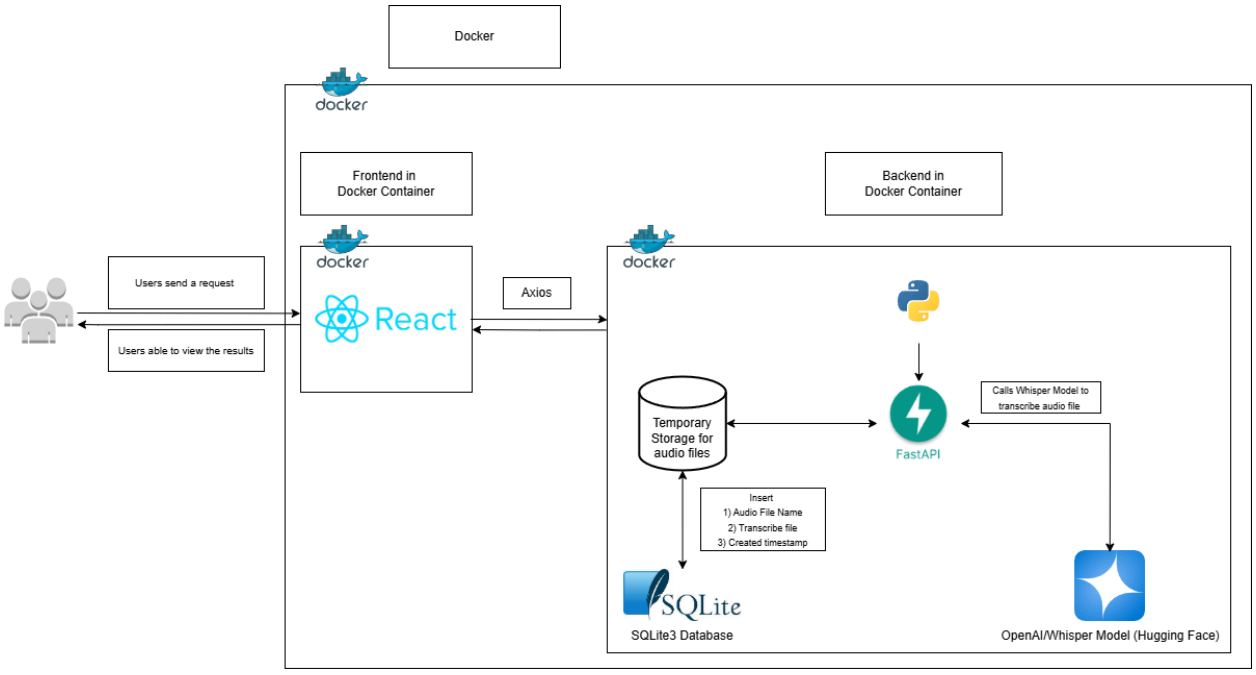
Current System Architecture



Frontend

I have structured the frontend into modular components, with the **upload audio file**, **search bar**, and **transcription table** as separated files in the “features” folder. This approach enhances:

- **Modularity:** Each feature can be developed and tested independently.
- **Reusability:** Components can be reused across different parts of the application (e.g., file upload for other features).
- **Maintainability:** Easier debugging, code organization, and future scalability.

Backend

The backend is built using **FastAPI**, which integrates the **Whisper model** for audio transcription. The process follows these steps:

1. **Receive File via API endpoint** (POST /transcribe)
2. **Store File** temporarily or in cloud storage.
3. **Pass File to Whisper** for transcription.
4. **Save Transcription Result** in the database, including:
    - File name
    - Transcribed text

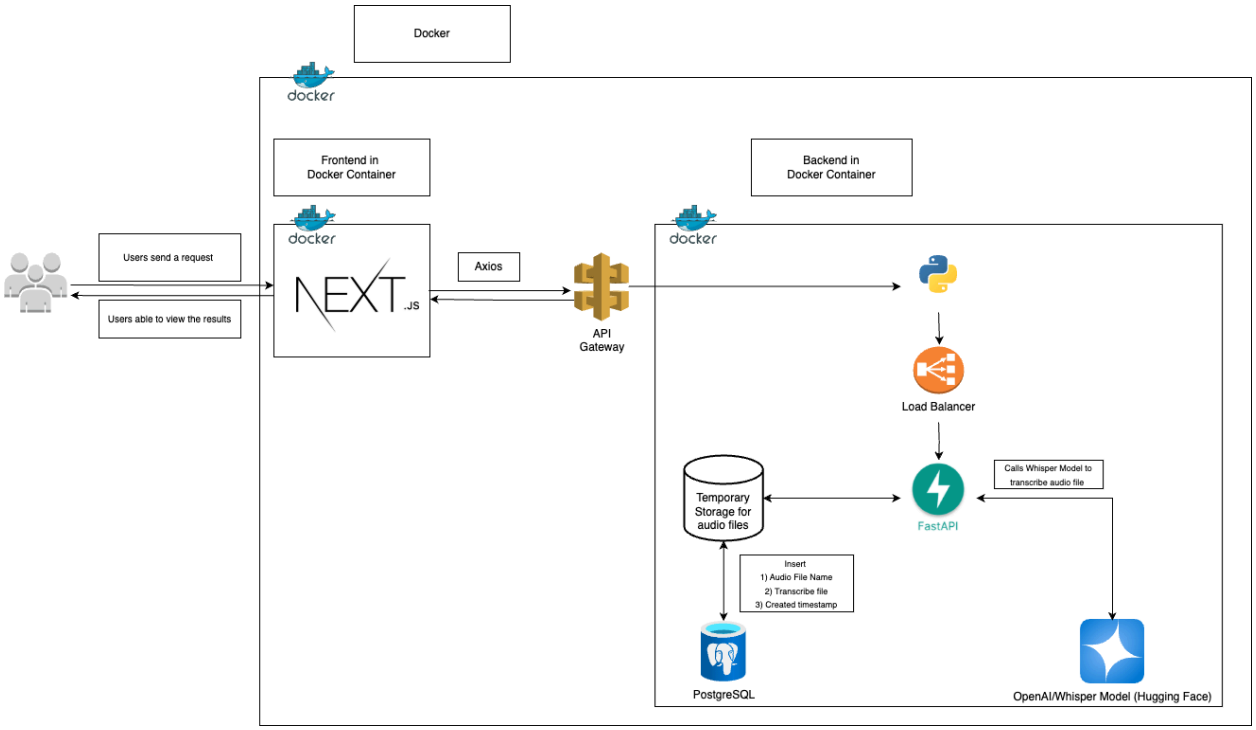
    - Timestamp
5. **Return Response** to the frontend with the transcription data.

Database

The system uses an **SQL database** to store transcription-related data, including:

- **File name** (for reference).
- **Transcribed text** (speech-to-text output).
- **Timestamp** (record creation time).

Suggested Improvement System Architecture



Considerations

Frontend

To enhance performance and scalability as traffic increases:

- **Enable Server-Side Rendering (SSR):** Improves page load speed and SEO (for eg, NextJS).
- **Optimize Performance:** Utilize lazy loading, code splitting, and caching to reduce load times.

- **Implement Pagination:** Enhances user experience and improves performance when handling large amounts of data.

Backend

To ensure smooth scaling and high availability:

- **Implement a Load Balancer** to distribute traffic across multiple backend instances.
- **Use Rate Limiting & Authentication** to prevent abuse and enhance security.
- **Sanitisation** of user input to prevent SQL injection
- Limit upload to audio files only to prevent upload of suspicious/malicious files into database

Database & Storage

To improve data persistence and scalability:

- **Persistent Storage for Audio Files:**
    - Migrate from temporary storage to **AWS S3, Google Cloud Storage, or Azure Blob Storage** to prevent data loss when containers restart.
- **Scalable Database Options:**
    - Replace SQLite with **PostgreSQL or MySQL** for better performance in production.
    - Use **MongoDB (NoSQL)** if a more flexible schema is required.
    - Consider **Cloud-managed databases (AWS RDS, Firebase, etc.)** for auto-scaling and backups.