**End-to-End AI Voice Assistance Pipeline**

**Introduction**

This document outlines the design and implementation of an End-to-End AI Voice Assistance

Pipeline. The pipeline converts a voice query into text, processes it using a Large Language Model

(LLM), and then converts the generated response back into speech. The system is designed to be

low latency, includes Voice Activity Detection (VAD), restricts the response to 2 sentences, and

allows for tunable parameters such as pitch, male/female voice, and speed.

**Step 1: Voice-to-Text Conversion**

Model Used: Whisper by OpenAI (en-US model)

Libraries: whisper, pydub, webrtcvad

Implementation Details:

- Sampling Rate: 16 kHz

- Audio Channel Count: 1 (Mono)

- VAD Threshold: 0.5

I had used Whisper for converting voice input into text due to its high accuracy and robustness in

various acoustic environments. The model is configured with the following parameters to handle real-time voice queries.

**Code:-**

import whisper

import webrtcvad

import pydub

from pydub import AudioSegment

model = whisper.load\_model("base.en")

audio = AudioSegment.from\_file("input\_audio.wav", format="wav")

audio = audio.set\_channels(1).set\_frame\_rate(16000)

vad = webrtcvad.Vad(0.5)

frames = pydub.utils.make\_chunks(audio, 30)

speech\_frames = [frame for frame in frames if vad.is\_speech(frame.raw\_data, 16000)]

text = model.transcribe(speech\_frames)

**Step 2: Text Input into LLM**

Model Used: Llama2 via Hugging Face Transformers

Libraries: transformers

Implementation Details:

- Pre-trained Model: Llama2 or Mistral depending on the specific application requirements.

- Text Input: The text generated from the Speech-to-Text step is fed into the LLM to generate a

response.

The LLM chosen is based on the application's need for a balance between speed and

comprehension. Llama2 is suitable for generating concise responses, which aligns with the

requirement of restricting output to 2 sentences.

**Code :**

from transformers import AutoModelForCausalLM, AutoTokenizer

tokenizer = AutoTokenizer.from\_pretrained("huggingface/llama2")

model = AutoModelForCausalLM.from\_pretrained("huggingface/llama2")

inputs = tokenizer(text['text'], return\_tensors="pt")

response = model.generate(inputs.input\_ids, max\_length=50)

output\_text = tokenizer.decode(response[0], skip\_special\_tokens=True).split('.')[0:2] # Restrict to 2

sentences

**Step 3: Text-to-Speech Conversion**

Model Used: SpeechT5 from Microsoft

Libraries: edge-tts, transformers

Implementation Details:

- Output Format: .mp3 or .wav

- Tunable Parameters: Pitch, Male/Female Voice, Speed

Text-to-Speech conversion is handled using Microsoft's SpeechT5 model, which provides

high-quality voice synthesis. The model allows for adjustments in pitch, gender, and speed, which

are crucial for personalized voice assistance.

**Code :**

from edge\_tts import TTS, VoicesManager

tts = TTS(voice="en-US-JoannaNeural", pitch="high", rate="1.25")

audio\_output = tts.speak(output\_text, "output\_audio.mp3")

Additional Requirements

1. Latency Optimization:

- Implement WebRTC (WRTC) to minimize latency below 500ms.

- Optimize model loading and processing times by using quantized models or faster inference

frameworks like ONNX.

- Use asynchronous processing to handle multiple steps in parallel.

2. Voice Activity Detection (VAD):

- Implemented using webrtcvad library, which effectively filters out silence and non-speech

sounds.

- Adjust the sensitivity threshold to 0.5 for optimal detection.

3. Output Restriction:

- The LLM response is restricted to a maximum of 2 sentences using token or sentence-level

truncation.

4. Tunable Parameters:

- Pitch: Adjust pitch using the edge-tts parameters.

- Male/Female Voice: Select different voice profiles (e.g., Joanna for female, Matthew for male).

- Speed: Control the rate of speech using the TTS engine?s rate parameter.

**Conclusion**

This document outlines the implementation of an End-to-End AI Voice Assistance Pipeline using

open-source models and libraries. The system is designed to be low latency, highly customizable,

and effective in real-time voice assistance scenarios. The provided code snippets demonstrate the

core components of the pipeline, which can be further expanded and optimized depending on

specific use cases.