**Lab Report: Development of a Meeting Transcription and Summary Tool**

**Objective**

The aim of this project was to design and implement a Python-based tool capable of:

1. Transcribing speech input into text.
2. Summarizing the transcribed text using a pre-trained summarization model.
3. Converting the summary into speech output.

The tool integrates speech recognition, text summarization, and text-to-speech (TTS) functionalities to provide a seamless experience for users in meeting scenarios.

**Materials and Tools**

1. **Programming Language**: Python
2. **Libraries and Frameworks**:
   * gtts for text-to-speech conversion.
   * speech\_recognition for converting speech to text.
   * transformers (Hugging Face) for text summarization using the facebook/bart-large-cnn model.
   * os module for handling file operations and running system commands.
3. **Hardware**:
   * Microphone for speech input.
   * Computer with Python environment for executing the script.

**Methodology**

The tool consists of several components integrated into a single Python script to provide end-to-end functionality. Below is the step-by-step approach to building and running the program:

**1. Speech-to-Text Conversion**

* **Input**: Voice input from a microphone.
* **Process**:
  + speech\_recognition.Recognizer was used to capture and convert live audio input into text.
  + The audio was processed using Google Web Speech API.
  + Background noise adjustments were made with adjust\_for\_ambient\_noise.
* **Output**: Transcribed text in string format.

**2. Text Summarization**

* **Input**: Transcribed text.
* **Process**:
  + The Hugging Face facebook/bart-large-cnn model was used for summarizing the text.
  + The summarizer pipeline was configured to generate concise summaries with specified constraints (min\_length=30, max\_length=100).
* **Output**: Summarized text.

**3. Text-to-Speech Conversion**

* **Input**: Summarized text.
* **Process**:
  + The gtts library generated an audio file (summary.mp3) from the summarized text.
  + The system's media player was invoked using the os.system command to play the audio.
* **Output**: Audible summary of the transcription.

**4. User Interface**

A menu-driven interface was created to allow users to:

1. Start a new transcription.
2. Generate and read a summary of the transcription.
3. Exit the program.

**Code Implementation:**

from gtts import gTTS

import os

import speech\_recognition as sr

from transformers import pipeline

# Initialize the summarization model

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

def text\_to\_speech(text):

    tts = gTTS(text)

    tts.save("summary.mp3")

    os.system("start summary.mp3")

def speech\_to\_text():

    recognizer = sr.Recognizer()

    with sr.Microphone() as source:

        print("Please speak... (Press Ctrl+C to stop recording)")

        recognizer.adjust\_for\_ambient\_noise(source, duration=0.2)

        try:

            audio = recognizer.listen(source)

            print("Recognizing your voice...")

            text = recognizer.recognize\_google(audio)

            print(f"Transcription: {text}")

            return text

        except sr.UnknownValueError:

            print("Sorry, could not understand audio.")

            return None

        except sr.RequestError as e:

            print(f"Request Error: {e}")

            return None

def summarize\_text(text):

    try:

        print("Generating summary...")

        summary = summarizer(text, max\_length=100, min\_length=30, do\_sample=False)

        summary\_text = summary[0]['summary\_text']

        print(f"Summary: {summary\_text}")

        return summary\_text

    except Exception as e:

        print(f"Error during summarization: {e}")

        return "Could not generate summary."

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

    full\_transcription = []

    while True:

        print("\n--- Meeting Transcription and Summary Tool ---")

        print("1. Start Transcription")

        print("2. Generate and Read Summary")

        print("3. Exit")

        choice = input("Choose an option (1/2/3): ")

        if choice == '1':

            print("Starting transcription. Speak into the microphone...")

            transcription = speech\_to\_text()

            if transcription:

                full\_transcription.append(transcription)

        elif choice == '2':

            if not full\_transcription:

                print("No transcription available. Please start transcription first.")

            else:

                full\_text = " ".join(full\_transcription)

                summary = summarize\_text(full\_text)

                print("\n--- Full Summary ---")

                print(summary)

                text\_to\_speech(summary)

        elif choice == '3':

            print("Exiting the program. Goodbye!")

            break

        else:

            print("Invalid choice. Please try again.")

**Results**

The tool performed as expected, with the following observations:

1. Speech-to-Text Conversion:
   * Accurate transcription for clear speech with minimal background noise.
   * Some errors occurred in cases of unclear diction or excessive noise.
2. Text Summarization:
   * Produced coherent summaries for small-to-moderate-sized transcriptions.
   * Longer transcripts (exceeding 1000 characters) required splitting or truncation for accurate summarization.
3. Text-to-Speech Conversion:
   * The audio output was clear, but it was dependent on the system's audio player compatibility.

**A black background with white text

Description automatically generated**

**Conclusion**

The meeting transcription and summary tool successfully integrated speech-to-text, summarization, and text-to-speech functionalities. This tool provides a practical solution for summarizing meeting conversations and can be further enhanced for improved performance and user experience.