from openai import OpenAI

import time

import streamlit as st

from PIL import Image

import pyttsx3

import threading

import pydub

from pydub.playback import play

def main():

    engine = pyttsx3.init()

    voices = engine.getProperty('voices')

    voice\_id = 1  # Select the desired voice index

    engine.setProperty('voice', voices[voice\_id].id)

    rate = engine.getProperty('rate')

    engine.setProperty('rate', rate - 8)

    st.set\_page\_config(

        page\_title="Cigna AI Assistant",

        page\_icon="📚",

        layout="wide"

    )

    #image\_path = r'C:\Users\jacqu\assistant-api-streamlit-chatbot\2.jpg'

    #st.image(image\_path, caption='')

    video\_path = r'C:\Users\jacqu\assistant-api-streamlit-chatbot\video.mp4'

    st.video(video\_path,start\_time=0)

    image\_path = r'C:\Users\jacqu\assistant-api-streamlit-chatbot\2.png'

    st.sidebar.image(image\_path, caption='')

    # Add a selectbox to the sidebar:

    add\_selectbox = st.sidebar.selectbox(

    'How would you like to be contacted?',

    ('Email', 'Home phone', 'Mobile phone')

)

    slider\_value = st.sidebar.slider("How satisfied were you with your last Cigna interaction?", 0, 100, 50)

    image\_path = r'C:\Users\jacqu\assistant-api-streamlit-chatbot\new.png'

    st.sidebar.image(image\_path, caption='')

    api\_key = "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"

    assistant\_id = "asst\_mHv0pDlN8le4m7xtKPi7QMzI"

    # Initiate st.session\_state

    st.session\_state.client = OpenAI(api\_key=api\_key)

    if "messages" not in st.session\_state:

        st.session\_state.messages = []

    if "start\_chat" not in st.session\_state:

        st.session\_state.start\_chat = False

    if st.session\_state.client:

        st.session\_state.start\_chat = True

    if st.session\_state.start\_chat:

        # Display existing messages in the chat

        for message in st.session\_state.messages:

            with st.chat\_message(message["role"]):

                st.markdown(message["content"])

                # Accept user input

        if prompt := st.chat\_input("Hello how can I help?"):

            # Add user message to chat history

            st.session\_state.messages.append({"role": "user", "content": prompt})

            # Display user message in chat message container

            with st.chat\_message("user"):

                st.markdown(prompt)

            # Create a thread

            st.session\_state.thread = st.session\_state.client.beta.threads.create()

            # Add a Message to the thread

            st.session\_state.client.beta.threads.messages.create(

                thread\_id=st.session\_state.thread.id,

                role="user",

                content=prompt,

            )

            # As of now, assistant and thread are not associated to eash other

            # You need to create a run in order to tell the assistant at which thread to look at

            run = st.session\_state.client.beta.threads.runs.create(

                thread\_id=st.session\_state.thread.id,

                assistant\_id=assistant\_id,

            )

            # with while loop continuously check the status of a run until it neither 'queued' nor 'in progress'

            def wait\_for\_complete(run, thread):

                while run.status == "queued" or run.status == "in\_progress":

                    run = st.session\_state.client.beta.threads.runs.retrieve(

                        thread\_id=thread.id,

                        run\_id=run.id,

                    )

                    time.sleep(0.5)

                return run

            run = wait\_for\_complete(run, st.session\_state.thread)

            # once the run has completed, list the messages in the thread -> they are ordered in reverse chronological order

            replies = st.session\_state.client.beta.threads.messages.list(

                thread\_id=st.session\_state.thread.id

            )

            # This function will parse citations and make them readable

            def process\_replies(replies):

                citations = []

                # Iterate over all replies

                for r in replies:

                    if r.role == "assistant":

                        message\_content = r.content[0].text

                        annotations = message\_content.annotations

                        # Iterate over the annotations and add footnotes

                        for index, annotation in enumerate(annotations):

                            # Replace the text with a footnote

                            message\_content.value = message\_content.value.replace(

                                annotation.text, f" [{index}]"

                            )

                            # Gather citations based on annotation attributes

                            if file\_citation := getattr(

                                annotation, "file\_citation", None

                            ):

                                cited\_file = st.session\_state.client.files.retrieve(

                                    file\_citation.file\_id

                                )

                                citations.append(

                                    f"[{index}] {file\_citation.quote} from {cited\_file.filename}"

                                )

                            elif file\_path := getattr(annotation, "file\_path", None):

                                cited\_file = st.session\_state.client.files.retrieve(

                                    file\_path.file\_id

                                )

                                citations.append(

                                    f"[{index}] Click <here> to download {cited\_file.filename}"

                                )

                # Combine message content and citations

                full\_response = message\_content.value + "\n" + "\n".join(citations)

                return full\_response

            # Add the processed response to session state

            processed\_response = process\_replies(replies)

            st.session\_state.messages.append(

                {"role": "assistant", "content": processed\_response}

            )

            # Display assistant response in chat message container

            with st.chat\_message("assistant"):

                st.markdown(processed\_response, unsafe\_allow\_html=True)

                engine.say(processed\_response)

                engine.runAndWait()

                #speak\_thread = threading.Thread(target=speak\_text, args=(processed\_response,))

                #speak\_thread.start()

#def speak\_text(text):

# engine = pyttsx3.init()

# engine.say(text)

#  engine.runAndWait()

if \_\_name\_\_ == "\_\_main\_\_":

    main()