Documentation: Video Upload and Processing Website with Subtitle Extraction and Search

1. Project Overview

This project involves the development of a web application that allows users to upload videos, process them in the background, extract subtitles using `FFmpeg`, and display those subtitles as closed captions on the video. Users can also search for phrases within the video, and the search results will provide timestamps that allow users to play the video from a specific point. The entire system is containerized using Docker, with a Django backend and a PostgreSQL database.

2. Requirements

- **Video Upload**: Users can upload video files to the server.
- **Background Processing**: The uploaded video is processed asynchronously to extract subtitles.
- **Subtitle Extraction**: Use `FFmpeg` to extract subtitles from the uploaded videos. This tool is mandatory for this project.
- **Subtitle Display**: Display subtitles on the frontend, synchronized with the video, as closed captions.
- **Search Functionality**: Allow users to search for a phrase in the subtitles and return the exact timestamp of its occurrence.
- **List View**: Provide a list view of all uploaded videos for easy access and management.
- **Containerization**: The application is fully containerized using Docker, including the Django backend and PostgreSQL database.
- **Storage**: Videos are stored in the Django media folder, and subtitles are stored in a PostgreSQL database.

3. Technologies Used

- **Backend**: Django (Python)
- **Database**: PostgreSQL
- **Frontend**: HTML, CSS, and basic JavaScript (No emphasis on frontend design)
- **Video Processing**: `FFmpeg` for subtitle extraction
- **Containerization**: Docker and Docker Compose

4. System Architecture

The application follows a typical web architecture:

1. **Frontend (UI)**: A simple, functional interface allowing users to upload videos, view subtitles, search within subtitles, and play videos.
2. **Backend (Django)**: Handles file uploads, background video processing, subtitle extraction, and management of video metadata.
3. **Database (PostgreSQL)**: Stores video metadata, extracted subtitles, and user search queries.
4. **FFmpeg**: Processes video files for subtitle extraction in various languages.
5. **Docker**: Containerised the entire application for easy setup and deployment.

5. Installation Guide

Prerequisites

- Docker and Docker Compose installed
- `FFmpeg` installed locally for subtitle extraction

6. Application Components

Backend: Django

- **File Upload**: Django's `FileField` and media handling system are used to allow users to upload videos to the backend.

Frontend: Simple UI

- The frontend uses minimal HTML, CSS, to display a file upload form, video playback, and a search box.
- Subtitles are displayed as closed captions using the `<track>` tag in HTML5 video.

PostgreSQL Database

- PostgreSQL is used to store the video and the subtitles extracted from each video.

Video Processing: FFMPEG

- `FFmpeg` is used to extract subtitles in multiple languages from the uploaded video.

7. Feature Implementation

1. Video Upload

- Users upload video files using a simple HTML form.
- The uploaded files are saved in the Django media folder and an entry is created in the database.

2. Subtitle Extraction

- After the video upload, `FFmpeg` is invoked to extract the subtitles from the video.
- Subtitles are saved in the database in a structured format, including timestamps and language codes.

3. Subtitle Search and Timestamp Retrieval

- The user can search for phrases in the subtitle data using a search form.
- The search is case-insensitive and retrieves the matching timestamps for each phrase.
- Clicking on a timestamp will trigger the video to start playing from that point.

4. List View for Uploaded Videos

- The list view displays all uploaded videos.
- When a user selects a video, the corresponding subtitles are retrieved, and the video is displayed with the closed captions.

8. Test Case

The application should be tested with a provided sample video. After uploading the sample video:

1. The video should be processed in the background.
2. Subtitles should be extracted using `FFmpeg`.
3. The subtitles should be displayed correctly as closed captions.
4. Users should be able to search for a phrase, and clicking the result should play the video from the corresponding timestamp.

9. Future Enhancements

- **Multiple Video Formats**: Support additional video formats for subtitle extraction.
- **Language Detection**: Automatically detect the language of the subtitles.
- **Advanced Search**: Implement fuzzy search to improve subtitle search functionality.