Documentation

VocaSynth- A Universal Voice Companion

Overview

VocaSynth is a speech to speech application designed as a universal voice companion. It provides two primary functionalities:

1. **Speaking Bot**: Allows users to interact via speech and get responses from a generative AI model.
2. **Document Summarization**: Enables users to upload documents and receive AI-generated summaries.

The primary purpose of this application is to integrate it with other platforms to reduce the burden of maintaining customer service. Instead of relying on traditional customer care methods, businesses can deploy this application with custom language models to address user queries automatically. This eliminates the need for customer service personnel and enhances efficiency in query resolution for websites, apps, and other platforms.

Application Purpose and Integration

Purpose: The application is designed to streamline customer service operations by integrating with existing platforms, reducing the need for manual customer care services. Businesses can deploy this application to handle user queries efficiently using customized language models tailored to their specific requirements.

Integration: VocaSynth can be integrated into websites, mobile applications, and enterprise systems to provide automated query resolution. By leveraging its speech-to-speech interaction capabilities and document summarization, businesses can enhance user experience while minimizing operational costs. Custom LLMs ensure that responses are accurate and context-specific, making the application adaptable across various industries.

Advantages:

- Reduces dependency on traditional customer care services.
- Provides instant, AI-driven responses to user queries.

Features

Title Section

- Customizable title section with styled headings:
  - **Main Heading**: "VocaSynth" in bold green color.
  - **Subheading**: "A Universal Voice Companion" in gray color.
- Styling achieved using HTML and CSS embedded in the Streamlit `st.markdown` function.

Libraries Used

- **Streamlit**: For building the web app interface.
- **SpeechRecognition**: For speech-to-text functionality.
- **pyttsx3**: For text-to-speech functionality.
- **google.generativeai**: For interacting with Google's Generative AI API.
- **PyPDF2**: For extracting text from PDF files.
- **docx**: For reading Microsoft Word documents.
- **pandas**: For processing Excel files.
- **base64**: For encoding and embedding background images.

Key Functionalities

1. Speaking Bot

Description: Allows users to speak into their microphone, receive AI-generated responses, and optionally stop speech output.

Components:

- **Speech Recognition:**
  - Captures audio input using `speech_recognition` library.
  - Recognizes and transcribes spoken text.
- **Generative AI Response:**
  - Uses the Google Generative AI API (Gemini model) to process recognized speech and generate a response.
- **Text-to-Speech:**
  - Converts the AI response into speech using `pyttsx3`.
- **Chat History:**
  - Displays past interactions between the user and the bot.

- **Stop Speech Button:**
    - Allows users to stop ongoing text-to-speech playback.

Workflow:

1. User clicks the "Start Talking" button.
2. Speech is captured and transcribed.
3. Transcription is sent to the AI model for processing.
4. Generated response is displayed in the chat history and spoken aloud.
5. User can stop speech playback with the "Stop" button.

Error Handling:

- Handles errors like unrecognized speech or API request failures with appropriate messages.

---

2. Document Summarization

Description: Processes uploaded documents (PDF, Word, Excel, or text) and generates a concise summary using the generative AI model.

Supported File Types:

- **PDF**: Extracted using `PyPDF2`.
- **Word Documents**: Read using the `docx` library.
- **Excel Files**: Processed with `pandas`.
- **Text Files**: Read as plain text.

Workflow:

1. User uploads a document.
2. Content is extracted and displayed in a text area.
3. User clicks "Summarize Document" to generate a summary.
4. Summary is displayed below the text area.

Error Handling:

- Unsupported file types raise a descriptive error.
- Exceptions during file processing are caught and displayed to the user.

Background Customization

- **Feature:** Adds a background image to the app.
- **Implementation:**
  - Reads an image file.
  - Encodes the image in Base64 format.
  - Embeds the image in the app using custom CSS.
- **Function:** `add_background(image_path)`.
- **Example Usage:** Adds the background image located at "E:/Project/BG/2.png".

Google Generative AI API Integration

Configuration:

- Requires a valid API key stored in the variable `GOOGLE_API_KEY`.
- Configured using `genai.configure(api_key=GOOGLE_API_KEY)`.

Functionality:

- Processes input text using the `llm(text)` function.
- Uses the "gemini-1.5-flash" model for generating responses.
- Returns the generated response as text.

User Interface

Tabs

1. **Speaking Bot Tab:**
   - Contains buttons for starting/stopping speech recognition and chat history.

2. **Document Summarization Tab:**
   - Contains file uploader and summarization functionality.

Buttons and Actions:

- **"Start Talking":** Initiates speech recognition.
- **"Stop":** Stops text-to-speech playback.

- **"Summarize Document":** Generates a summary of the uploaded document.

Session State Management

- **chat_history:** Stores the conversation history.
- **is_speaking:** Tracks the text-to-speech playback state.

Error Handling

- Gracefully handles errors during:
  - Speech recognition.
  - File uploads and content extraction.
  - AI model processing.
- Provides user-friendly error messages.

Deployment

- Ensure all required libraries are installed.
- Replace YOUR_API_KEY with a valid Google API key.
- Run the app using:
- streamlit run app.py