Full-Stack YouTube Clone Project

Project Overview

The Full-Stack YouTube Clone is a MERN-stack application designed to replicate the core functionalities of YouTube, such as video streaming, user authentication, search and filtering, channel management, and comment handling. The goal of this project is to create a scalable, responsive, and user-friendly platform using MongoDB, Express.js, React, and Node.js (MERN stack).

Features and Requirements

Frontend (React)

1. **Home Page:** Header with branding, search bar, static sidebar (toggleable with a hamburger menu), and video grid with title, thumbnail, channel name, and view count.
2. **User Authentication:** Registration and login with username, email, and password. Secure session via JWT. Post-login, user's name appears in the header, redirecting to the homepage.
3. **Search and Filter Functionality:** Search bar filters videos by title, with category-based filters for refined results.
4. **Video Player Page:** Embedded video player, metadata display (title, description, channel name), like/dislike buttons, and comments (add, edit, delete).
5. **Channel Page:** Allows signed-in users to create and manage channels (add, edit, delete videos). Displays a list of channel videos.
6. **Responsive Design:** Fully optimized interface for desktops, tablets, and mobile devices.

Backend (Node.js, Express)

1. **API Endpoints**

- **User Authentication:** Sign-up, login, and secure session management with JWT.
- **Channel Management:** Create and fetch channel details.
- **Video Management:** CRUD operations for video data.
- **Comments:** Add and retrieve video comments.
- **Challenges:** Data relationships and meaningful error handling.

2. **Database (MongoDB)**

- **Collections:** User , Video , Channel , Comment.
- **Challenges:** Balancing embedded and referenced data structures for scalability and optimizing query performance.

Technologies Used

- **Frontend:** React, React Router, Axios, Tailwind CSS.
- **Backend:** Node.js, Express.js.
- **Authentication:** JWT for secure sessions.
- **Database:** MongoDB for storing and managing data.
- **Version Control:** Git for source code management.

Challenges and Solutions

1. **JWT Authentication:**
    - **Challenge:** Securing endpoints without impacting performance.
    - **Solution:** Implement middleware to validate tokens and handle expired tokens gracefully.
2. **Video Filtering:**
    - **Challenge:** Handling complex filtering and searching on large datasets.
    - **Solution:** Use indexed fields in MongoDB and server-side filtering for efficient searches.
3. **Frontend-Backend Integration:**
    - **Challenge:** Ensuring seamless communication between React and Express.
    - **Solution:** Standardize API responses and implement global error handling.
4. **Responsive Design:**
    - **Challenge:** Maintaining usability across devices.
    - **Solution:** Use Tailwind CSS utilities and test extensively on different devices.
5. **Data Relationships:**
    - **Challenge:** Managing relationships between users, channels, and videos.
    - **Solution:** Use a hybrid approach with embedded documents and references for scalability.

6. All the functionality works but reflect after the hard reloading

---

Note :-

1. For now please use a you tube link or any other link to upload without having an upload functionality by selecting from local storage as it's not mentioned there that it is mandatory or not.

Project Structure (Just covered the main one)

Frontend

```
src/
├── components/
│   ├── Header.jsx
│   ├── Sidebar.jsx
│   ├── FilterButtons.jsx
│   └── VideoCard.jsx
├── pages/
│   ├── Home.jsx
│   ├── Login.jsx
│   ├── Register.jsx
│   ├── VideoPlayer.jsx
│   └── Channel.jsx
├── redux/
│   ├── actions/
│   ├── reducers/
│   └── store.js
├── App.jsx
├── index.js
└── index.css
```

Backend

```
src/
├── controllers/
│   ├── authController.js
│   ├── channelController.js
│   ├── videoController.js
│   └── commentController.js
├── models/
│   ├── User.js
│   ├── Video.js
│   ├── Channel.js
│   └── Comment.js
├── routes/
│   ├── authRoutes.js
│   ├── channelRoutes.js
│   ├── videoRoutes.js
│   └── commentRoutes.js
```

```
├── middleware/
│   ├── authMiddleware.js
├── app.js
└── server.js
```

Future Enhancements

- Add nested comments and reply functionality.
- Implement video uploading with cloud storage (e.g., AWS S3).
- Add real-time chat or notifications.
- Integrate a recommendation system based on user activity.

Conclusion:

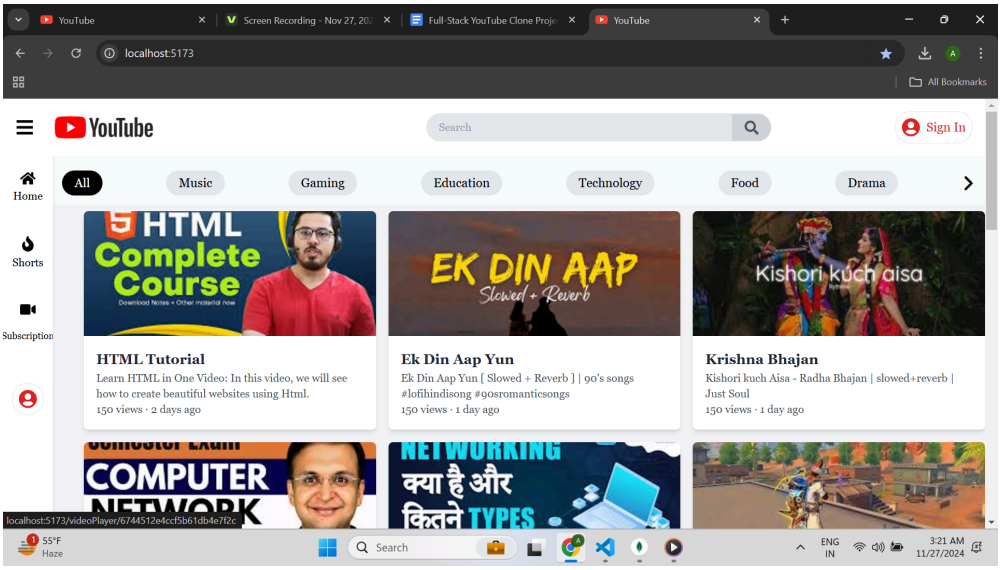
This project not only deepened my knowledge of the **MERN stack** but also improved my problem-solving abilities, particularly around handling complex user interactions and managing data relationships. The final product is a functional YouTube clone, showcasing my ability to develop full-stack applications, integrate authentication systems, and design responsive layouts.

The experience of tackling real-world challenges, such as UI state synchronization, database schema design, and ensuring smooth user experiences across various devices, has significantly enhanced my full-stack development skills. This project serves as a strong foundation for future development work, particularly in building scalable and user-friendly applications.

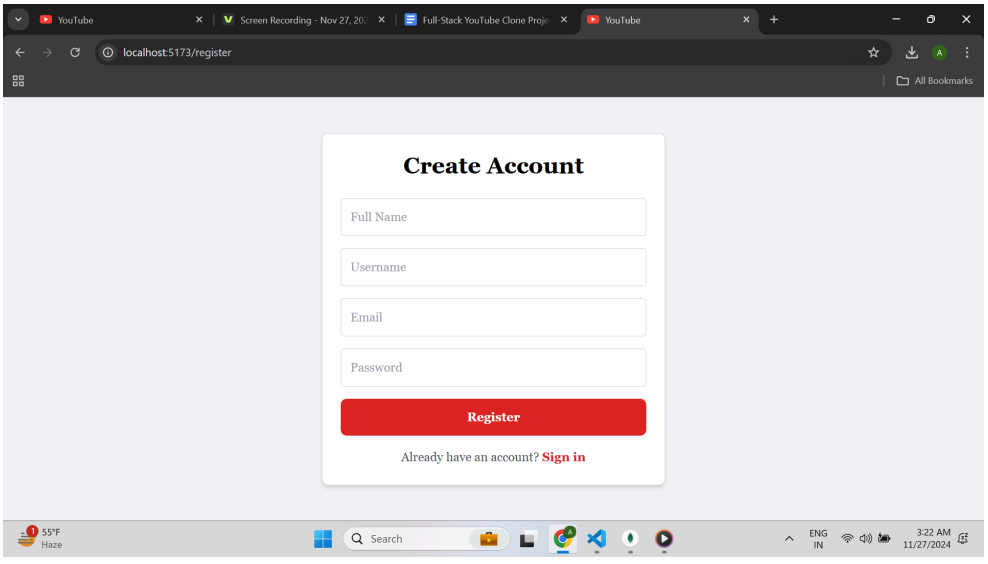
GITHUB LINK :

https://github.com/Ashmita2282/YouTube_CaptionProject

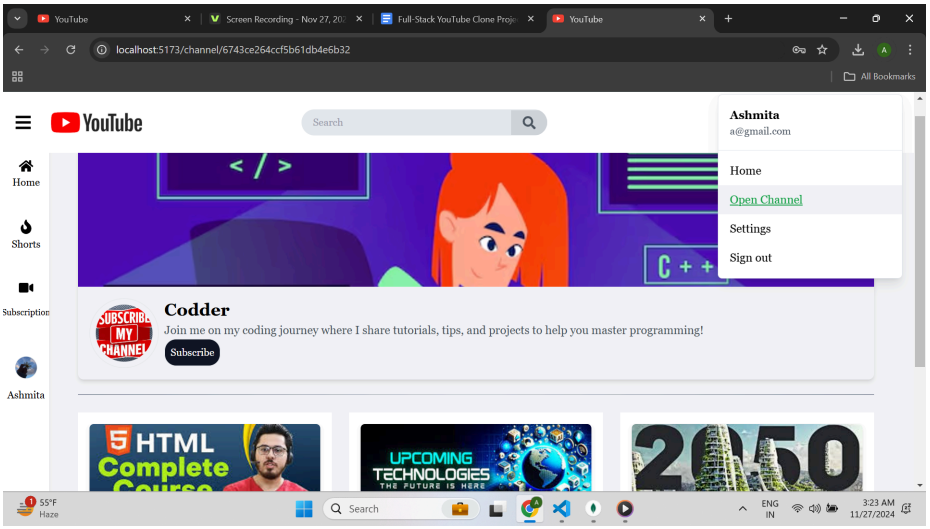
Home Page :



Login / Register



Channel Page :



Video Player Page:

Upload Video