**ChatBot**

**ChatGPT version:** 4o-mini

**My Prompt:**

Modify the code to generate a chatbot through the ChatGPT API, the “messages” sent to the ChatGPT API need to include the user question and the historical conversation between the user and chatbot. The following code shows how to load the existing conversation in the “chatbot” array and the new user input to the “messages” that are then sent to the ChatGPT API: Please make sure that in Gradio UI interface, it shows all user’s inputs and ChatGPT responses in the chatbot box. Moreover, the chatbot can understand the context of the conversation. The following shows an example:

***Code snippet(I included part3 code as well):***

*messages = []*

*for input\_text, response\_text in chatbot:*

*messages.append({'role': 'user', 'content': input\_text})*

*messages.append({'role': 'assistant', 'content': response\_text})*

*messages.append({'role': 'user', 'content': user\_input})*

*response = client.chat.completions.create(*

*model="gpt-3.5-turbo",*

*messages = messages,*

*temperature = temp,*

*max\_tokens=200,*

*)*

*chatbot.append((user\_input, response.choices[0].message.content))*

**ChatGPT response:**To modify the code for creating a chatbot using the ChatGPT API while maintaining the historical conversation between the user and the chatbot, we need to:

1. Track the conversation history: Store both user inputs and ChatGPT responses in the chatbot array.
2. Update the interact\_summarization function: This function will need to accumulate conversation history and ensure that the entire conversation is passed to the API.
3. Gradio UI: Ensure that the Gradio interface shows both the user's questions and the chatbot's responses in real-time.

Code:  
*import gradio as gr*

*from typing import List, Tuple*

*# Initialize the chatbot conversation history*

*chatbot = []*

*# Function to reset the conversation history*

*def reset() -> List:*

*global chatbot*

*chatbot = []*

*return chatbot*

*# Function to interact with the chatbot and maintain conversation history*

*def interact\_summarization(user\_input: str, temp = 1.0) -> List[Tuple[str, str]]:*

*global chatbot*

*# Prepare the input for the ChatGPT API*

*messages = [{'role': 'user', 'content': user\_input}]*

*# Add historical messages to the current conversation context*

*for input\_text, response\_text in chatbot:*

*messages.append({'role': 'user', 'content': input\_text})*

*messages.append({'role': 'assistant', 'content': response\_text})*

*# Call the ChatGPT API*

*response = client.chat.completions.create(*

*model="gpt-3.5-turbo",*

*messages=messages,*

*temperature=temp,*

*max\_tokens=200*

*)*

*# Extract the response from the API*

*assistant\_response = response.choices[0].message.content*

*# Append the new user input and response to the chatbot history*

*chatbot.append((user\_input, assistant\_response))*

*# Return the updated chatbot history for display*

*return chatbot*

*# Define the Gradio UI interface*

*with gr.Blocks() as demo:*

*gr.Markdown("# Chatbot with conversation history")*

*chatbot\_ui = gr.Chatbot()*

*user\_input\_textbox = gr.Textbox(label="Ask a question", interactive=True)*

*with gr.Column():*

*gr.Markdown("# Temperature\n Temperature controls how creative the responses will be.")*

*temperature\_slider = gr.Slider(0.0, 2.0, 1.0, step=0.1, label="Temperature")*

*with gr.Row():*

*send\_button = gr.Button(value="Send")*

*reset\_button = gr.Button(value="Reset")*

*# Connect UI components to functions*

*send\_button.click(interact\_summarization, inputs=[user\_input\_textbox, temperature\_slider], outputs=[chatbot\_ui])*

*reset\_button.click(reset, outputs=[chatbot\_ui])*

*# Launch the Gradio interface*

*demo.launch(debug=True)*

Then ,I renamed the variable/function names and few changes to the final code.