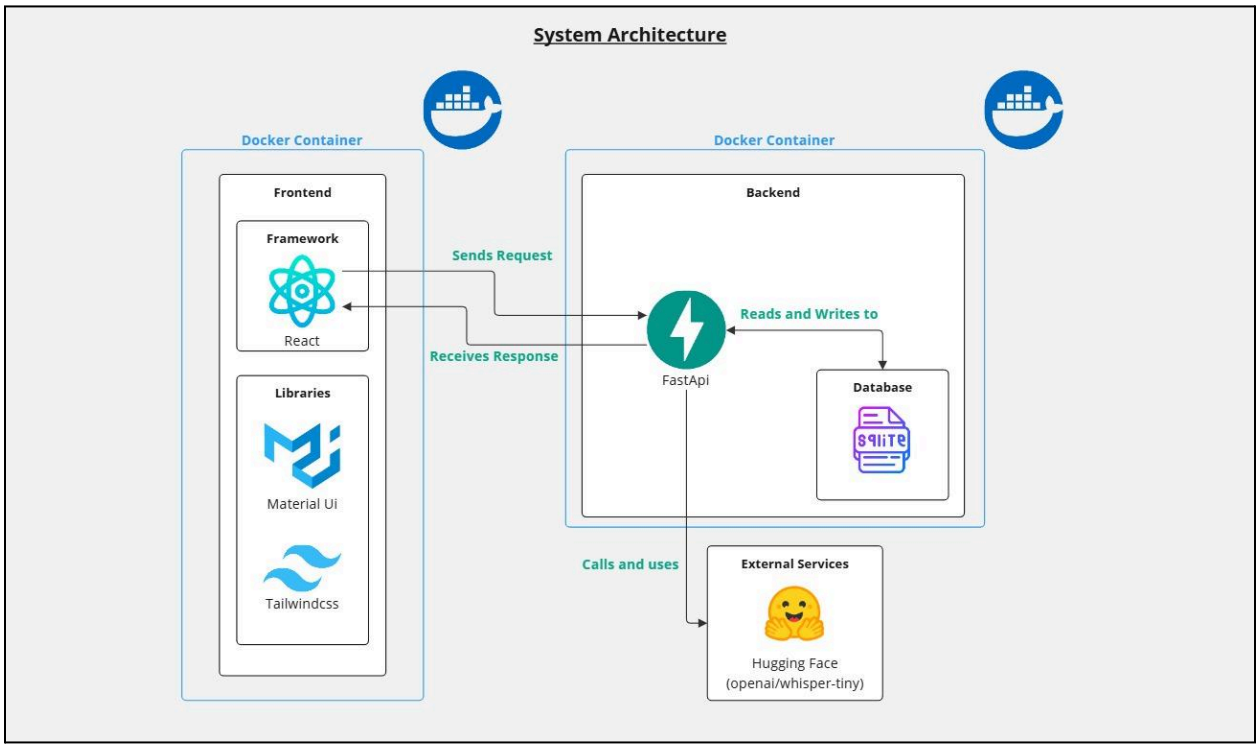
Architecture Diagram



This system architecture comprises two primary Docker containers: one hosting the frontend and one hosting the backend. The frontend is built with React and styled using Material UI and Tailwind CSS, providing a modern and responsive user interface. All user interactions are routed through the frontend, which sends requests to the FastAPI-based backend. The backend handles these requests, interacting with a SQLite database which is used for storing transcriptions. Additionally, it connects to external services on Hugging Face, specifically the `openai/whisper-tiny` model, to perform audio transcriptions.

Assumptions:

2ai) GET /health: Returns the status of the service.

- Service refers to the RESTful API itself and its critical dependencies such as the SQLite database and Whisper-tiny Model.
- Point of this endpoint is to ensure that the API can run properly.
- No frontend component needed as not mentioned in requirements.

2aii) POST /transcribe: Accepts audio files, performs transcription and save results in database.

- Based on https://help.openai.com/en/articles/7031512-whisper-audio-api-faq and audio files given, file formats accepted by the application would be [m4a, mp3, webm, mp4,

mpga, wav and mpeg] and Max file size limit is 25MB as whisper would not be able to process anything larger than that.
- Application only meant to transcribe English audio based on samples given so the model is set to English.
- Errors to be expected when a mix of languages used as Whisper can only transcribe one language at a time. E.g. “yong tau foo” transcribed as “young tofu” from Sample 3.mp3.
- English accuracy won’t always be 100% either as the model may ‘mishear’ accents.
- No need to finetune the model as it is not stated in the requirements.

2aiv) GET /search: Performs a full-text search on transcriptions based on audio file name.
- Search by file name and return the matching transcription(s).

2bii) Implement necessary audio preprocessing
- WhisperProcessor is sufficient enough as the preprocessor as Whisper functions properly with just it alone.

5) Based on the full-stack application you developed, create an architecture diagram that outlines the following components
- High level architecture diagram is expected.
- Hugging face (openai/whiper-tiny) is considered an external service.

Considerations:

2aii) POST /transcribe: Accepts audio files, performs transcription and save results in database.
- No duplicate filenames allowed to be saved to the database to prevent confusion for users.

2aiv) GET /search: Performs a full-text search on transcriptions based on audio file name.
- Partial matching for search is used as it is more practical for real users as opposed to searching for an exact match.

3) Create a directory called frontend in your repository for all frontend-related code.
- Grouped frontend by features.