Project - 3

Nishanth Raju
nraju2024@fau.edu

App URL: https://project-3-ks5zqe362q-ue.a.run.app/

GCP Project Name: conversational AI project
GCP Project ID: leafy-unity-436523-a9

Introduction

This project is a web-based application that allows users to interact with Google Cloud's vertexAI API. The primary goal is to provide a user interface where users can:

1. Upload audio and have it transcribed into text.
2. Implement sentiment analysis on that.

Architecture

High-Level Architecture:

- **Frontend**: A simple HTML interface to handle audio recording and uploading.
- **Backend**: A Flask application that interfaces with Google Cloud's VertexAI api for transcription generation and sentiment analysis.
- **Google Cloud Services**:
  - VertexAI api
- **File Storage**: Files (audio) are recorded and uploaded by the user to the cloud.

Implementation:

Frontend:

- The page includes a button for recording, uploading the audio file and analysis.

Backend:

- The backend is a Flask web application hosted on Google Cloud.
- The audio files are recorded and uploaded by the user from their machine to the vertexAI api.

Project - 3

Google Cloud API:

- The application communicates with Google's API via service account credentials. The application uses these credentials to authenticate and interact with the API without requiring API keys in the frontend

Pros and Cons

Pros:

- **Scalability**: The application uses Google Cloud, which means the APIs can handle substantial traffic and workloads with high reliability.
- **Integration**: Seamless integration of api within a single platform.
- **Simplicity**: The app is easy to use, with a clean UI for recording audio and generating text/audio files.

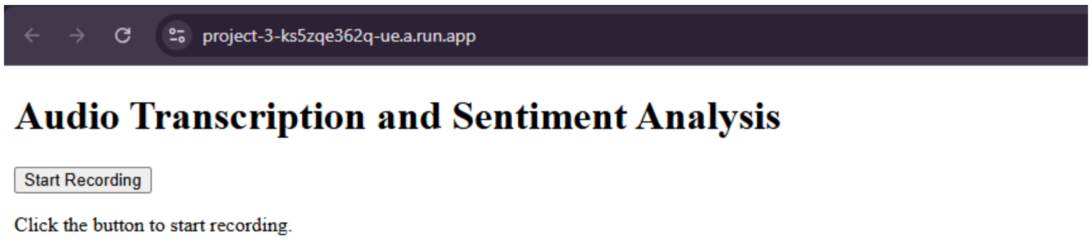
Cons:

- It is expensive to maintain over a long period of time.

Application instructions:

➢ **Uploading Audio**:

  ▪ Click on the "Start recording" button to record and upload audio file.

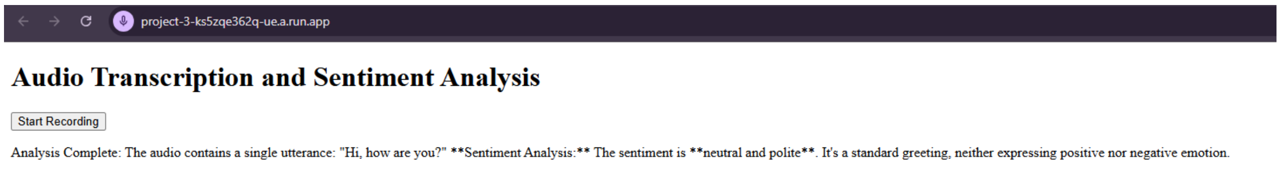

project-3-ks5zqe362q-ue.a.run.app

Audio Transcription and Sentiment Analysis

Click the button to start recording.

Project - 3

- Press the “stop recording” button and the transcription and sentiment analysis will be shown on the webpage.



project-3-ks5zqe362q-ue.a.run.app
Audio Transcription and Sentiment Analysis
Start Recording

Analysis Complete: The audio contains a single utterance: "Hi, how are you?" **Sentiment Analysis:** The sentiment is **neutral and polite**. It's a standard greeting, neither expressing positive nor negative emotion.

Lessons learned

➢ **API Integration**: learned how to integrate cloud APIs such as Google’s vertexAI api into a web application.

➢ **HTML and Flask**: I gained a deeper understanding of using html for handling client-side interactions (e.g.,uploading audio) and Flask for building the backend.

➢ **Debugging Web Applications**: I learned valuable skills in debugging python and Flask applications, particularly dealing with file uploads and API calls.

Project - 3

App.py:

```python
import os
import base64
from flask import Flask, request, Response, render_template
import vertexai
from vertexai.generative_models import GenerativeModel, Part, SafetySetting

app = Flask(__name__)

PROJECT_ID = "leafy-unity-436523-a9"
LOCATION = "us-east1"
vertexai.init(project=PROJECT_ID, location=LOCATION)

MODEL_NAME = "gemini-1.5-flash-002"
GENERATION_CONFIG = {
    "max_output_tokens": 8192,
    "temperature": 1,
    "top_p": 0.95,
}
SAFETY_SETTINGS = [
    SafetySetting(
        category=SafetySetting.HarmCategory.HARM_CATEGORY_HATE_SPEECH,
        threshold=SafetySetting.HarmBlockThreshold.OFF,
    ),
    SafetySetting(
        category=SafetySetting.HarmCategory.HARM_CATEGORY_DANGEROUS_CONTENT,
        threshold=SafetySetting.HarmBlockThreshold.OFF,
    ),
    SafetySetting(
        category=SafetySetting.HarmCategory.HARM_CATEGORY_SEXUALLY_EXPLICIT,
        threshold=SafetySetting.HarmBlockThreshold.OFF,
    ),
    SafetySetting(
        category=SafetySetting.HarmCategory.HARM_CATEGORY_HARASSMENT,
        threshold=SafetySetting.HarmBlockThreshold.OFF,
    ),
]

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/analyze-audio", methods=["POST"])
def analyze_audio():
    try:
        if "audioFile" not in request.files:
            return Response("No audio file provided", status=400,
mimetype="text/plain")
```

Project - 3

```
        audio_file = request.files["audioFile"]

        if audio_file.filename == "":
            return Response("No selected file", status=400, mimetype="text/plain")

        file_path = os.path.join("uploads", audio_file.filename)
        os.makedirs("uploads", exist_ok=True)
        audio_file.save(file_path)

        with open(file_path, "rb") as f:
            audio_base64 = base64.b64encode(f.read()).decode("utf-8")

        model = GenerativeModel(MODEL_NAME)
        audio_part = Part.from_data(
            mime_type="audio/wav", data=base64.b64decode(audio_base64)
        )
        prompt = "transcribe and provide a sentiment analysis"
        responses = model.generate_content(
            [audio_part, prompt],
            generation_config=GENERATION_CONFIG,
            safety_settings=SAFETY_SETTINGS,
            stream=True,
        )

        response_text = "".join(response.text for response in responses)
        os.remove(file_path)

        return Response(response_text, mimetype="text/plain")

    except Exception as e:
        return Response(f"Error: {e}", status=500, mimetype="text/plain")

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8080, debug=True)
```

Project - 3

Index.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Audio Analysis</title>
</head>

<body>
    <h1>Audio Transcription and Sentiment Analysis</h1>
    <button id="recordButton">Start Recording</button>
    <p id="statusMessage">Click the button to start recording.</p>

    <form id="uploadForm" action="/analyze-audio" method="post"
enctype="multipart/form-data" style="display: none;">
        <input type="file" name="audioFile" id="audioFile" hidden>
        <button type="submit">Analyze</button>
    </form>

    <script>
        let mediaRecorder;
        let audioChunks = [];

        document.getElementById("recordButton").addEventListener("click", async ()
=> {
            const recordButton = document.getElementById("recordButton");
            const statusMessage = document.getElementById("statusMessage");

            if (mediaRecorder && mediaRecorder.state === "recording") {
                mediaRecorder.stop();
                recordButton.textContent = "Start Recording";
                statusMessage.textContent = "Recording stopped. Uploading...";
            } else {
                // Request access to the microphone
                const stream = await navigator.mediaDevices.getUserMedia({ audio:
true });
                mediaRecorder = new MediaRecorder(stream);

                mediaRecorder.ondataavailable = (event) => {
                    audioChunks.push(event.data);

                    if (mediaRecorder.state === "inactive") {
                        const audioBlob = new Blob(audioChunks, { type:
"audio/wav" });
                        const formData = new FormData();
                        formData.append("audioFile", audioBlob, "recording.wav");
```

Project - 3

```
                        // Upload the audio blob via a POST request
                        fetch("/analyze-audio", {
                            method: "POST",
                            body: formData,
                        })
                            .then((response) => response.text())
                            .then((data) => {
                                statusMessage.textContent = `Analysis Complete:
${data}`;
                            })
                            .catch((error) => {
                                statusMessage.textContent = `Error:
${error.message}`;
                            });
                    }
                };

                // Start recording
                mediaRecorder.start();
                audioChunks = [];
                recordButton.textContent = "Stop Recording";
                statusMessage.textContent = "Recording in progress...";
            }
        });
    </script>
</body>

</html>
```

Requirements.txt:

```
Flask
requests
debugpy # Required for debugging.
google-cloud-speech
google-cloud-texttospeech
google-cloud-language
google-cloud-aiplatform
google-ai-generativelanguage
google-generativeai
shapely
```