**Libraries used**

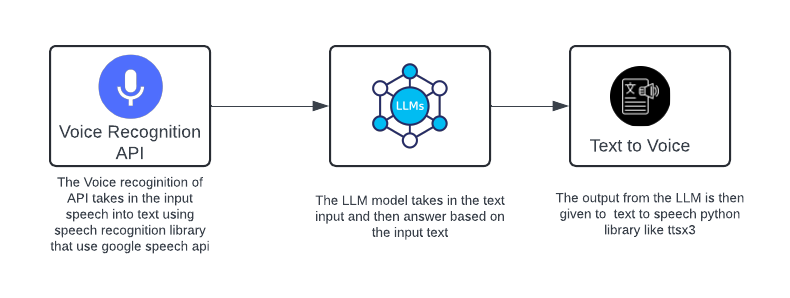
PyAudio, SpeechRecognition, pydub, langchain, langchain-google-genai, python-dotenv, pyttsx3, streamlit,numpy.

Models:

* Speech to Text:Speech recognition library containing Google Speech to text API
* Input Text to Model Reply:Gemini-1.5-pro model using API.
* Model Reply to Speech: Have used the Python library ttsx3 for text to speech with custom input from the user like pitch,speed and male or female audio

Parameter:

* Sampling Rate: 16k
* Output audio parameters: are based on user choice in Streamlit Page

**Working of the entire architecture**

Step 1:

Speech to text:

It involves using Speech recognition library which involves Google voice API and convert the speech to text

Step 2:

Text to LLM:

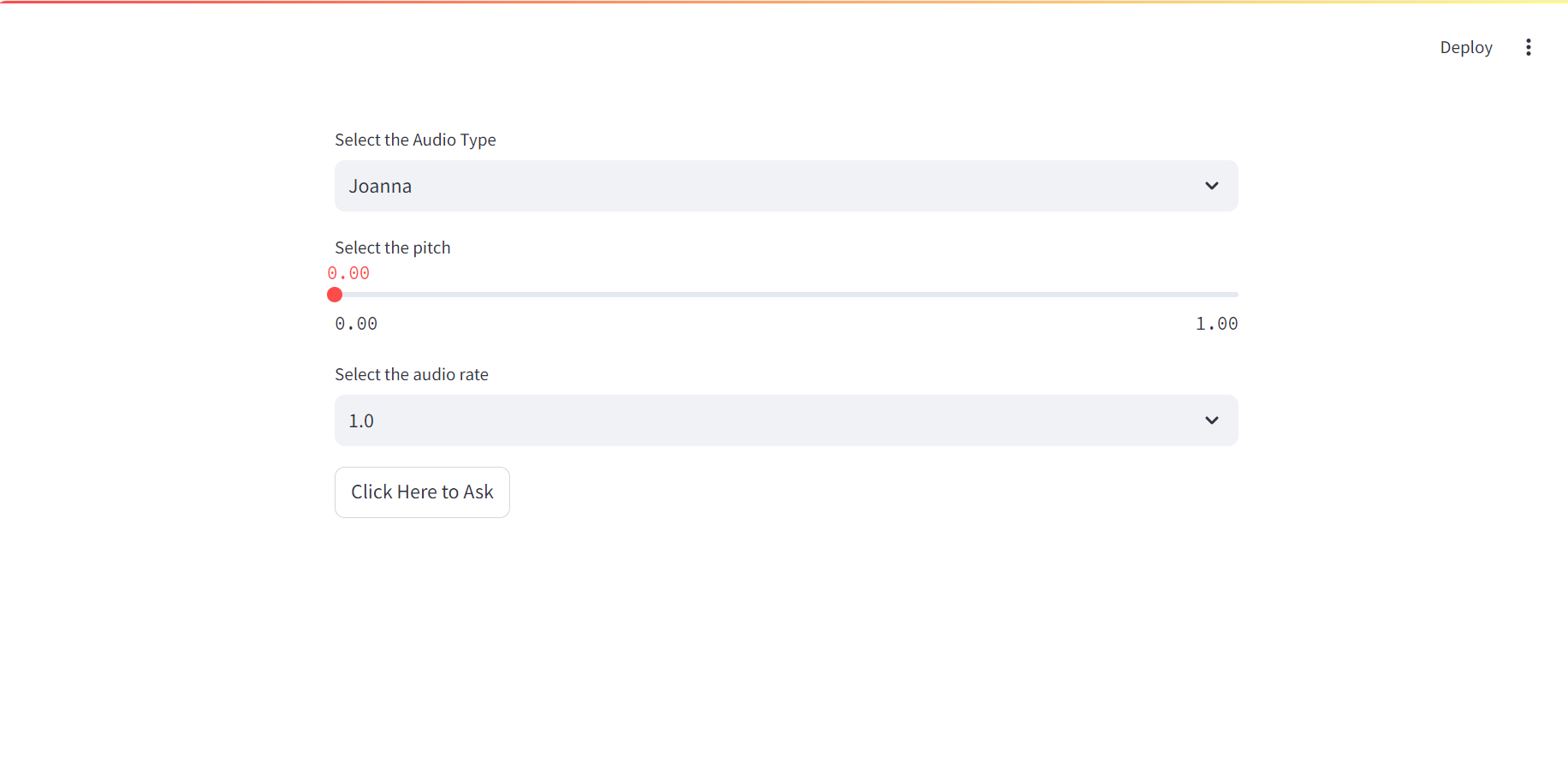
The text is then passed through the LLM here the LLM used is Gemini using the Google API with a prompt using Langchain prompt template to output the result in 2 sentences.

Step 3:

LLM output to Speech:

The output from the LLM is then passed through the python library ttsx3 to convert the LLM output to speech .The speech can be customized by changing the pitch ,voice of male or female and pitch of voice by making changes in properties of the ttsx3 .

User Interface to adjust the speech output is as below record voice is as below:



Code Snippets:

**Function Python modules:** contains functions that called in streamlit page

import speech\_recognition as sr

from langchain.prompts import PromptTemplate

from langchain\_google\_genai import ChatGoogleGenerativeAI

from dotenv import load\_dotenv

import pyttsx3 as tt

load\_dotenv()

# This function return the output from the voice text by using llm

def model\_output(text):

    model = ChatGoogleGenerativeAI(model='gemini-1.5-pro')

# Creating prompt to answer in two sentence

    template = '''answer the following  in 2 sentences:

    {user\_input}'''

    prompt = PromptTemplate(template=template, input\_variables=['user\_input'])

    formatted\_prompt = prompt.format(user\_input=text)

    print(formatted\_prompt)

    result = model.invoke(formatted\_prompt).content

    return result

# this fuction takes in audio input of user

def detect\_input():

    try:

        # Initialize recognizer class (for recognizing the speech)

        r = sr.Recognizer()

        # Use the microphone as the audio source

        with sr.Microphone(sample\_rate=16000) as source:

            print("Say something!")

            try:

                # Listen for the first phrase and extract it into audio data

                audio\_data = r.listen(source, timeout=2)

                # Recognize speech using Google Speech Recognition

                text = r.recognize\_google(audio\_data)

                print("You said: " + text)

                return text

            except sr.UnknownValueError:

                print("Your audio wasn't clear")

            except sr.WaitTimeoutError:

                print("Please speak something, listening timed out")

    except OSError:

        print('Connect your input Device')

# assigning the respective voice and audio rate

def output\_audio(result,speed,voice):

    speaker = tt.init()

    rate = speaker.getProperty('rate')

    speaker.setProperty('rate', speed)

    voices=speaker.getProperty('voices')

    # select voice based on user choice

    if voice=='Male':

        speaker.setProperty('voice',voices[0].id)

    else:

        speaker.setProperty('voice',voices[1].id)

    # Speak the result

    speaker.say(result)

    # Run the speech engine to actually speak the text

    speaker.runAndWait()

**Streamlit Page** : where the streamlit page runs and uses the function from Function python module

import streamlit as st

import numpy as np

from voice\_to\_audio import detect\_input, model\_output, output\_audio

audio\_rate = [x \* 0.25 for x in range(0, 13)]

#For selection the type of voice

voice\_gender = st.selectbox('Select the Audio Type', ('Male', 'Female'))

# selecting the speed for the audio

default\_index = audio\_rate.index(1.0)

speed = st.selectbox('Select the audio rate', tuple(audio\_rate), index=default\_index)

if st.button('Click Here to Ask'):

    with st.spinner('Hearing'):

        text = detect\_input()

        result = model\_output(text=text)

        # The audio rate start from 50 here as 50 refer to 0.25 speed and we have the values till 400 as it refers to 2 x speed

        # as audio rate ranges from 0.25 to 2 in put so we multiple with 200.

        # eg: for we can want the audio to be at speed 1X the its converted for pyttsx3 we multiple 1 \* 200 to get 1X speed in it.

        output\_audio(result=result, speed=speed\*200, voice=voice\_gender)