## Overview

This file, app.py, creates a Flask-based web application that functions as a conversational AI agent. It can process user input in two ways: through audio recordings or direct text input.

The application leverages several key technologies: - **Flask**: To create the web server and API endpoints. - **Google Gemini API**: To generate intelligent, human-like responses to user prompts. - **SpeechRecognition**: To transcribe spoken audio from the user into text. - **pyttsx3**: To convert the AI’s text responses back into audible speech (Text-to-Speech). - **pydub**: To handle audio file format conversion.

The core functionality involves receiving an input (audio or text), transcribing it if necessary, sending the text to the Gemini model, receiving a text response, converting that response to an audio file, and sending the audio back to the client. The application also manages temporary files created during this process, ensuring they are cleaned up after each request.

## Functions

### get\_gemini\_response(text\_prompt)

This helper function communicates with the Google Gemini API to get a generative AI response.

* **Parameters:**
  + text\_prompt (str): The text prompt to be sent to the Gemini model.
* **Returns:**
  + str: The text response generated by the Gemini model. If an error occurs during the API call, it returns a string formatted as "Error: <error\_message>".

### generate\_tts\_audio\_file(text\_to\_speak, output\_file\_path)

This helper function uses the pyttsx3 library to convert a given text string into a WAV audio file.

* **Parameters:**
  + text\_to\_speak (str): The text that needs to be converted to speech.
  + output\_file\_path (str): The full path where the generated audio file will be saved.
* **Returns:**
  + bool: True if the audio file was successfully created and is not empty, False otherwise. It will also clean up any partially created files on failure.

### index()

This is a Flask route function that serves the main user interface of the application.

* **Route:** /
* **Method:** GET
* **Returns:**
  + Renders and returns the index.html template.

### process\_full\_audio\_route()

This Flask route handles the end-to-end processing of a user’s audio input. It receives an audio file, converts it to text, gets a response from Gemini, converts that response back to audio, and returns it.

* **Route:** /process-full-audio
* **Method:** POST
* **Request Body:**
  + Expects a multipart/form-data request containing an audio file under the key audio\_blob.
* **Successful Response:**
  + **Status Code:** 200 OK
  + **Content-Type:** audio/wav
  + **Body:** The raw audio bytes of the generated speech response.
  + **Headers:**
    - X-Transcription: The URL-encoded text transcribed from the user’s input audio.
    - X-Gemini-Response: The URL-encoded text response generated by the Gemini model.
* **Error Response:**
  + **Status Code:** 400 Bad Request or 500 Internal Server Error.
  + **Body:** A JSON object with an error key, e.g., {"error": "No audio file part"}.

#### Usage Example (cURL)

curl -X POST \  
 -F "audio\_blob=@/path/to/your/audio.webm" \  
 http://127.0.0.1:5000/process-full-audio \  
 --output response.wav

### process\_text\_route()

This Flask route handles requests where the user provides text input directly. It gets a response from Gemini, converts it to an audio file, and returns the audio.

* **Route:** /process-text
* **Method:** POST
* **Request Body:**
  + Expects a JSON payload with a text key.
  + **Example:** {"text": "Hello, what is the capital of France?"}
* **Successful Response:**
  + **Status Code:** 200 OK
  + **Content-Type:** audio/wav
  + **Body:** The raw audio bytes of the generated speech response.
  + **Headers:**
    - X-Gemini-Response: The URL-encoded text response generated by the Gemini model.
* **Error Response:**
  + **Status Code:** 400 Bad Request or 500 Internal Server Error.
  + **Body:** A JSON object with an error key, e.g., {"error": "Request must be JSON"}.

#### Usage Example (cURL)

curl -X POST \  
 -H "Content-Type: application/json" \  
 -d '{"text": "Tell me a fun fact about space."}' \  
 http://127.0.0.1:5000/process-text \  
 --output response.wav