LLM Copilot Project Documentation

Name: Nagendra Maddela **E-mail:** maddela3435@gmail.com

1. Introduction

1.1 Project Overview

The LLM Copilot project is an AI-powered assistant built using Streamlit and Groq API. The application allows users to interact with language models for various tasks, including generating chat responses, handling speech input, and processing media files like PDFs, audio, and videos. It features a user-friendly interface with authentication, theme customization, and feedback collection mechanisms.

1.2 Key Features

- **Authentication:** Secure login system to manage user access.
- **Theme Customization:** Users can toggle between Dark and Light themes.
- **Chat Interaction:** Communicate with the model using text or speech.
- **Media Support:** Upload and interact with media files, including PDFs, audio, and videos.
- **Feedback & Ratings:** Collect user feedback and ratings to improve the experience.

1.3 Technologies Used

- **Streamlit:** For creating the web interface.
- **Groq API:** To access and interact with the language model.
- **Python Libraries:** Various libraries like pyttsx3 for text-to-speech, speech_recognition for speech input, PyPDF2 for PDF processing, and googlesearch-python for fetching relevant links.

---

2. Setup and Configuration

2.1 Prerequisites

Before setting up the project, ensure that you have the following installed:

- **Python 3.8+**
- **pip:** Python package manager

2.2 Project Structure

```
LLM-Copilot/
│
├── config.py            # Configuration file for loading environment variables
├── main.py              # Main application code with Streamlit and Groq integration
├── requirements.txt        # List of Python dependencies
├── .env                # Environment variables, including Groq API key
└── README.md             # Project overview and instructions
```

2.3 Installation

1. **Clone the Repository:**

```
git clone https://github.com/username/LLM-Copilot.git
cd LLM-Copilot
```

2. **Install Dependencies:**

```
pip install -r requirements.txt
```

3. **Set Up Environment Variables:**
    - Create a .env file in the project root directory.
    - Add your Groq API key and other necessary configurations:

```
GROQ_API_KEY=your_groq_api_key_here
```

4. **Run the Application:**

```
streamlit run main.py
```

---

3. Application Functionality

3.1 Authentication

The application requires users to log in using a username, email, and password. Dummy credentials are used for demonstration purposes:

- **Username:** user
- **Email:** user@example.com
- **Password:** password123

Successful authentication grants access to the main application features.

3.2 Chat Interaction

Users can chat with the model by typing in the input box or using voice commands. The Groq API processes the input, and the model's response is displayed incrementally with a simulated typing effect.

3.2.1 Speech-to-Text Integration

The application uses the speech_recognition library to convert spoken words into text. Users can activate this feature through the sidebar and speak directly into their microphone.

3.3 Media Handling

The app supports the upload and processing of various media files:

- **PDFs:** Extract and display text from uploaded PDF files using PyPDF2.
- **Audio:** Play uploaded audio files directly within the app.
- **Video:** Stream uploaded video files.

3.4 Theme Customization

Users can switch between Dark and Light themes via the sidebar. The selected theme is applied throughout the application, affecting the background and text colors.

3.5 Feedback and Ratings

The sidebar includes options for users to submit feedback on the model's responses. Users can also rate responses on a scale from 1 to 5. This feedback is stored in the session state and can be used to enhance future interactions.

---

4. Usage Guide

4.1 Starting the Application

1. **Launch the Application:** After running the Streamlit server, open the app in a web browser:

   ```
   streamlit run main.py
   ```

2. **Log In:** Enter the correct username, email, and password to access the main features.

4.2 Interacting with the Model

- **Text Input:** Type your queries into the input box at the bottom of the page and press Enter. The response will appear in the conversation history.
- **Voice Input:** Use the "Start Speech Input" button in the sidebar to speak your queries. The recognized text will be displayed and processed by the model.
- **Uploading Media:** Use the file uploader to select and upload PDF, audio, or video files. The app will display or play the content accordingly.

4.3 Submitting Feedback

- **Provide Feedback:** Enter your comments in the "Provide your feedback on the response" box and click "Submit Feedback."
- **Rate Responses:** Use the rating slider to rate the assistant's performance and click "Submit Rating."

---

5. Future Enhancements

5.1 Feature Extensions

- **Advanced Analytics:** Implement user analytics to track interactions and improve model responses based on historical data.
- **Voice Output:** Enable the application to respond via voice, using pyttsx3 to read out the model's responses.

5.2 Integration with External APIs

- **Weather API:** Integrate with a weather API to provide real-time weather information.

- **Recipe API:** Fetch detailed recipes from a cooking API when users ask for culinary instructions.

---

6. Conclusion

The LLM Copilot project showcases the integration of advanced language models with a user-friendly web interface. Through the use of various Python libraries and APIs, the application offers a robust platform for users to interact with AI models in a meaningful and engaging way.