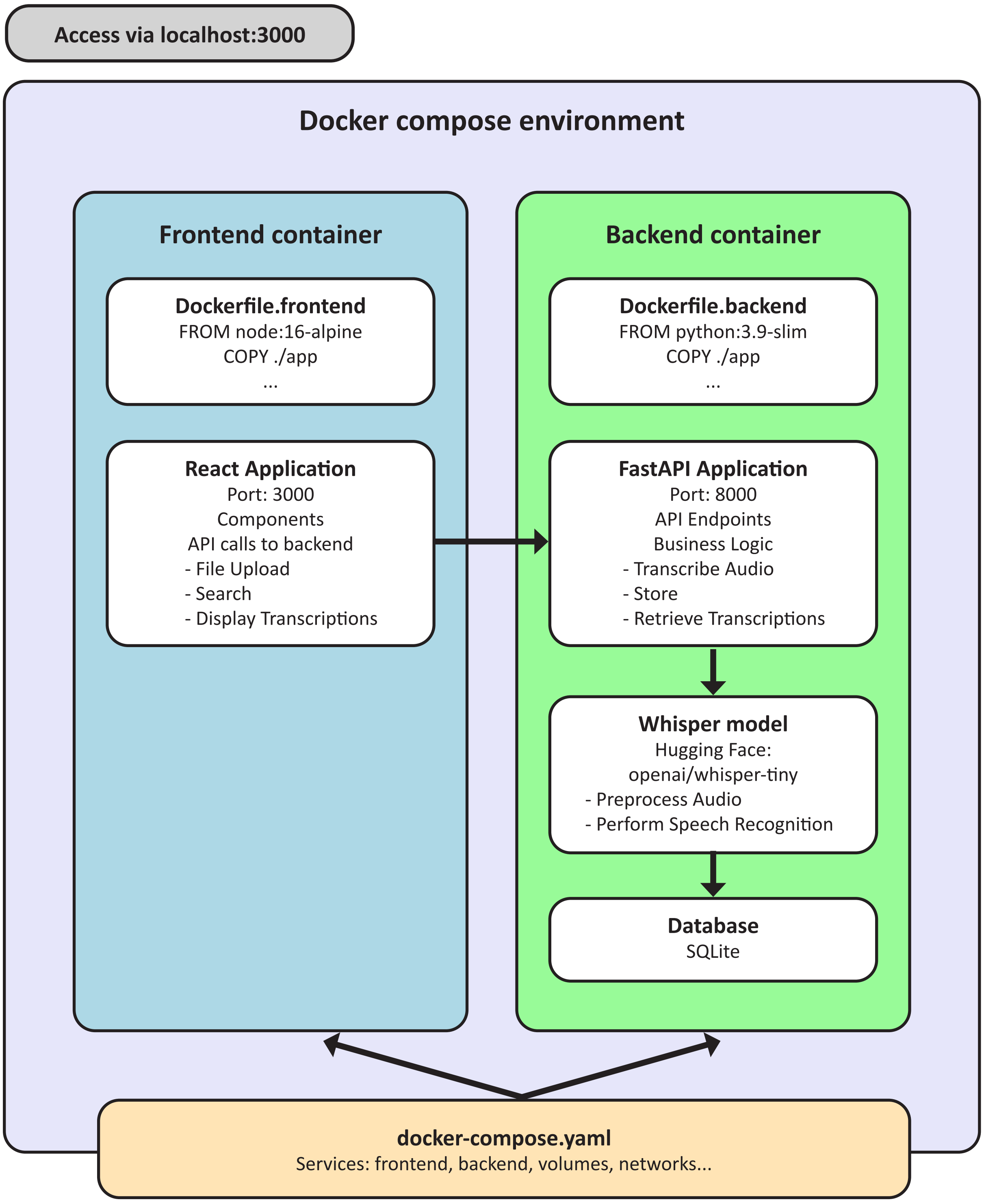

Summary

This full-stack application consisting of a backend and frontend, with specific functionalities for audio transcription using the Whisper speech recognition model and SQLite for storage. The project also includes containerization for deployment.

Explanation of Architecture

Frontend:

- Single-page application built with Javascript framework - React
- Users interact with the application via localhost:3000
- Users can upload audio files, view the list of transcriptions, and search for specific files.
- The frontend communicates with the backend via RESTful API endpoints.

Backend:

- Implemented using Python web framework FastAPI
- Handles API requests from the frontend and performs the following:
    - Audio file upload and preprocessing.
    - Transcription using the Whisper model.
    - Database operations: storing, retrieving, and searching transcriptions.
- Key endpoints:
    - **GET /health**: Returns the health status of the service.
    - **POST /transcribe**: Accepts audio files, performs transcription using the Whisper model, and saves results in SQLite.
    - **GET /transcriptions**: Retrieves all stored transcriptions from the database.
    - **GET /search**: Searches for transcriptions based on audio file names.

Whisper Model:

- The Whisper-tiny model (via Hugging Face) is integrated to perform audio-to-text transcription.

SQLite Database:

- Stores:
    - Audio file names.
    - Transcribed text.
    - Timestamps of when the transcription was created.

Communication Flow:

- **File Upload**:
    - The user uploads audio files via the SPA, which sends them to the backend /transcribe endpoint.
- **Transcription Processing**:

  - The backend processes the file, uses Whisper for transcription, and stores the results in SQLite.
- **Retrieve Transcriptions**:
  - The SPA requests all transcriptions from the /transcriptions endpoint and displays them.
- **Search Transcriptions**:
  - The SPA sends a search query to /search, and the backend queries the database, returning the matching results.

Containerization:

- Both the backend and frontend services are containerized using Docker, ensuring they can run in isolated, consistent environments.