# Project Documentation: Search App

## 1. Project Overview

The Search App is a web application that allows users to search for relevant YouTube videos and articles based on a query. The app fetches data from the YouTube Data API and Google Custom Search API, and displays the results based on the user's preferences for sorting (views or likes) and filtering (YouTube or Articles).

## 2. Technologies Used

### Frontend:

React.js

Bootstrap (CSS framework)

### Backend:

Node.js

Express.js

node-fetch (for HTTP requests to external APIs)

### APIs:

YouTube Data API (for fetching YouTube videos and statistics)

Google Custom Search API (for fetching relevant articles)

## 3. Project Structure

/Search-App  
│  
├── /Components  
│ ├── Navbar  
│ │ └── Navbar.jsx  
│ ├── Search-Input  
│ │ └── Search.jsx  
│ ├── MainContainer  
│ │ └── Main.jsx  
│ ├── Footer  
│ │ └── Footer.jsx  
│   
├── App.jsx  
├── server.js  
├── package.json  
└── node\_modules

- App.jsx: Main application file for the frontend.  
- server.js: The backend server responsible for fetching data from external APIs.  
- /Components: Contains all React components (Navbar, Search Input, Main container for results, and Footer).

## 4. Functionalities

### Frontend:

- Search: Users can input a search term in the search bar, and based on the query, results from YouTube and Google Custom Search will be fetched.

- Filtering: Users can filter the results to show either 'YouTube' videos or 'Articles' or 'All' by selecting from a dropdown.

- Sorting: Users can sort YouTube videos by 'Most Views' or 'Most Likes' before displaying the results.

- Results Display: The search results are displayed in a scrollable container with YouTube videos showing view and like counts, and articles having clickable links to the source.

- Responsive UI: The UI is responsive and styled using Bootstrap.

### Backend:

- Fetching YouTube Data: The backend fetches videos using the YouTube Data API based on the search query.

- Fetching Article Data: The backend fetches articles using the Google Custom Search API.

- Combining Results: The backend combines the YouTube and article results and sends them back to the frontend.

- Error Handling: If an error occurs during the data fetch, appropriate error messages are shown.

## 5. API Configuration

To make this project work, you need to configure the following API keys:

### YouTube Data API Key:

- Get your YouTube Data API key from Google Cloud Console.  
- Replace YOUR\_YOUTUBE\_API\_KEY in the server.js file with the actual API key.

### Google Custom Search API Key:

- Create a Google Custom Search Engine and get the API key from the Google Developer Console.  
- Replace YOUR\_GOOGLE\_CUSTOM\_SEARCH\_API\_KEY and YOUR\_SEARCH\_ENGINE\_ID in the server.js file.

## 6. Components Overview

### App.jsx:

The main component that coordinates the data flow between the child components and manages the application’s state, including:  
- searchTerm: Stores the current search input.  
- results: Stores the fetched search results (videos and articles).  
- selectedFilter: Determines whether the user wants to see 'All', 'YouTube', or 'Articles'.  
- selectedSort: Determines whether YouTube results are sorted by 'Most Views' or 'Most Likes'.

### Navbar.jsx:

A simple navigation bar that contains the app’s title and a toggler to display additional information, such as a link to the footer.

### Search.jsx:

This component handles user input, filter selection, and sorting:  
- The search term is captured and sent to the backend when the search button is clicked.  
- Provides options to filter results and sort them.

### Main.jsx:

Displays the search results. This component renders YouTube videos and articles based on the search results provided by the backend. It also handles the display of view and like counts for YouTube videos.

### Footer.jsx:

Contains details like an “About Me” section, contact information, and links to social media profiles.

## 7. Server-side Logic (server.js)

The server.js file is responsible for handling requests from the frontend and fetching data from external APIs. The main logic is as follows:

### Express Server:

The server listens for requests on http://localhost:5000/search with a query parameter.

### YouTube Fetch:

It makes an API request to YouTube Data API using the fetch function to retrieve videos based on the search query.

### Google Search Fetch:

It also makes an API request to the Google Custom Search API to fetch articles related to the search term.

### Combine Results:

Both the YouTube and article results are combined into a single array and sent back to the client in JSON format.

## 8. Running the Application

To run this project on your local machine:

### Clone the Repository:

bash  
 git clone https://github.com/your-repo/search-app.git  
 cd search-app

### Install Dependencies:

bash  
 npm install

### Add Your API Keys:

Open server.js and replace the placeholder API keys with your actual API keys.

### Start the Backend:

bash  
 node server.js

The backend server will start at http://localhost:5000.

### Start the Frontend:

Open another terminal and start the frontend using:  
 bash  
 npm start

The React app will start at http://localhost:3000.

### Perform a Search:

Open your browser and navigate to http://localhost:3000. Use the search bar to perform searches and see the results from YouTube and Google.

## Challenges Faced

- Environment Variables: The variables in the .env file are not loading in the frontend App.jsx file, resulting in undefined values for the API keys.

## Conclusion

This Search App is a powerful tool to search for both YouTube videos and related articles. With its easy-to-use interface, filtering, and sorting capabilities, the app provides an intuitive user experience while fetching data from two different APIs.