**CONVERSATIONAL AI**

**Project 1**

**Name:** Ramya Yalamarthi **Mail-id:** ramyayalamarthi24@gmail.com

**GCP Project Name:** [**My Project Trial 32055**](https://console.cloud.google.com/home/dashboard?authuser=1&cloudshell=true&inv=1&invt=Abnuyg&project=my-project-trial-32055)  
**GCP Project ID: my-project-trial-32055**

**Cloud Run Service name:** [convai-project1](https://console.cloud.google.com/run/detail/us-east1/convai-project1?authuser=1&cloudshell=true&inv=1&invt=Abnuyg&project=my-project-trial-32055)

(https://console.cloud.google.com/run/detail/us-east1/convai-project1/metrics?authuser=1&cloudshell=true&inv=1&invt=Abnuyg&project=my-project-trial-32055)  
  
  
**App URL:** <https://convai-project1-186442252159.us-east1.run.app>

**Code Repository:** (https://console.cloud.google.com/storage/browser/projectreport2222/project-1%20?pageState=(%22StorageObjectListTable%22:(%22f%22:%22%255B%255D%22))&authuser=1&invt=AbnvLg&project=my-project-trial-32055)

**Introduction**

The primary goal of this project is to use Google Cloud APIs to develop text-to-speech (TTS) and speech-to-text (STT) features in a simple application. The major objective is to use Google's sophisticated machine learning and artificial intelligence (AI) capabilities to accurately transcribe spoken words into text and to transform written text into voice that sounds natural. The project is to investigate the capabilities and operation of Google Cloud's "Cloud Text-to-Speech" and "Cloud Speech-to-Text" APIs by integrating the services of the Google cloud platform. Real-time transcription of recorded audio to text and smooth real-time translation of input text to speech are two of the key characteristics.

**Architecture**

**User Interface**

- (HTML)  
-(Audio/Text) Input

**Google Cloud Platform**

-Text-to-Speech API  
-Speech-to-Text API

**Storage**

- text files, audio files

**Application Backend**

- Flask Application  
- Request handling

**User Interface:** This is a typical HTML landing page serves as the project application's user interface, and it uses GET and POST requests to upload and retrieve data. A flask application on the backend is used to render this HTML page. Users can enter audio to be converted to text using the recorder, and they can enter text to be synthesized and converted to audio using the text box. The output is produced, correspondingly, beneath the input blocks. To view the text that has been transcribed from their audio, users can click on the reference link below and listen to the audio for the synthesized text input by playing the audio files.

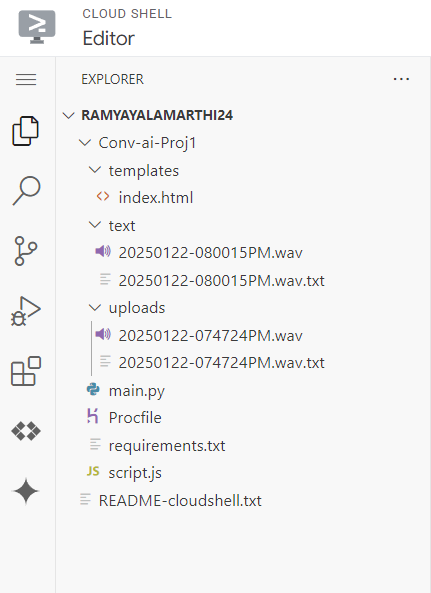
**Application Backend:** The application's backend is a built-in Python programming framework called Flask. It assists in managing requests from the application's front end, in this case the HTML page. All of the code and libraries needed to implement and integrate the data flow between the user interface and the APIs are included in the Flask application. In order to obtain and upload data to and from directory folders or to initiate the Google Cloud APIs for the purpose of synthesizing or transcribing the input data, the backend application intercepts the user's GET or POST requests.

**Google Cloud Platform:** In order to synthesis audio from text or transcribe text from audio, respectively, the backend application sends triggers to the Text-to-Speech and Speech-to-Text API services in the GCP.

**Storage:** Text or audio files are used to store the data for this application.

**Implementation**

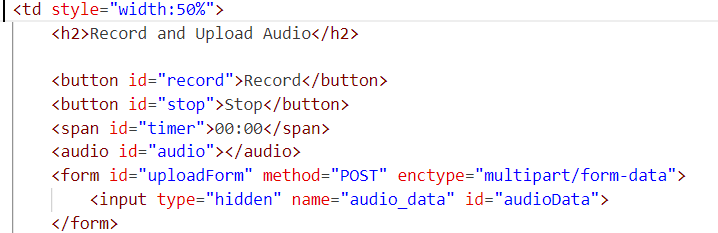
**Project Structure:**

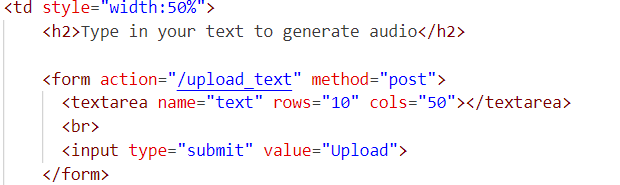
****

**User Interface**:

1. index.html – A typical HTML landing page with two data entry options and output displays beneath them. Using GET and POST requests to the backend, a text box form is used to upload text data, and a record and stop button is used to enter audio.

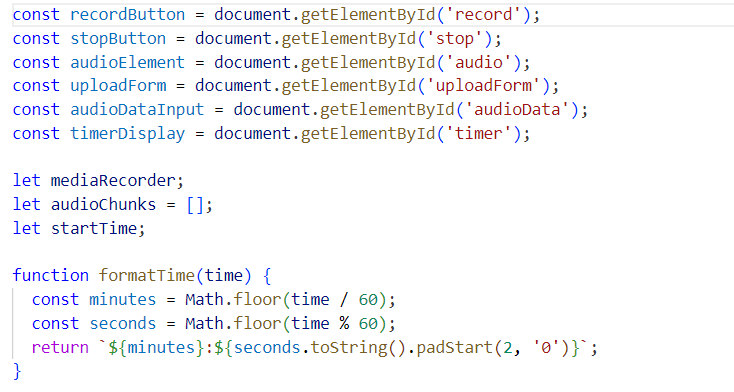
Code for inputs:





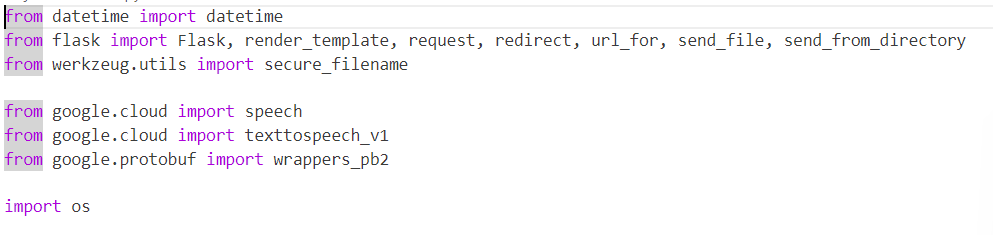
1. Script.js – The user interface's record, stop buttons are enabled via a JavaScript file that is linked to the HTML file.

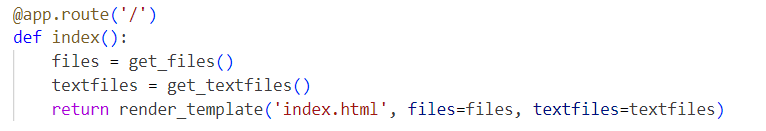
Code in script.js to access the buttons in html file:



**Application backend:**

1. main.py - The is the main file of the project. It manages the website's front end's GET and POST requests.

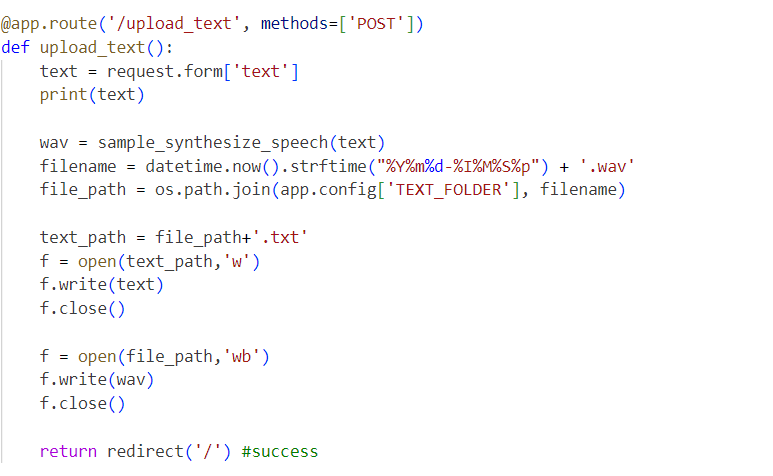
Imported Libraries for the Application code:



The following code handles the speech-to-text feature of the application. It gets the post request from the html page with the input audio. This input audio will be sent to the google clouds speech-to-text API to generate the output text. Both the input audio and output text are saved in the uploads folder in the project directory.



The following code handles the text-to-speech feature of the application. It gets the post request from the html page with the input text. This input text will be sent to the google clouds text-to-speech API to generate the output audio. Both the input text and output audio are saved in the text folder in the project directory.



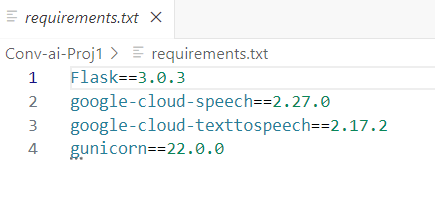
Code for text-to-speech and speech-to-text API clients :



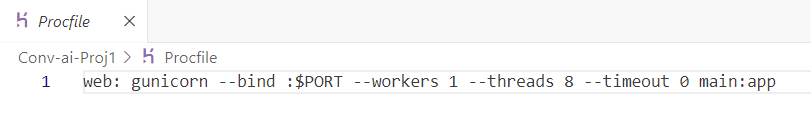


**Configuration:**

1. Requirements.txt – This file contains the dependencies for the project. These dependencies are installed when the application is hosted in the google cloud service.



1. Procfile – This file has the commands to run when the application starts.



**Pros and Cons**

**Pros**: It is a simple implementation of text-to-speech and speech-to-text API features using a flask application. It is easy to use and navigate the application to synthesize and generate speech and text respectively.

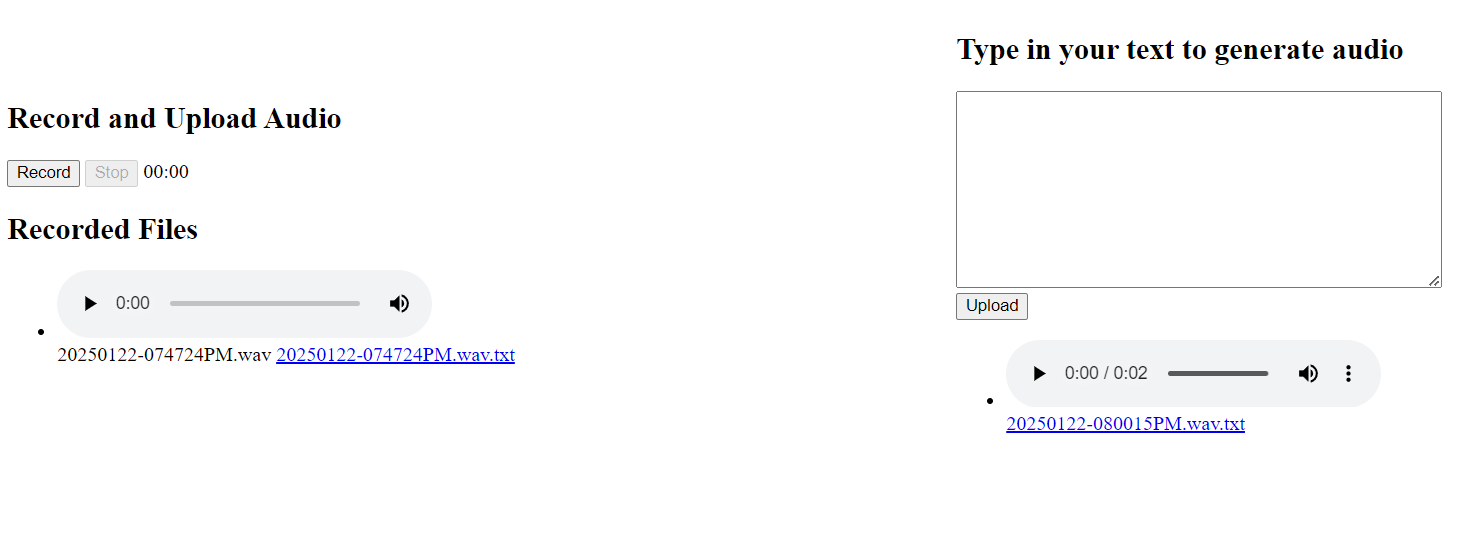
**Cons:** This is a basic application with no dedicated database. The output files are stored in the project directory. So it will be a problem to scale the application when dealing with large amounts of files.

**Problems encountered and Solutions**

I have encountered multiple issues when deploying the application into the google cloud service. The build and revision to deploy were failing continuously. This issue got resolved after correcting the project dependencies in the requirements.txt file.

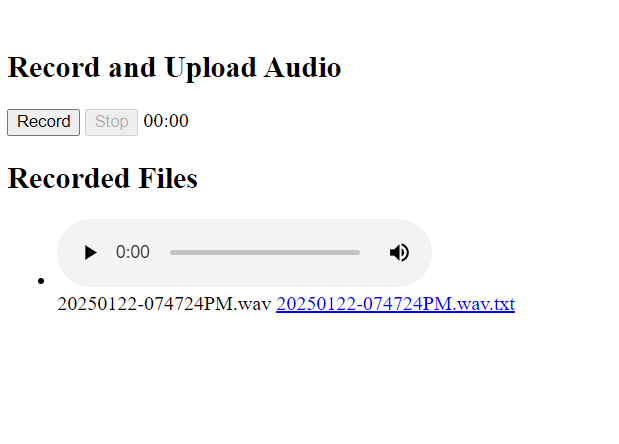
**Application instructions**

Landing page:

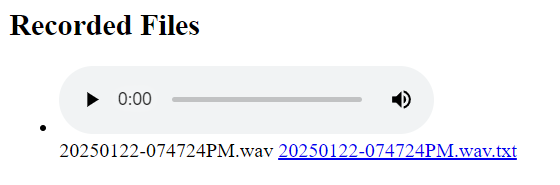


Speech-to-text implementation:

* Click on the record button to record the audio and hit stop when the input is delivered

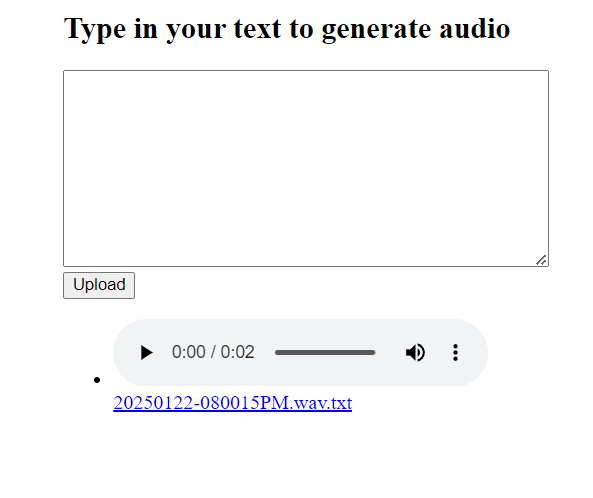


* Once the stop button is clicked the audio is transcribed using the speech-to-text API in the backend and audio and text output is generated below.
* Click on the link generated below in the recorded files to see the generated text output in a new tab.

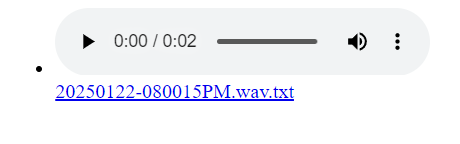


Text-to-Speech Implementation:

* Type the input text in the text box form and hit upload
* Once the upload is clicked the text-to-speech API is used to synthesize the output audio and is displayed below along with its input. Click show input to see the input for that audio.



* Click to play the output audio file generated.



**Lessons learned**

* How to build a basic flask application using python flask framework.
* Integrate google cloud text-to-speech and speech-to-text APIs into a basic flask application.
* Deploy and host an application in google cloud platform using google cloud run.

**Appendix**

**main.py :**

**from datetime import datetime**

**from flask import Flask, render\_template, request, redirect, url\_for, send\_file, send\_from\_directory**

**from werkzeug.utils import secure\_filename**

**from google.cloud import speech**

**from google.cloud import texttospeech\_v1**

**from google.protobuf import wrappers\_pb2**

**import os**

**client1 = speech.SpeechClient()**

**client2 = texttospeech\_v1.TextToSpeechClient()**

**def sample\_recognize(content):**

**audio=speech.RecognitionAudio(content=content)**

**config=speech.RecognitionConfig(**

**# encoding=speech.RecognitionConfig.AudioEncoding.MP3,**

**# sample\_rate\_hertz=24000,**

**language\_code="en-US",**

**model="latest\_long",**

**audio\_channel\_count=1,**

**enable\_word\_confidence=True,**

**enable\_word\_time\_offsets=True,**

**)**

**operation=client1.long\_running\_recognize(config=config, audio=audio)**

**response=operation.result(timeout=90)**

**txt = ''**

**for result in response.results:**

**txt = txt + result.alternatives[0].transcript + '\n'**

**return txt**

**def sample\_synthesize\_speech(text=None, ssml=None):**

**input = texttospeech\_v1.SynthesisInput()**

**if ssml:**

**input.ssml = ssml**

**else:**

**input.text = text**

**voice = texttospeech\_v1.VoiceSelectionParams()**

**voice.language\_code = "en-UK"**

**# voice.ssml\_gender = "MALE"**

**audio\_config = texttospeech\_v1.AudioConfig()**

**audio\_config.audio\_encoding = "LINEAR16"**

**request = texttospeech\_v1.SynthesizeSpeechRequest(**

**input=input,**

**voice=voice,**

**audio\_config=audio\_config,**

**)**

**response = client2.synthesize\_speech(request=request)**

**return response.audio\_content**

**app = Flask('\_name\_')**

**# Configure upload folder**

**UPLOAD\_FOLDER = 'uploads'**

**ALLOWED\_EXTENSIONS = {'wav'}**

**app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER**

**os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)**

**# Configure text folder**

**TEXT\_FOLDER = 'text'**

**ALLOWED\_EXTENSIONS = {'wav'}**

**app.config['TEXT\_FOLDER'] = TEXT\_FOLDER**

**os.makedirs(TEXT\_FOLDER, exist\_ok=True)**

**def allowed\_file(filename):**

**return '.' in filename and \**

**filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS**

**def get\_files():**

**files = []**

**for filename in os.listdir(UPLOAD\_FOLDER):**

**if allowed\_file(filename):**

**files.append(filename)**

**print(filename)**

**files.sort(reverse=True)**

**return files**

**def get\_textfiles():**

**files = []**

**for filename in os.listdir(TEXT\_FOLDER):**

**if allowed\_file(filename):**

**files.append(filename)**

**print(filename)**

**files.sort(reverse=True)**

**return files**

**@app.route('/')**

**def index():**

**files = get\_files()**

**textfiles = get\_textfiles()**

**return render\_template('index.html', files=files, textfiles=textfiles)**

**@app.route('/upload', methods=['POST'])**

**def upload\_audio():**

**if 'audio\_data' not in request.files:**

**flash('No audio data')**

**return redirect(request.url)**

**file = request.files['audio\_data']**

**if file.filename == '':**

**flash('No selected file')**

**return redirect(request.url)**

**if file:**

**# filename = secure\_filename(file.filename)**

**filename = datetime.now().strftime("%Y%m%d-%I%M%S%p") + '.wav'**

**file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], filename)**

**file.save(file\_path)**

**f = open(file\_path,'rb')**

**data = f.read()**

**f.close()**

**text = sample\_recognize(data)**

**text\_path = file\_path+'.txt'**

**f = open(text\_path,'w')**

**f.write(text)**

**f.close()**

**return redirect('/') #success**

**@app.route('/upload/<filename>')**

**def get\_file(filename):**

**return send\_file(filename)**

**@app.route('/upload\_text', methods=['POST'])**

**def upload\_text():**

**text = request.form['text']**

**print(text)**

**wav = sample\_synthesize\_speech(text)**

**filename = datetime.now().strftime("%Y%m%d-%I%M%S%p") + '.wav'**

**file\_path = os.path.join(app.config['TEXT\_FOLDER'], filename)**

**text\_path = file\_path+'.txt'**

**f = open(text\_path,'w')**

**f.write(text)**

**f.close()**

**f = open(file\_path,'wb')**

**f.write(wav)**

**f.close()**

**return redirect('/') #success**

**@app.route('/script.js',methods=['GET'])**

**def scripts\_js():**

**return send\_file('./script.js')**

**@app.route('/text/<filename>')**

**def text\_file(filename):**

**return send\_from\_directory(app.config['TEXT\_FOLDER'], filename)**

**@app.route('/uploads/<filename>')**

**def uploaded\_file(filename):**

**return send\_from\_directory(app.config['UPLOAD\_FOLDER'], filename)**

**if \_\_name\_\_ == '\_\_main\_\_':**

**app.run(debug=True)**

**Index.html**

<!DOCTYPE html>

<html>

<head>

    <title>Audio Recorder</title>

</head>

<body>

    <table style="width:100%">

        <tr>

            <td style="width:50%">

                <h2>Record and Upload Audio</h2>

                <button id="record">Record</button>

                <button id="stop">Stop</button>

                <span id="timer">00:00</span>

                <audio id="audio"></audio>

                <form id="uploadForm" method="POST" enctype="multipart/form-data">

                    <input type="hidden" name="audio\_data" id="audioData">

                </form>

                <script src="script.js"></script>

                <hl></hl>

                <h2>Recorded Files</h2>

                <ul>

                    {% for file in files %}

                    <li>

                        <audio controls>

                            <source src="{{ url\_for('uploaded\_file', filename=file) }}">

                            Your browser does not support the audio element.

                        </audio><br>

                        {{file}}

                        <a href="uploads/{{file}}.txt">{{file}}.txt</a>

                    </li>

                    {% endfor %}

                </ul>

            </td>

            <td style="width:50%">

                <h2>Type in your text to generate audio</h2>

                <form action="/upload\_text" method="post">

                  <textarea name="text" rows="10" cols="50"></textarea>

                  <br>

                  <input type="submit" value="Upload">

                </form>

                <ul>

                    {% for textfile in textfiles %}

                    <li>

                        <audio controls>

                            <source src="{{ url\_for('text\_file', filename=textfile) }}">

                            Your browser does not support the audio element.

                        </audio><br>

                        {{file}}

                        <a href="text/{{textfile}}.txt">{{textfile}}.txt</a>

                    </li>

                    {% endfor %}

                </ul>

            </td>

        </tr>

    </table>

</body>

</html>

**Script.js :**

const recordButton = document.getElementById('record');

const stopButton = document.getElementById('stop');

const audioElement = document.getElementById('audio');

const uploadForm = document.getElementById('uploadForm');

const audioDataInput = document.getElementById('audioData');

const timerDisplay = document.getElementById('timer');

let mediaRecorder;

let audioChunks = [];

let startTime;

function formatTime(time) {

  const minutes = Math.floor(time / 60);

  const seconds = Math.floor(time % 60);

  return `${minutes}:${seconds.toString().padStart(2, '0')}`;

}

recordButton.addEventListener('click', () => {

  navigator.mediaDevices.getUserMedia({ audio: true })

    .then(stream => {

      mediaRecorder = new MediaRecorder(stream);

      mediaRecorder.start();

      startTime = Date.now();

      let timerInterval = setInterval(() => {

        const elapsedTime = Math.floor((Date.now() - startTime) / 1000);

        timerDisplay.textContent = formatTime(elapsedTime);

      }, 1000);

      mediaRecorder.ondataavailable = e => {

        audioChunks.push(e.data);

      };

      mediaRecorder.onstop = () => {

        const audioBlob = new Blob(audioChunks, { type: 'audio/wav' });

        const formData = new FormData();

        formData.append('audio\_data', audioBlob, 'recorded\_audio.wav');

        fetch('/upload', {

            method: 'POST',

            body: formData

        })

        .then(response => {

            if (!response.ok) {

                throw new Error('Network response was not ok');

            }

            location.reload(); // Force refresh

            return response.text();

        })

        .then(data => {

            console.log('Audio uploaded successfully:', data);

            // Redirect to playback page or display success message

        })

        .catch(error => {

            console.error('Error uploading audio:', error);

        });

      };

    })

    .catch(error => {

      console.error('Error accessing microphone:', error);

    });

  recordButton.disabled = true;

  stopButton.disabled = false;

});

stopButton.addEventListener('click', () => {

  if (mediaRecorder) {

    mediaRecorder.stop();

  }

  recordButton.disabled = false;

  stopButton.disabled = true;

});

// Initially disable the stop button

stopButton.disabled = true;

Procfile :

web: gunicorn --bind :$PORT --workers 1 --threads 8 --timeout 0 main:app

**Requirements.text**

Flask==3.0.3

google-cloud-speech==2.27.0

google-cloud-texttospeech==2.17.2

gunicorn==22.0.0