# **LiveKit Voice Pipeline with Audio Length Validation**

## **Objective**

This task is to build a voice assistant using **LiveKit's Voice Pipeline Agent**. The implementation requires validating audio length before converting text to speech. If the audio length exceeds 60 seconds, the server will return a trimmed version.

## **Table of Contents**

Initial Setup

Voice Assistant Pipeline Implementation

Audio Length Validation Server

Running the Flask Server

## **1. Initial Setup**

Start by setting up the environment and dependencies:

### **Steps:**

1. **Create a virtual environment** and activate it:

python -m venv venv  
source venv/bin/activate # or venv\Scripts\activate on Windows

1. **Install dependencies**:

pip install flask flask-cors python-dotenv livekit deepgram openai silero

1. **Set environment variables** in .env for API keys and settings.

## **2. Voice Assistant Pipeline Implementation**

### **Key components:**

1. **Imports and Environment Setup**

import asyncio  
import logging  
import requests  
from dotenv import load\_dotenv  
from livekit import rtc  
from livekit.agents import JobContext, JobProcess  
from livekit.agents.pipeline import VoicePipelineAgent  
from livekit.plugins import deepgram, openai, silero  
load\_dotenv()

1. **Prewarm Function** - Preload voice activity detection (VAD):

def prewarm(proc: JobProcess):  
 proc.userdata["vad"] = silero.VAD.load()

1. **Entry Point** - Initializes assistant context:

async def entrypoint(ctx: JobContext):  
 chat\_ctx = llm.ChatContext().append(  
 role="system",  
 text="You are a voice assistant providing concise responses."  
 )

1. **Audio Validation** - Call server to check and trim audio:

def before\_tts\_callback(assistant, text: str) -> str:  
 response = requests.post('https://your-server-url/validate\_audio\_length', json={'text': text})  
 return response.json().get('validated\_text', text)

1. **VoicePipelineAgent** - Setting up the pipeline:

agent = VoicePipelineAgent(  
 vad=ctx.proc.userdata["vad"],  
 stt=deepgram.STT(model="nova-3-general"),  
 llm=openai.LLM(),  
 tts=openai.TTS(),  
 chat\_ctx=chat\_ctx,  
 before\_tts\_cb=before\_tts\_callback  
)

## **3. Audio Length Validation Server**

The **Flask server** validates audio length and trims text when necessary.

### **Core Functions:**

1. **Flask Setup**:

from flask import Flask, request, jsonify  
from flask\_cors import CORS  
app = Flask(\_\_name\_\_)  
CORS(app)

1. **Helper Functions**:
   1. estimate\_audio\_length: Estimates based on word count.
   2. trim\_text: Trims text to fit 60 seconds.

def estimate\_audio\_length(text):  
 words = text.split()  
 return len(words) / 2 # assuming 2 words per second  
  
def trim\_text(text, max\_duration=60):  
 words = text.split()  
 max\_words = max\_duration \* 2  
 return ' '.join(words[(len(words) - max\_words)//2:(len(words) + max\_words)//2])

1. **Validation Endpoint**:

@app.route('/validate\_audio\_length', methods=['POST'])  
def validate\_audio():  
 text = request.json['text']  
 audio\_length = estimate\_audio\_length(text)  
 if audio\_length > 60:  
 text = trim\_text(text)  
 return jsonify({'validated\_text': text})

## **Run the Server**

python app.py

Ngrok server

Minimal\_assistant.py