Project Challenges

During the course of this project, several challenges were encountered and addressed. Below is a detailed breakdown of these challenges, the steps taken to resolve them, and the insights gained.

1. Real-time Speech-to-Text (STT) Integration

Challenge:

A significant challenge in the project was finding a real-time Speech-to-Text API suitable for transcription purposes. Many popular transcription services, such as Google Cloud Speech-to-Text, AWS Speech-to-Text, Assembly AI, and OpenAI's Whisper, were explored. The major limitation encountered was the restrictive free-tier offerings of these APIs, which were not ideal for minimizing project costs.

Solution:

After extensive testing, the **SpeechRecognition** library was chosen due to its efficiency and cost-effectiveness. This library was well-suited for real-time transcription needs and aligned with the project's requirement to minimize costs.

2. Speed and Efficiency of Llama3.2 LLM

Challenge:

The response time and efficiency of the Llama3.2 language model (LLM) were slower than desired when answering questions. This hindered the overall user experience and performance of the application.

Solution:

To address this, prompt engineering and code optimization were undertaken to streamline the Llama3.2's performance. By refining the input structure and adjusting processing workflows, the response speed was improved. Continued optimization is necessary to further enhance this aspect.

3. Robotic AI Examiner

Challenge:

The AI examiner, which was responsible for evaluating user responses, sounded very robotic and lacked a natural conversational tone. This negatively impacted the user experience.

Solution:

Efforts were made to humanize the AI examiner's responses. Modifications to the conversational tone and the inclusion of varied sentence structures were implemented to reduce the robotic nature of the AI interactions.

4. Slow Data Retrieval from API

Challenge:

Data retrieval from the API was slower than expected, causing delays in obtaining responses for the user.

Solution:

The initial implementation utilized `fetch` for API requests, but the response times were suboptimal. After attempting to use **Axios** (a promise-based HTTP client), it was found that Axios was incompatible with Cross-Origin Resource Sharing (CORS). This led to a reversion to `fetch`, and a new real-time Speech-to-Text module was implemented, resulting in improved data retrieval speed and API interaction.

5. Real-time Speech-to-Text with React

Challenge:

Real-time Speech-to-Text (STT) integration for transcription posed difficulties, especially in aligning the transcription with the rest of the system's functionalities.

Solution:

React-Speech-to-Text was selected as the new STT module. This library proved to be effective in capturing real-time speech input and converting it into text, meeting the project's needs for real-time transcription capabilities.

6. Linking Transcription to Trained LLM

Challenge:

A final major challenge was linking the transcriptions from the frontend (via the Speech-to-Text feature) to the trained Llama3.2 LLM. This integration was key to ensuring the transcriptions were effectively used for generating responses from the LLM.

Solution:

Despite significant effort to resolve this issue, the challenge of seamlessly integrating the transcription data with the LLM could not be fully overcome within the time constraints of the project. Further development and refinement would be required to ensure smooth linkage between these components.
