Technical Documentation for Video Captioning Web App

# Overview

This web application allows users to add captions to a video by entering a URL of a hosted video, inputting captions with specific timestamps, and displaying the video with the captions. Users can pause and play the video, and captions are displayed at the specified times.

# Table of Contents

* Project Structure
* Component Breakdown
* Technical Decisions
* User Experience
* Edge Cases
* Future Improvements
* Setup Instructions

# Project Structure

src/  
|-- assets/  
| |-- logo.png  
| |-- react.svg  
|-- components/  
| |-- CaptionInput.jsx  
| |-- CurrentCaption.jsx  
| |-- Header.jsx  
| |-- VideoCaptioner.jsx  
| |-- VideoInput.jsx  
| |-- VideoPlayer.jsx  
|-- App.jsx  
|-- App.css  
|-- index.css  
|-- main.jsx

# Component Breakdown

## App.jsx

The main entry point of the application. It renders the `Header` and `VideoCaptioner` components.

## Header.jsx

Displays the logo and a brief description of the app. Styled using `Header.css`.

## VideoCaptioner.jsx

The core component that manages the state of the video URL, captions, and current playback time. It renders `VideoInput`, `VideoPlayer`, `CaptionInput`, and `CurrentCaption`.

## VideoInput.jsx

Allows users to input a video URL. It captures the URL and triggers a callback to update the video URL state in `VideoCaptioner`.

## VideoPlayer.jsx

A wrapper around the `ReactPlayer` component. It handles video playback and periodically updates the current playback time.

## CaptionInput.jsx

Provides a textarea for users to input captions along with their timestamps. Captions are parsed and validated before being passed to the parent component.

## CurrentCaption.jsx

Displays the current caption based on the current playback time.

## App.css and Header.css

Contain styling for the application, ensuring a consistent and user-friendly interface.

# Technical Decisions

* React: The application is built using React due to its component-based architecture and efficient state management, which simplifies the process of building interactive UIs.
* ReactPlayer: Used `ReactPlayer` for handling video playback because it supports a wide range of video URLs and provides easy control over playback functionality.It also allows fine grained information access to the currently playing video functions like get current time and onPlay and Pause
* CSS Modules: Used CSS for styling to ensure modular and maintainable styles. This approach keeps styles scoped to specific components, avoiding conflicts.
* State Management: Used React's `useState` and `useEffect` hooks for managing state and side effects, making the app responsive to user interactions.

# User Experience

* Input Validation:   
  - Video URL Input: Ensures the user inputs a valid URL.   
  - Caption Input: Validates the format of the caption input, ensuring that timestamps are correctly formatted and that start times are before end times.
* Feedback:   
  - Error Messages: Displayed when the input format is incorrect.   
  - Success Messages: Displayed when captions are successfully added.
* Responsive Design:   
  - Layout: The app is styled to be responsive, ensuring a good user experience on different screen sizes.

# Edge Cases

* Invalid URL: If the user inputs an invalid video URL, the app handles it gracefully without crashing.
* Empty Input: Handles cases where the input fields are empty, providing appropriate feedback to the user.

# Future Improvements

* Enhanced Validation: Improve caption validation to handle more complex scenarios.
* We could leverage a LLM to validate the caption input prompt and get the start time and the time for wide variety of user inputs
* Video Player Enhancement: A better video with an interactive UI to directly fix timestamps for the start and the end time in the video player itself.
* Advanced Styling: Implement more advanced styling and transitions for a better user experience.
* Caption Editing: Allow users to edit and delete captions after they have been added.
* Multiple Video Sources: Support for more video sources beyond URLs, such as file uploads.
* Localization: Support multiple languages for captions.

# Setup Instructions

## Prerequisites

Node.js and npm installed

## Installation

* Clone the repository:   
  ```  
  git clone https://github.com/MarezKwiup/spyne-assignment.git  
  ```
* Install dependencies:   
  ```  
  npm install  
  ```
* Run the application:   
  ```  
  npm run dev  
  ```
* Open the application in your browser:   
  ```  
  http://localhost:3000  
  ```

## Hosting

This app has been hosted on vercel at the link : <https://spyne-assignment-git-se-97f3cd-vardan-sharmas-projects-770fc74a.vercel.app/>