Table of Contents

**************DEMO LINK : Demo

Introduction

The Video Streaming Service is a full-stack web application built using **Spring Boot** for the backend and a custom frontend for client interaction. The platform allows users to:

1. Upload video files.
2. Process videos into **HLS (HTTP Live Streaming)** format for optimized streaming.
3. Stream the processed videos directly in the frontend player.

This manual provides instructions for setting up, using, and understanding the features and architecture of the application.

Features

Backend

1. **File Upload:** Upload video files to the server.
2. **Video Processing:** Automatically processes videos into segmented **HLS format** for efficient playback.
3. **Streaming:** Serves video files to the frontend for seamless playback in the browser.

Frontend

1. **Interactive UI:** Upload and play videos directly via the web interface.
2. **Video Player Integration:** Streams HLS-formatted videos using a built-in video player.

System Requirements

Hardware:

- Processor: Dual-core or higher
- RAM: 4GB or more
- Disk Space: Minimum of 1GB for storing video files

Software:

- **Backend:** Java 23, Spring Boot
- **Frontend:** Compatible browser (e.g., Chrome, Firefox)
- **Dependencies:**
    - FFmpeg (installed and added to system PATH)
    - Maven (for backend dependency management)
    - Node.js (optional, if the frontend is built using JavaScript frameworks)

Backend Overview

The backend is built with **Spring Boot** and provides REST APIs for:

1. Uploading video files.
2. Converting video files into HLS format.
3. Serving processed videos for playback.

Backend Core Components:

1. **VideoServiceImplementation Class:**
    - Handles file uploads and stores them locally.
    - Processes videos into HLS format using FFmpeg.
2. **VideoStore:**
    - Class for storing video metadata.
3. **API Controllers:**
    - Define endpoints for file upload, fetching videos, and streaming.

Frontend Overview

The frontend is a lightweight web interface designed to:

1. Upload video files to the backend.

2. Display the list of uploaded videos.
3. Play HLS-streamed videos using a built-in player.

Frontend Features:

- **File Upload Interface:** Simple form to upload video files.
- **Video List:** Displays available videos retrieved from the backend.
- **Player Integration:** Streams videos in HLS format using a compatible player.

API Endpoints

1. File Upload

- **Endpoint:** `/api/v1/videos`
- **Method:** `POST`
- **Description:** Uploads a video file to the backend for processing.
- **Request Body:** `MultipartFile` (video file)
- **Response:** Metadata of the uploaded video (e.g., ID, title, file path).

2. Fetch Video Metadata

- **Endpoint:** `/api/v1/videos/{videoId}`
- **Method:** `GET`
- **Description:** Fetches metadata for a specific video.
- **Response:** JSON object containing video details (e.g., title, file path).

3. Get All Videos

- **Endpoint:** `/api/v1/videos`
- **Method:** `GET`
- **Description:** Fetches a list of all uploaded videos.
- **Response:** List of video metadata.

4. Stream Video

- **Endpoint:** `/api/v1/{videoId}/master.m3u8` and `/api/v1/{videoId}/{segment}.ts`
- **Method:** `GET`
- **Description:** Streams the processed HLS video for the given video ID.
- **Response:** HLS playlist (`master.m3u8`) for the requested video.

How to Use the Application

Step 1: Setup Backend

1. Clone the project repository.
2. Ensure FFmpeg is installed on your system.
3. Define your MySQL connection string in the spring boot configuration file (http://localhost:3306/{YOUR_DB_NAME}) if using local database
4. Navigate to the backend project directory.

   Run the following commands:

   ```
   mvn clean install
   ```

   ```
   mvn spring-boot:run
   ```

5.
6. The backend will start at `http://localhost:8080`.

Step 2: Setup Frontend

1. Navigate to the frontend directory.

   Install dependencies (if using a JavaScript framework):

   ```
   npm install
   ```

2.

   Start the frontend server:

   ```
   npm start
   ```

3.
4. Access the frontend at `http://localhost:3000` (default React/Node.js port).

Step 3: Upload Videos

1. Open the frontend in your browser.
2. Use the **Upload Video** form to upload a file.
3. Wait for the upload to complete. The backend will process the video into HLS format.

Step 4: Play Videos

1. Go to the video list in the frontend.

2. Select a video to play. The video will stream directly in the integrated player.

Technical Details

Video Processing Workflow

1. **Upload:**
    - The video is uploaded via the `/api/videos` endpoint.
    - Metadata is saved to the database, and the file is stored locally.
2. **HLS Conversion:**
    - Videos are processed using FFmpeg into the HLS format (`master.m3u8`).
    - The processed segments (`.ts`) and playlists are stored in the specified HLS directory.
3. **Streaming:**
    - The HLS playlist is served via the `api` endpoint.
    - The video player fetches and plays the HLS segments.

Folder Structure

- **Video Storage:** `video-folder` (configured in `application.properties`)
- **HLS Output:** `video-hls` (processed HLS files)

FFmpeg Command for HLS Conversion

The backend uses the following FFmpeg command for processing:

```
ffmpeg -i {input_video_path} -c:v libx264 -c:a aac -strict -2
-f hls \-hls_time 10 -hls_list_size 0 -hls_segment_filename
"{output_path}/segment_%03d.ts" \

"{output_path}/master.m3u8"
```

Known Issues and Future Improvements

Known Issues:

1. **Frontend Bugs:**
    - Some UI elements may not function correctly due to incomplete error handling.
2. **Limited Scalability:**
    - Currently, the backend relies on local storage, which may not scale well for larger workloads.

Future Improvements:

1. **Cloud Integration:**
    - Store videos and HLS segments in cloud storage (e.g., AWS S3).
2. **Authentication:**
    - Add user authentication for secure video uploads and access.
3. **Improved Player Integration:**
    - Support for adaptive bitrate streaming in the video player.
4. **Frontend Enhancements:**
    - Optimize UI and fix existing bugs.

Conclusion

This project demonstrates the implementation of a basic video streaming service with file upload, HLS processing, and streaming capabilities. While there are areas for improvement, the service provides a strong foundation for further development and scaling.

Enjoy streaming! 🎬