Project Documentation

Workflow Description:

The application consists of two primary user flows:

User (Patient) Workflow:
1. User initiates interaction with a voice greeting.
2. User's voice input is captured and sent to the server.
3. The server transcribes the speech to text and processes it using OpenAI's Chat API.
4. Based on the AI's interpretation, appropriate database operations are performed.
5. The user receives a voice response communicating the outcome of their request.

Doctor (Admin) Workflow:
1. The doctor enters the admin interface.
2. The doctor can view, add, or remove available time slots.
3. Changes are reflected in the database and influence the AI's scheduling dialogue with users.

Design Choices:

The project employs a monolithic architecture with distinct services for voice processing, AI interaction, and database management. Here are the technologies I chose for this project:

1. FastAPI for its async support and scalability for python
2. React for a dynamic frontend and ease of integration with the backend.
3. Google Cloud's Speech-to-Text and Text-to-Speech APIs used for their accuracy and reliability.
4. Open AI’s Chat API for its excellent context based response.
5. Docker to containerize the application and allow ease of use.
6. Sqlite 3 for its easy and quick in-memory functionality.

Limitations:

- Users will interact with the system in a quiet environment conducive to voice recognition.
- Users will only use English to converse with the voicebot.
- For a working prototype solution, I have assumed that only 1 user and only 1 Doctor can use the application at a time.
- The doctor will only maintain their schedule through the provided interface, keeping it up-to-date.
- This application works only for a single Doctor’s clinic.
- Identification of patients only happens using their name.
- To reset the voice interaction context, the user needs to go back to the home page and restart.
- This bot currently supports using a gpt-4 engine/model. The application will require modifications to support lower level engines/models.
- There is a slight delay (3-4 seconds) in receiving response from the voicebot.

Challenges Faced and Resolutions:

- Designing a prompt for ChatGPT was a major challenge:
    - Relying on an AI model is a probabilistic approach and it does not guarantee 100% accuracy. So getting it close to its optimal accuracy was a challenge that I tried to resolve by using best practices of Prompt Engineering.
- Related to the above, evaluation/accuracy calculation of such an AI based system is difficult, since the metrics are not standardized yet.

Future Enhancements:

- Web App Enhancements:
    - Integration with a Real-Time Calendar: Sync with Google Calendar for live availability updates.
    - User Authentication: To personalize interactions, remember user history, and allow multi-user interaction.
    - Add the ability to ask for the user's phone number and store it in DB.
- VoiceBot:
    - Vector DB: Use a vector DB (like FAISS) to store user-specific context which will allow multiple user accessibility and to keep context length in check.
    - Finetune Chat GPT: To respond to the prompts in a format that I want.
    - User and finetune an open source language mode like Mistral, phi-2 to reduce costs and if needed host the language model on-prem to reduce OpenAI cost.
- Add more features to make it a one-stop-shop for all patient-clinic interaction.
- Include audio streaming as IO to avoid conversation delays

Cost Incurred:

- Cost incurred while making this demo was $20 for using chatGPT.