# Speech to Text Application – Architecture

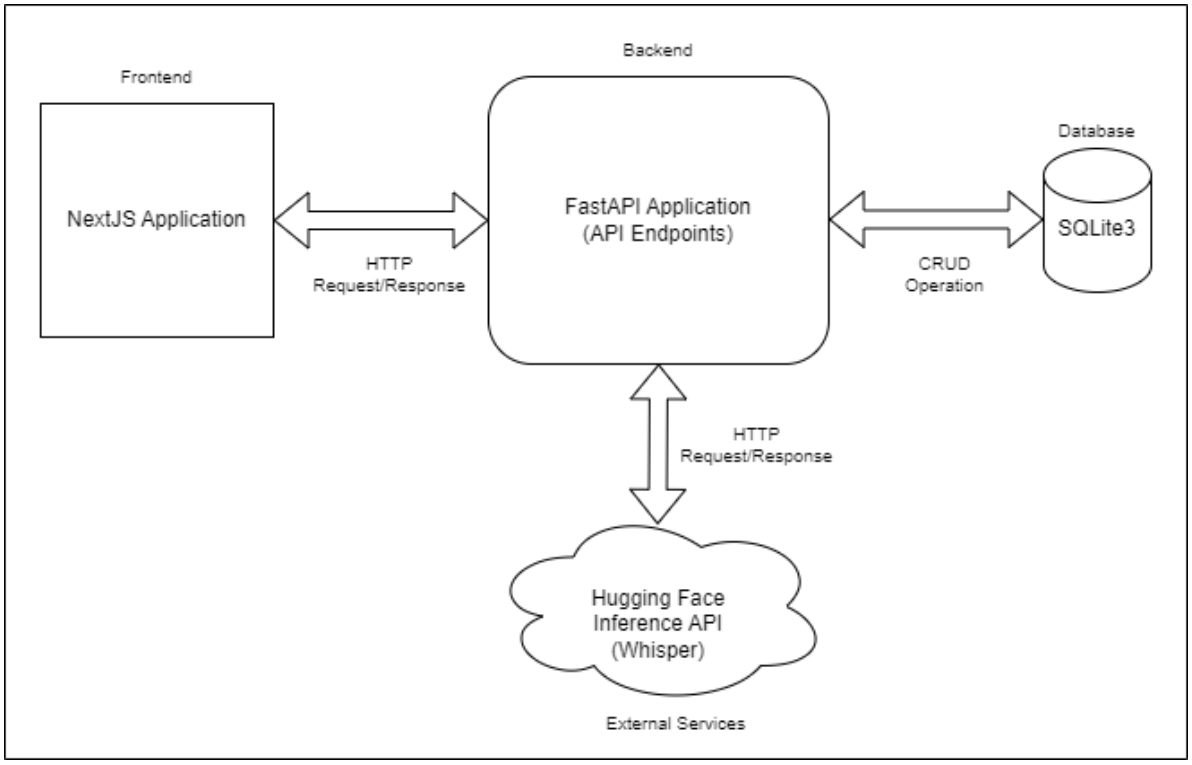


*Fig 1 System Architecture of application*

## 1. Overview

The Speech-to-Text application processes audio files into text transcriptions using Hugging Face's Whisper models. The system features configurable model selection, automated preprocessing, and data storage capabilities.

In the Docker setup (Container), the log file(log.log) and SQLite record(transcriptions.db) are volume mounted for data persistence and robust performance.

The application consists of four endpoints:

| **Endpoints** | **Action** |
|---|---|
| [POST] /stt/transcribe | Processes audio for text conversion |
| [GET] /data/transcriptions | Returns list of transcription records stored in database |
| [GET] /data/search | Queries database for matching transcriptions base on keywords matching filename or content |
| [DELETE] /data/delete | Deletes transcription record base on record_id |

## 2. Assumptions and Limitations

### 2.1 Infrastructure Constraints

- **Model Hosting:** No local infrastructure for Whisper model deployment; relies on Hugging Face's Inference API
- **Hardware Limitations:** Operates on CPU-only server environment without GPU acceleration for tasks. (e.g. VAD, PyTorch)

### 2.2 Audio Processing Constraints

- **Audio Format:** Designed for single-channel audio processing only
- **Not Supported:**
    - Dual channel recordings (e.g., call center recordings)
    - Complex audio sources (e.g., YouTube clips with multiple tracks)
    - Audio requiring advanced preprocessing

### 2.3 Prototype Scope

- **Purpose:** Evaluation and testing environment only, not for production use
- **Testing Focus:**
    - Model accuracy assessment
    - Performance benchmarking across different Whisper models (Tiny, Base, etc.)
    - Inference time measurement
- **Duration Limit:** Maximum 30 seconds per audio file

### 2.4 API Limitations

- **Rate Limiting:** Subject to Hugging Face's API quotas and rate limits
- **File Constraints:**
    - Duration restrictions based on file type
    - Large files require manual segmentation

## 3. Considerations and Implementation

### 1. Model Selection Flexibility (Runtime)

- Configurable via environment variable WHISPER_MODEL
- Enables easy switching between models (tiny, base, etc.) for testing

  Example: WHISPER_MODEL=openai/whisper-large

### 2. Cold Start Handling

- Problem: Initial Hugging Face API calls fail due to model loading time
- Solution: Implemented startup warm-up mechanism
    - Sends test request during application startup
    - Ensures model is ready for first user request

### 3. Audio Preprocessing To Enhance Transcription Quality

- Standardizes various input formats (MP3, WAV, FLAC) to match Hugging Face Whisper training dataset:
    - WAV format
    - Single channel
    - 16kHz sample rate
- Voice activity detection to remove silences from audio (Adjustable aggressiveness level during runtime)
- Ensures consistent transcription quality

# 4. Application Pipeline

### 1. Backend Services

When the application starts, it creates these cores singleton instances in order:

1. **Logger Instance**
    - Creates one shared logger that will be used throughout the entire application
    - All parts of the application use this same logger to record what's happening
2. **SQLite Database Instance**
    - Creates one shared database connection service
    - All database operations use this same connection
3. **Voice Detection (VAD) Instance**
    - Creates one shared Silero VAD service
    - All voice detection tasks use this same model
4. **Whisper Model Setup**
    - Tests the connection to Huggingface's speech recognition service
    - Uses a test audio file to make sure the service is ready
    - If the connection fails, it waits a bit and tries again

### 2. Frontend Services

The frontend application features three interconnected components that operate simultaneously, providing real-time updates to users:

#### 1. User Experience Flow

- User can interact with all three components simultaneously:
    - Upload audio file(s)
    - Searching through existing records (1 result per page)
    - Viewing the current list of transcriptions (3 per page)
- New uploads immediately appear in the records list
- Search results include newly added transcriptions

**2. File Upload Component**

- Handles single or multiple audio file uploads
- Sends files for transcription processing and provide feedback:
    - Number of successful transcriptions
    - Number of failed transcription requests
- Triggers real-time updates to other components after upload

**3. Search Component**

- Enables users to search existing transcriptions
- Search criteria include:
    - File names
    - Transcription content
- Results Pagination:
    - Displays one transcription result per page
    - Navigation controls to move between results

**4. Records Display Component**

- Shows all transcription records in chronological order (Newest at the top)
- Pagination Layout:
    - Displays 3 transcriptions per page
    - Navigation controls for browsing pages
- Updates automatically when new files are uploaded and transcribed without refresh