DOCUMENTATION ON THE DEVELOPMENT OF SPEECH RECOGNITION APP THAT CONVERTS SPOKEN WORDS TO TEXT.

This project was carried out by:

1. Omorere Idowu
2. Femi Adegbenro
3. Nwakwo Praise
4. Abraham Micheal
5. Adanlawo Oladimeji Kehinde

This is a real time speech transcription app that was built using Streamlit, SpeechRecognition and Threading libraries. The main function of this app is to convert and display spoken words to transcribed text in real time.

CODE EXPLANATION

1. **Importing libraries:** The following libraries were imported into the code: streamlit as st, speech_recognition as sr, threading and queue.
    - **streamlit (st)** was imported for building the web app.
    - **speech_recognition (sr)** was imported for its speech-to-text function.
    - **threading** was imported to run the transcription process in the background.
    - **queue** was imported to store the transcribed text.
2. **SpeechTranscriber Class:** The speech transcriber class function in this code is to initialize the speech recognizer, microphone, transcript queue flags (it's like a switch) for listening and threading. The codes that was used to achieve these are:
   i) class SpeechTranscriber: → This line of code defines a new class called speech transcriber and it contains methods and attributes relating to speech recognition.
   ii) def __init__(self): → This line defines the constructor method for the SpeechTranscriber class, and it is called when an object is created from the class, and it initializes the object's attributes.
   iii) self.recognizer = sr.Recognizer() → This line creates a new speech recognizer object using the sr.Recognizer() class from the speech recognition library. It was used to recognize spoken words and phrases.

iv) self.microphone = sr.Microphone() → This line creates a new microphone object using the sr.Microphone() class from the speech recognition library. It was used to capture the audio input from the user's microphone.

v) self.transcript_queue = queue.Queue() → This line of code creates a new queue object using the queue.Queue() class from the queue library. It was used to store the transcribed text from the speech recognizer.

vi) self.is_listening = False → This line of code initializes a flag variable called is_listening to False. It was used to indicate whether the speech transcriber is currently listening for audio input or not.

vii) self.thread = None → This line of code assigns a variable called thread to None. It was used to store a reference to a thread object that was created later to run the speech transcription process.

3. **Transcribe Audio Method:** The transcribe audio method adjusts for background noise, listens for audio input, recognizes the speech, and puts the transcribed text into the queue. Below is the function each chunk of code perform:

i) def transcribe_audio(self): → This line defines a method called transcribe_audio which is part of the SpeechTranscriber class.

ii) with self.microphone as source: → This line creates a context manager for the microphone object, it ensures that the microphone is properly configured and released when it goes out of scope.

iii) self.recognizer.adjust_for_ambient_noise(source) → This line adjusts the recognizer for background noise and it is done by analyzing the audio from the microphone for a short period of time (about 1 second) to determine the ambient noise level.

iv) st.write("Listening...") → This line of code is used to display a message "Listening..." in the Streamlit app to indicate that the transcription process has started.

v) while self.is_listening: → This line starts a loop that continues as long as the is_listening flag is True.

vi) try:

Listen for audio input

audio = self.recognizer.listen(source, timeout=None, phrase_time_limit=5) → This line listens for audio input from the microphone. The listen method blocks until it detects speech, and it has a timeout of 5 seconds after the last phrase.

vii) text = self.recognizer.recognize_google(audio) → This line recognizes the speech in the audio input using the Google Speech Recognition API.

vii) self.transcript_queue.put(text) → This line of code adds the recognized text to the transcript queue.

viii) except sr.UnknownValueError:

```
    st.write("Could not understand audio")
except sr.RequestError as e:
    st.write(f"Could not request results; {e}")
except Exception as e:
    st.write(f"An error occurred: {e}")
```

The above lines of code handle exceptions that may occur during the transcription process. If an exception occurs, an error message is displayed in the Streamlit.

4. **Start and Stop Transcription Methods:**
   A. **The start transcription method :** The lines of code in the start section start the transcription process by setting the listening flag, creating a new thread to run the transcription method, and starting the thread. Below is the function each line of code in the start transcription section perform:

      i) def start_transcription(self): → This line of code defines a method called start_transcription which is part of the SpeechTranscriber class.

      ii) self.is_listening = True → This line of code sets the is_listening flag to True, indicating that the transcription process should start listening for the audio input.

      iii) self.thread = threading.Thread(target=self.transcribe_audio) → This line creates a new thread object that will run the transcribe_audio method in the background, while the target parameter specifies the method that the thread should execute.

      iv) self.thread.start() → This line of code starts the thread, which begins executing the transcribe_audio method.

   B. **The stop transcription method:** The lines of code in the stop section code stops the transcription process by setting the listening flag to False and joining the transcription thread to ensure it terminates cleanly. Below is the function each line of code in the stop transcription section perform:

      i) def stop_transcription(self): → This line of code defines a method called stop_transcription which is part of the SpeechTranscriber class.

ii) self.is_listening = False → This line of code sets the is_listening flag to False, indicating that the transcription process should stop listening for audio input.

iii) if self.thread: → This line of code checks if a thread object has been assigned to the self.thread attribute.

iv) self.thread.join() → This line of code activates if a thread exists, this line joins the thread, which means that the main thread waits for the transcription thread to finish its execution before continuing then the join method ensures that the transcription thread is properly terminated and its resources are released.

5. The Main Function: This section contains the line of code that brought all the function of the speech recognizer app together because it contains codes that:

i) initializes the app, sets up the control buttons, and handles the transcription control logic.

ii) When the "Start Transcription" button is clicked, the start_transcription method is called.

iii) When the "Stop Transcription" button is clicked, the stop_transcription method is called.

iv) Allows the app to update the transcript display area in real-time using the transcript_queue.

HOW TO RUN THE APP

1. Install the required libraries by running pip install streamlit speechrecognition.
2. Save the code in a file named app.py.
3. Run the app by executing streamlit run app.py in the terminal.
4. Open a web browser and navigate to http://localhost:8501 to access the app.

OUTPUT

The app will display a simple interface with two buttons: "Start Transcription" and "Stop Transcription".