CHATBOT WITH SPEECH RECOGNITION

Introduction:

The project is a Chatbot with Speech Recognition and GPT-3.5 API integration. It allows users to interact with the chatbot through text input or speech recognition. The chatbot leverages GPT-3.5, a powerful language model by OpenAI, to provide accurate answers to user questions. The user interface is designed to resemble a chat-like environment, providing a conversational experience.

Key Components and Libraries Used:

A. **Streamlit:** A Python library for building web applications easily and quickly.

B. **Speech Recognition:** A library for converting speech to text.

C. **TextBlob:** A library for spelling correction and text processing.

D. **OpenAI GPT-3.5 API:** An API for accessing the GPT-3.5 language model from OpenAI.

Main Features:

A. **Text Input:** Users can type their questions into an input box to interact with the chatbot.

B. **Speech Recognition:** Users have the option to enable speech recognition to ask questions verbally.

C. **Spelling Correction:** The user's text input is checked for spelling mistakes and corrected using TextBlob.

D. **GPT-3.5 API Integration:** The chatbot uses the GPT-3.5 engine to generate accurate answers to user questions.

E. **Chat History:** Previous user queries and chatbot responses are displayed in a sidebar, providing a chat history.

F. **User Interface:** The user interface resembles a chat-like environment, making it easy for users to interact with the chatbot.

Workflow:

A. **User Interaction:** Users can interact with the chatbot by typing their questions or using speech recognition.

B. **Spelling Correction:** If the user's text input has spelling mistakes, they are automatically corrected using TextBlob.

C. **GPT-3.5 Integration:** The corrected user query is passed to the GPT-3.5 engine through the OpenAI API for generating an answer. D. **Chatbot Response:** The chatbot's response is displayed on the interface, providing an accurate answer to the user's question.

E. **Chat History:** Each user query and the corresponding chatbot response are stored in the sidebar as a chat history.

F. **New Chat Option:** Users can start a new chat by clicking a button, which resets the input box and chat history.

Benefits:

A. **Natural Language Interaction:** Users can interact with the chatbot using both text and speech, making it convenient for different users.

B. **Accurate Answers:** The integration with GPT-3.5 ensures that the chatbot provides accurate and relevant answers to user queries.

C. **Conversational Experience:** The chat-like interface creates a conversational experience, making the interaction more engaging.

Codings:

```
import streamlit as st
import speech_recognition as sr
from textblob import TextBlob
import openai

# Function to correct the spelling of input text
def correct_spelling(input_text):
    if not input_text:
        return input_text

    blob = TextBlob(input_text)
    corrected_text = blob.correct()
    return str(corrected_text)

# Function to chat with GPT-3.5 using OpenAI API
def chat_with_gpt(input_topic):
    # GPT-3.5 API call using OpenAI API
    # Replace 'YOUR_OPENAI_API_KEY' with your actual OpenAI API key
    openai.api_key = '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'
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=input_topic,
        max_tokens=100
    )
    answer = response['choices'][0]['text'].strip()
    return answer
```

```
def main():
    st.title("Ask your Chatbot")

    # Sidebar with black background containing chat history and previous
chats
    st.sidebar.title("Chat History")
    chat_history = st.sidebar.empty()
    previous_chats = []

    # Text input box for user to type or use speech recognition
    input_topic = st.text_input("Your Message:")
    speech_button = st.button("🎤 Voice Recognize")

    if speech_button:
        # Use speech recognition to get user input
        recognizer = sr.Recognizer()
        with sr.Microphone() as source:
            st.write("Start Asking your question...")
            audio = recognizer.listen(source)

        try:
            st.write("Please wait...")
            input_topic = recognizer.recognize_google(audio)
            st.text_area("Your Message:", value=input_topic,
key='input_text')
        except sr.UnknownValueError:
            st.write("Speech Recognition could not understand audio.")
        except sr.RequestError:
            st.write("Could not request results from Speech Recognition
service.")

    if input_topic:
        # Correct the input sentence for spelling mistakes
        corrected_topic = correct_spelling(input_topic)

        # Chat with GPT and get the answer
        answer = chat_with_gpt(corrected_topic)

        # Save chat history
        previous_chats.append((corrected_topic, answer))

        # Display previous chats in sidebar
        chat_history.markdown("\n\n".join([f"**You:** {chat[0]}\n\n**Bot:**
{chat[1]}" for chat in previous_chats]))

        # Display the answer
        st.subheader("Answer:")
        st.write(answer)

if __name__ == "__main__":
    main()
```

Sample output of the project:

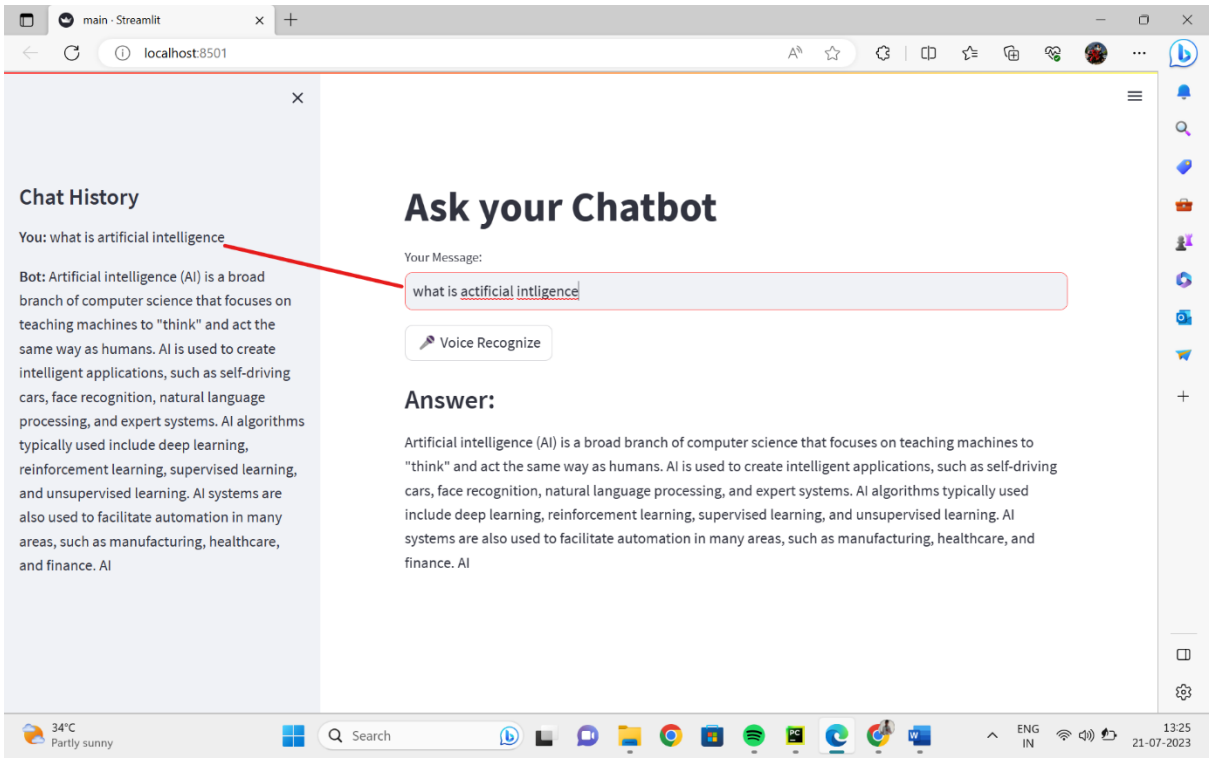

Fig1: Autocorrect the input text by using textblob

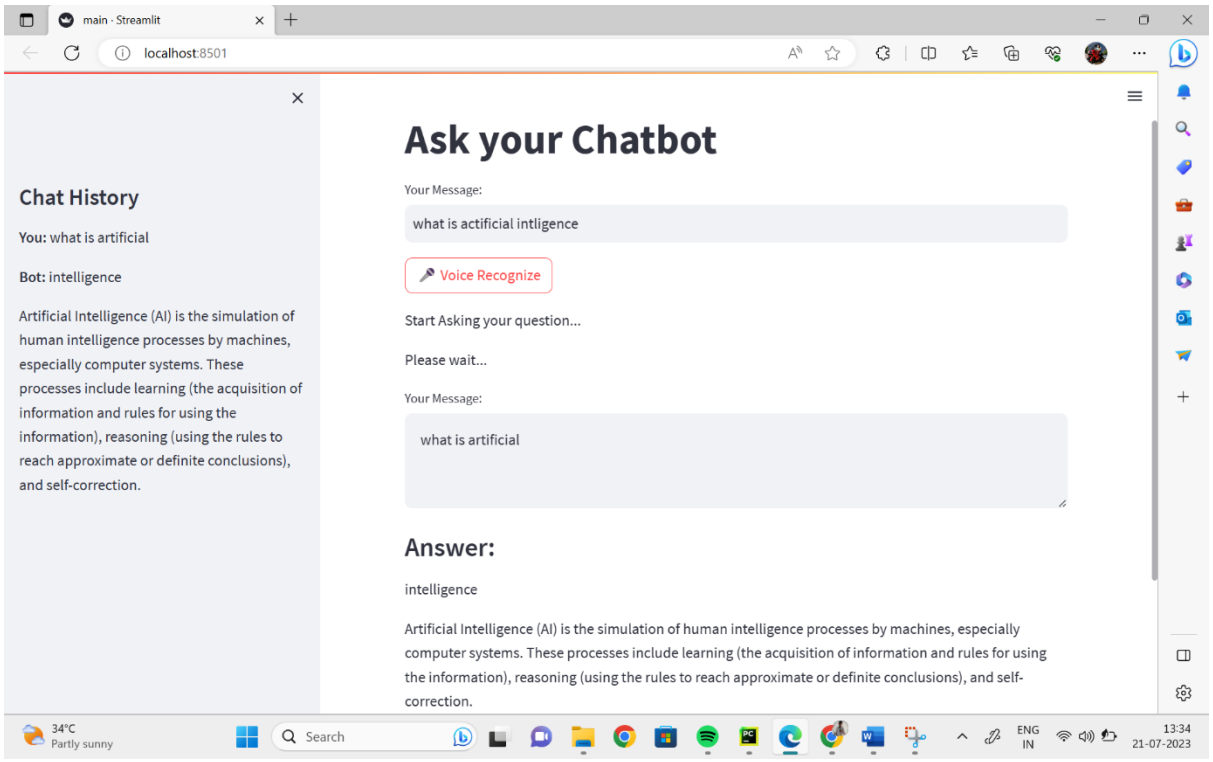

Fig2: Using voice recognization to ask questions

Conclusion:

The Chatbot with Speech Recognition and GPT-3.5 API Integration project combines the power of speech recognition, spelling correction, and GPT-3.5 language model to deliver an interactive and accurate chatbot experience. The user-friendly interface, chat history, and seamless speech recognition make it a versatile and effective tool for answering user questions.