Architecture Diagram

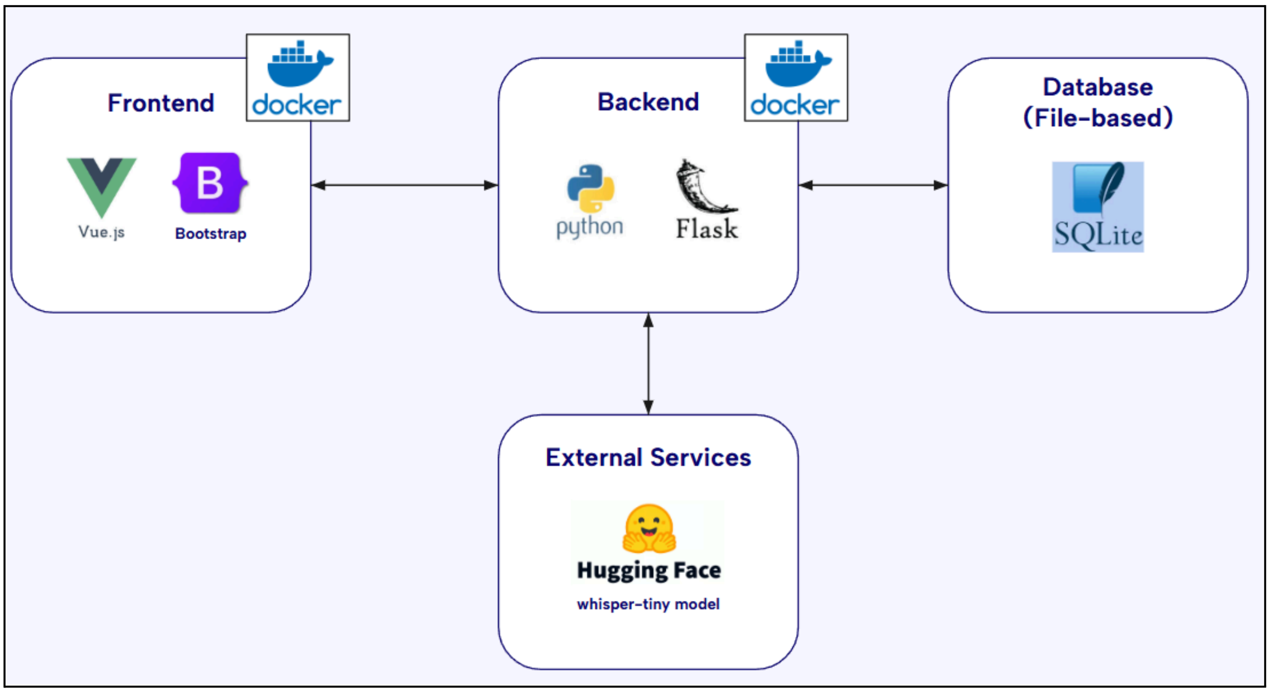

Diagram Explanation

1. Frontend

The frontend communicates with the backend through RESTful API calls (GET, POST), making CRUD requests for tasks like retrieving audio data, creating new audio data, and filtering features.

2. Backend

The backend exposes these RESTful APIs for the frontend to interact with. It handles the CRUD operations and communicates with the SQLite database. It interacts with the whisper-tiny model to transcribe audio files uploaded by the user.

3. Whisper-tiny model

The whisper-tiny model is responsible for transcribing the audio file and returning the transcribed text to the backend.

4. Database (File-based)

SQLite serves as the database for this application. It consists of two tables Audio and Transcription. Audio stores the filename, file_data (binary content), created_timestamp. Transcription stores file_id which is the foreign key of the file in Audio, transcribed_text of the audio file, created_timestamp

5. Docker

Docker is used to containerize both the frontend and backend, along with the SQLite image.

Assumptions and Considerations

1. Assuming this application handles a small to medium volume of data, using SQLite and Flask should be sufficient to manage the data load.

2. The whisper-face model transcribes the given audio file accurately and is always available.

3. SQLite is a file-based database, making it vulnerable to corruption or data loss. A possible workaround is to implement scheduled backups or consider switching to a database server (e.g. PostgreSQL, MySQL).