML code from the output

[21]:
```
def extract_and_display_code(message):
    start_index = message.find("@startuml")
    end_index = message.find("@enduml")
    if start_index != -1 and end_index != -1:
        extracted_code = message[start_index:end_index + len("@enduml")]
        return display(Markdown("```plantuml\n{}\n```".format(extracted_code)))
    else:
        return "1"
```

0.0.3 This function encodes the image into base64 format.

[22]:
```
def encode_image(image_path):
    with open(image_path, "rb") as image_file:
        return base64.b64encode(image_file.read()).decode('utf-8')

api_key = "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"
openai.api_key = api_key
```

0.1 Approach: (First Approach)

0.1.1 First i generated the description of the UML image and then I used the UML image and the image description to generate PlantUML code using GPT - 4 vision preview. This approach wasn't showing good results.

[23]:
```
def generate_description_from_image(image_url):
    response = openai.ChatCompletion.create(
        model='gpt-4-vision-preview',
        messages=[
            {
```

```
                "role": "user",
                "content": [
                    {"type": "text", "text":  "Describe a PlantUML diagram␣
↪depicting a user authentication flow and such that it's description can be␣
↪used to generate PlantUML code .CORRECTLY check whether it is a participant␣
↪or actor and also the colour.FOCUS ON EVERY MINUTE DETAIL."},
                    {
                        "type": "image_url",
                        "image_url": image_url
                    }
                ],
            }
        ],
        max_tokens=300,
    )
    description = response.choices[0].message.content

    return description
```

[24]:
```
def generate_plantuml_code(description, image_url):
    response = openai.ChatCompletion.create(
        model='gpt-4-vision-preview',
        messages=[
            {
                "role": "user",
                "content": [
                    {"type": "text", "text": description},
                    {
                        "type": "image_url",
                        "image_url": image_url
                    },
                    {"type": "text", "text": "Generate the PlantUML code using␣
↪the above description and the image, now correct the PlantUMLcode for every␣
↪minute difference between the image and code depicting the image."}
                ],
            }
        ],
        max_tokens=250,
    )
    plantuml_code_response = response.choices[0].message.content
    return plantuml_code_response
```

[25]:
```
image_local = "D:\\HP\\users\\OneDrive\\Pictures\\Screenshots\\Screenshot␣
↪2024-04-09 160445.png"
image_url = f"data:image/png;base64,{encode_image(image_local)}"
description = generate_description_from_image(image_url)
plantuml_code = generate_plantuml_code(description, image_url)
```

```
print("Generated PlantUML code:")

if (extract_and_display_code(plantuml_code))!= "1" :
    f = 0
    pass
else:
    if (extract_and_display_code(description))!= "1" :
        pass
    else:
        print(plantuml_code)
```

```
Generated PlantUML code:
Generated PlantUML code:


@startuml
actor Bob #red
participant "Alice" as A #green
participant "I have a really long name" as SYS #green

Bob -> A: Authentication Request
A -> Bob: Authentication Response
Bob -> SYS: Log transaction

@enduml
```

0.2 Second Approach :

0.2.1 Directly generate PlantUML code and description from the image by providing image and prompt to GPT - 4 vision Preview and then extracting a PlantUML code from the response.

[27]:
```
print("Generated PlantUML code:")
if (extract_and_display_code(description))!= "1" :
    pass
else:
    print(plantuml_code)
```

```
Generated PlantUML code:

@startuml
actor Bob #red
participant "Alice" as A #green
participant "I have a really long name" as SYS #green

Bob -> A: Authentication Request
A -> Bob: Authentication Response
Bob -> SYS: Log transaction

@enduml
```

0.2.2 To compare the results you can copy and paste the code here [https://www.planttext.com/] and compare with the original UML image.

[ ]:
```

```