## Overview  
  
This file (`app.py`) implements a Flask web application that functions as an interactive AI voice assistant. It leverages several services to process user input, generate intelligent responses, and provide audio feedback.  
  
The core functionalities include:  
1. \*\*Audio Processing\*\*: It can receive an audio recording from a user, convert it to a standard format, and transcribe it into text using speech recognition.  
2. \*\*Text Processing\*\*: It can also accept direct text input.  
3. \*\*AI Response Generation\*\*: It sends the transcribed or submitted text to the Google Gemini Pro AI model to generate a conversational response.  
4. \*\*Text-to-Speech (TTS)\*\*: The AI-generated text response is converted back into an audio file.  
5. \*\*API Endpoints\*\*: It exposes endpoints to handle both audio and text-based interactions, returning the generated audio response.  
  
The application manages temporary files created during these processes and ensures they are cleaned up after each request.  
  
## Functions  
  
### `get\_gemini\_response(text\_prompt)`  
  
This function communicates with the Google Gemini API to get a generative AI response for a given text prompt.  
  
- \*\*Parameters\*\*:  
 - `text\_prompt` (str): The text input 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 function uses the `pyttsx3` library to convert a given string of text into a spoken audio file and save it to a specified path.  
  
- \*\*Parameters\*\*:  
 - `text\_to\_speak` (str): The text to be converted into speech.  
 - `output\_file\_path` (str): The full path where the generated audio file (in `.wav` format) should be saved.  
- \*\*Returns\*\*:  
 - `bool`: `True` if the audio file was successfully generated and is not empty. `False` if the input text is empty or if an error occurred during generation.  
  
### `index()`  
  
This is a Flask route function that serves the main user interface for the application.  
  
- \*\*Route\*\*: `/`  
- \*\*HTTP Methods\*\*: `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-submitted audio file. The workflow includes receiving the audio, transcribing it, getting an AI response, converting the response to speech, and returning the final audio file.  
  
- \*\*Route\*\*: `/process-full-audio`  
- \*\*HTTP Methods\*\*: `POST`  
- \*\*Request Body\*\*: Expects a `multipart/form-data` request with a file part named `audio\_blob`.  
- \*\*Returns\*\*:  
 - On success: A WAV audio file (`audio/wav`) as an attachment. The HTTP response also includes custom headers:  
 - `X-Transcription`: The URL-encoded text transcribed from the user's audio.  
 - `X-Gemini-Response`: The URL-encoded text response from the Gemini model.  
 - On failure: A JSON object with an `error` key and a corresponding status code (400 for bad requests, 500 for server errors).  
  
#### Usage Example:  
  
```bash  
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 the processing of a user-submitted text string. It gets an AI response from Gemini, converts it to speech, and returns the resulting audio file.

* **Route**: /process-text
* **HTTP Methods**: POST
* **Request Body**: Expects a JSON object with a text key.
* {  
   "text": "Hello, what can you do?"  
  }
* **Returns**:
  + On success: A WAV audio file (audio/wav) as an attachment. The HTTP response also includes a custom header:
    - X-Gemini-Response: The URL-encoded text response from the Gemini model.
  + On failure: A JSON object with an error key and a corresponding status code (400 for bad requests, 500 for server errors).

#### Usage Example:

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

```