EchoAI Documentation

EchoAI is a full-stack web application that processes customer feedback by analyzing its sentiment and generating an appropriate response. The project leverages Azure AI Services for sentiment analysis and text-to-speech (TTS) conversion, and uses OpenAI's GPT model for creating professional, natural language responses. The application is built using a Flask backend and a Streamlit frontend.

Overview

EchoAI is designed to provide an intelligent, responsive, and interactive experience for analyzing user feedback. Given a piece of text feedback, the application:

- Uses Azure Text Analytics to determine if the feedback is positive, neutral, or negative, along with corresponding confidence scores.
- Uses OpenAI's GPT (gpt-3.5-turbo) to generate a professional and contextual response tailored to the sentiment of the feedback.
- Converts the AI-generated text response into speech using Azure Speech Services, with voice characteristics selected according to the detected sentiment.
- Provides a user-friendly interface built with Streamlit that visualizes sentiment distributions and allows users to play the synthesized speech response.

Technologies Used

1. **Backend:** Flask, Python
2. **Frontend:** Streamlit
3. **APIs and Services:**
    a. **Azure AI Services:**
        i. Text Analytics for sentiment analysis
        ii. Speech Services for text-to-speech synthesis
    b. **OpenAI API:** For generating AI-based responses

4. **Environment Management:** python-dotenv
5. **Other Libraries:** Requests, Logging, and various visualization and utility libraries.

Architecture and Components

The application comprises two main components:

1. **Flask Backend API:**
    a. **Endpoints:**
        i. analyze_sentiment (POST): Accepts feedback, performs sentiment analysis, generates an AI response, and converts it to speech.
        ii. get_audio (GET): Serves the generated audio file.
    b. **Integrations:**
        i. **Azure AI Services:** For both text analytics and speech synthesis.
        ii. **OpenAI:** For generating natural language responses.

2. **Streamlit Frontend:**
    a. **User Interface:**
        i. Accepts user input via a text area.
        ii. Displays sentiment results and AI responses.
        iii. Visualizes confidence scores with an interactive pie chart.
        iv. Provides functionality to play the synthesized audio response.
    b. **HTTP Requests:**
        i. Connects to the Flask backend to submit feedback and retrieve results.

Project Structure

```
EchoAI/
 ├ app.py                # Flask backend API
 ├ app_frontend.py       # Streamlit frontend
 ├azure_sentiment.py     # Azure sentiment analysis
 ├azure_tts.py           # Azure text-to-speech with emotion
 ├ openai_response.py    # OpenAI GPT responses
 ├config.py              # Environment variables & API creds
 ├requirement.txt        # Project dependencies
 ├README.md              # Project documentation
```

API Endpoints

1. analyze_sentiment (POST)

Purpose:

This endpoint is designed to analyze the sentiment of user-provided feedback, generate an AI-based response, and synthesize the response into speech. It leverages Azure AI Services for sentiment analysis and text-to-speech conversion, as well as the OpenAI API for generating natural language responses.

Response:

Upon successful processing, the endpoint returns a JSON response that includes:

- The overall sentiment (e.g., "positive", "neutral", or "negative"),
- Detailed confidence scores for each sentiment category, and
- The AI-generated response tailored to the analyzed sentiment.

2. get_audio (GET)

Purpose:

This endpoint is responsible for serving the audio file (typically named response.wav) that is generated by the Azure Speech Service. The file contains the speech-synthesized version of the AI response.

Response:

- If the audio file is available, the endpoint returns the file in the appropriate audio format.

Scripts

1. app.py

Role:

The app.py module hosts the Flask backend API. It defines the endpoints required for processing user feedback, including sentiment analysis and audio file delivery.

Key Function:

- **analyze_sentiment:**
  Manages the complete processing workflow by receiving user feedback, performing sentiment analysis, generating an AI response, and triggering text-to-speech conversion.
- **get_audio:**
  Facilitates the delivery of the generated audio file to the frontend.

2. app_frontend.py

Role:

The app_frontend.py module implements the Streamlit-based user interface. It serves as the interactive layer where users input their feedback and view the corresponding analysis and response.

Key Features:

- Provides an input text area for user feedback.
- Submits the feedback to the backend and retrieves sentiment analysis and AI-generated responses.
- Visualizes the sentiment confidence scores using an interactive Plotly pie chart.
- Offers functionality to play the synthesized audio response.

3. azure_sentiment.py

Role:

The azure_sentiment.py module interfaces with the Azure Text Analytics API to analyze the sentiment of the provided feedback.

Key Function:

- analyze_sentiment (feedback):
  Processes the input text to determine the overall sentiment and returns associated confidence scores.

4. azure_tts.py

Role:

This module handles the conversion of text to speech using the Azure Speech Service. It incorporates emotion-based voice selection to enhance the responsiveness of the output.

Key Function:

- text_to_speech (text, sentiment, filename="response.wav"):
  Synthesizes speech from the provided text and saves it as an audio file.
- select_voice (sentiment):
  Determines the appropriate voice for speech synthesis based on the sentiment of the feedback.

- **generate_ssml (text, sentiment):**
  Constructs the SSML (Speech Synthesis Markup Language) required for producing emotion-infused speech output.

5. openai_response.py

Role:
The openai_response.py module is responsible for generating an empathetic and concise AI response based on the analyzed sentiment and the original feedback.

Key Function:

- **get_ai_response (sentiment, feedback):**
  Constructs a prompt using the provided feedback and sentiment, and interacts with the OpenAI API to produce a tailored response.

6. config.py

Role:
The config.py module is tasked with loading environment variables and setting the necessary API credentials for both Azure and OpenAI services.

Key Function:

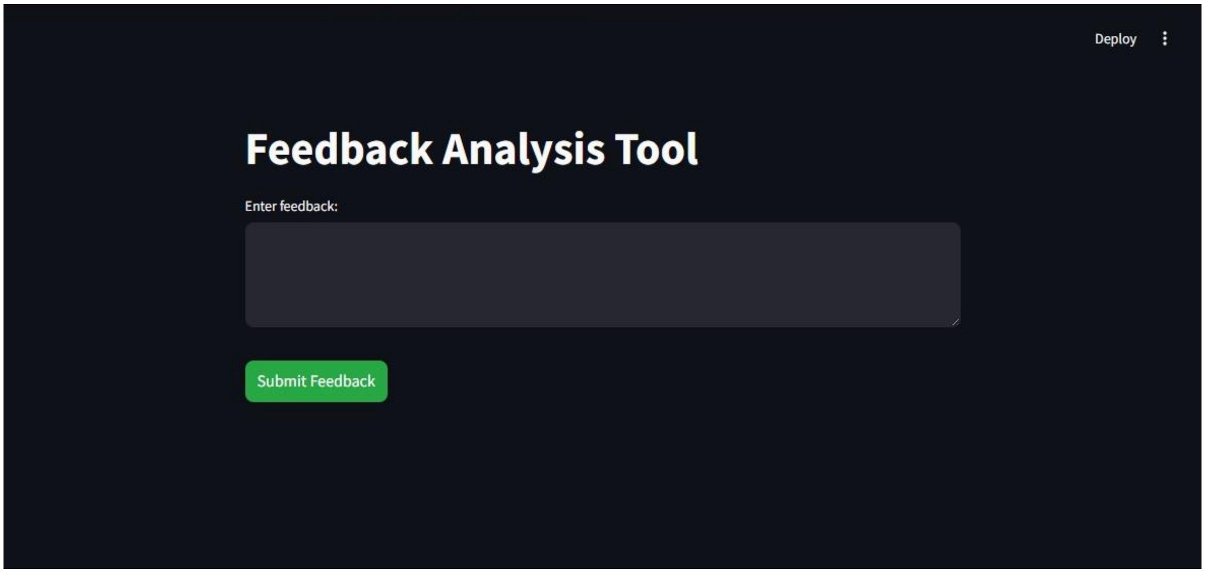
- Utilizes the python-dotenv library to manage configuration from a .env file.
- Logs detailed error messages if any required API keys or endpoints are missing, ensuring proper configuration for the application.

User Interaction with the Frontend Application

After execution of the scripts according to the ReadMe file, the Streamlit-based frontend interface will automatically launch in the user's default web browser. The interaction process is as follows:

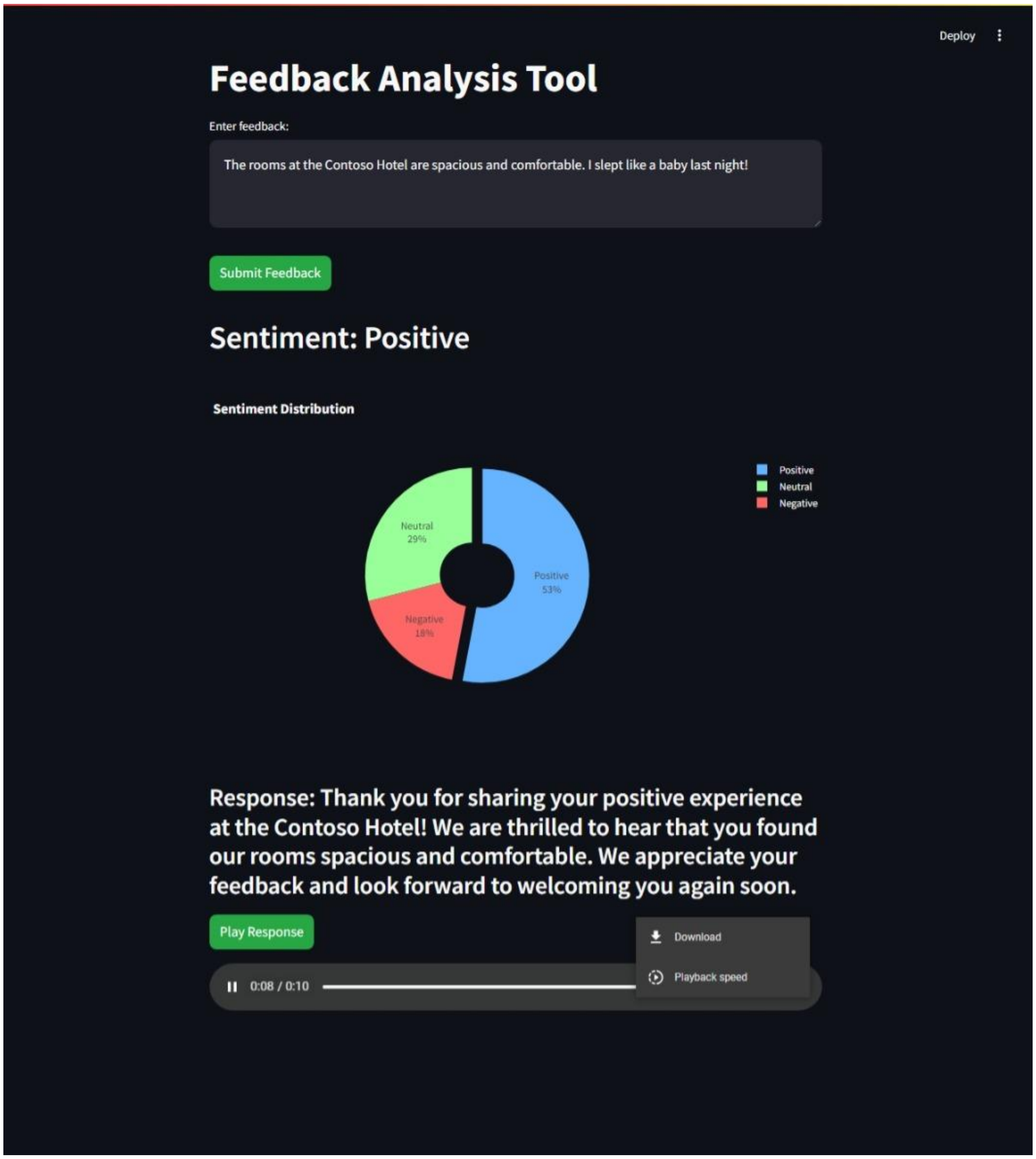
1. **Entering Feedback:** The user is prompted to enter their feedback into the designated text input field.

2. **Submitting Input:** Upon entering the feedback, the user clicks the **Submit** button to initiate processing.



3. **Sentiment Analysis Visualization:** The system analyzes the sentiment of the input text using Azure's AI services. The confidence scores for **positive**, **neutral**, and **negative** sentiments are then displayed in an interactive pie chart for transparency and interpretability.
4. **AI Response Generation:** Based on the analyzed sentiment, the OpenAI GPT model generates a relevant, context-aware response.
5. **Audio Output & Download Option:** The generated response is synthesized into speech using Azure Speech Services. The user has the option to **listen** to the AI-generated response directly within the interface or **download** the corresponding audio file for later use.

This structured workflow ensures a seamless user experience, combining sentiment analysis, intelligent response generation, and speech synthesis into a unified, interactive application.



Feedback Analysis Tool

Enter feedback:

The rooms at the Contoso Hotel are spacious and comfortable. I slept like a baby last night!

Sentiment: Positive

Sentiment Distribution

Response: Thank you for sharing your positive experience at the Contoso Hotel! We are thrilled to hear that you found our rooms spacious and comfortable. We appreciate your feedback and look forward to welcoming you again soon.