

# Titles

Project Name – YouTube Clone

Build by – Rakesh koduri

Batch – 10<sup>TH</sup> March 2025

# Table of Contents

1. Introduction
2. Organized File Structure
3. Creativity and Presentation
  - o Technologies Used o code
  - o Screenshot
4. GitHub Link

## Introduction

This YouTube Clone project is a full-stack video streaming web application built using the MERN stack (MongoDB, Express.js, React.js, Node.js) and styled with Tailwind CSS. It replicates the core features of YouTube, enabling users to:

- Sign up and log in securely using JWT-based authentication.
- Create their own channel (only one per user).
- Upload videos with metadata such as title, description, thumbnail, category, and URL.
- View and interact with videos through likes, dislikes, and comments.
- Browse content using search and category filters.
- Manage their own uploaded videos (edit/delete).

The goal of this project is to understand how real-world, production-grade fullstack applications are built and structured — involving both frontend and backend integration, RESTful API design, authentication, CRUD operations, and responsive UI/UX practices.

This project is ideal for developers who want hands-on experience building a complete media-sharing platform with user interaction, protected routes, and modern UI responsiveness.

# Organized File Structure

Creating a well-organized file structure is crucial for readability, maintainability, and collaboration in any web project. My project is already structured well, and here's a breakdown of each component:

Description of Files:

## **backend/ – Node.js, Express, MongoDB (API Layer) server.js**

- Entry point for the backend server.
- Sets up Express, connects to MongoDB, and mounts routes.
- Runs on a port (e.g., 5000).

---

## **Controllers/ – Route handlers**

- authController.js
  - Handles user registration and login.
  - Returns JWT on login.
- channelController.js
  - Handles channel creation, fetching channel by ID/user.
  - Manages one-channel-per-user rule.
- videoController.js
  - Handles uploading, updating, deleting, liking/disliking, and retrieving videos.
- commentController.js
  - Handles posting, editing, deleting, and fetching comments for a video.

---

## **Models/ – Mongoose schemas**

- User.js
  - Schema for user (email, username, password).
- Channel.js
  - Schema for user's channel.
  - References user and videos.
- Video.js

- Schema for videos.
  - Includes title, category, thumbnail, likes/dislikes, views, channelId, uploader.
- Comment.js
  - Schema for user comments.
  - References video and user.

---

## Routes/ – Express routes

- authRoutes.js
  - POST /Api/auth/register
  - POST /Api/auth/login
  - GET /api/auth/me
- channelRoutes.js
  - POST /Api/channels – create new channel
  - GET /Api/channels/:channelId
  - GET /Api/channels/user/:userId
- videoRoutes.js
  - POST /Api/videos – upload
  - GET /Api/videos – list all
  - GET /api/videos/:videoId – details
  - PUT /Api/videos/:videoId – edit
  - DELETE /Api/videos/:videoId – delete
  - POST /Api/videos/:videoId/like
  - POST /Api/videos/:videoId/dislike
- commentRoutes.js
  - POST /Api/comments/:videoId
  - GET /Api/comments/:videoId
  - PUT /Api/comments/:commentId
  - DELETE /Api/comments/:commentId

---

## Middleware/

- authMiddleware.js
  - Verifies JWT in request headers.
  - Sets req.user if valid.

---

## frontend/ – React + Tailwind CSS (UI Layer) src/

---

### main.jsx

- Entry point for the React app.
- Wraps app in context providers.

## **App.jsx**

- Main layout.
  - Renders Header, Sidebar, and route-based pages.
- 

## **pages/**

- Home.jsx
    - Lists all videos.
    - Filters by category.
    - Fetches videos from context.
  - VideoPlayer.jsx
    - Displays a selected video using iframe.
    - Handles likes, dislikes, and comment functionality.
  - Channel.jsx
    - Allows user to create channel.
    - Lists their uploaded videos.
    - Option to edit or delete.
  - UploadVideo.jsx
    - Uploads new video with details like thumbnail, title, category.
    - Validates and POSTs data.
  - EditVideo.jsx
    - Edit existing uploaded video.
  - Login.jsx
    - User login form.
    - Stores JWT token.
  - Register.jsx
    - User registration form.
- 

## **components/**

- Header.jsx
  - Logo, search bar, sign-in/sign-out button.
  - Responsive toggle for sidebar.
- Sidebar.jsx
  - Navigation links: Home, My Channel, Explore.
  - Responsive: hidden on small screen, toggleable on medium+ screens.
- VideoCard.jsx
  - Reusable card component to display video thumbnail, title, and metadata.

---

## **context/**

- AuthContext.jsx
  - Stores user info and JWT.
  - Handles login, logout.
  - Used across protected routes and components.
- VideoContext.jsx
  - Central video store.
  - Provides all videos and filtered results.
  - Search handler logic for header and home.

---

## **api/**

- axios.js
  - Axios instance with base URL (<http://localhost:5000/api>)
  - Simplifies importing axios in components.

---

## **routes/**

- ProtectedRoute.jsx
  - Wraps around routes that require authentication.
  - Redirects unauthenticated users to login.

# **Creativity and Presentation**

This project was developed with both functionality and user experience in mind. The goal was to build a YouTube-like video-sharing platform with a clean UI, responsive layout, and interactive features. Below is an overview of the technologies used to build and style the application.

---

## **Technologies Used**

## **Backend:**

- Node.js – JavaScript runtime for building server-side logic.
- Express.js – Web framework for creating RESTful APIs.
- MongoDB – NoSQL database used for storing users, videos, comments, and channels.
- Mongoose – ODM for MongoDB, helps with schema and database operations.
- JWT (JSON Web Token) – Used for user authentication and authorization.
- Bcrypt.js – For password hashing.
- CORS & dotenv – Cross-origin access and environment variable configuration.

## **Frontend:**

- React.js – Frontend JavaScript library for building the UI.
- React Router – Handles navigation between pages (Home, Video, Channel, etc.).
- Axios – For making HTTP requests to the backend.
- Tailwind CSS – Utility-first CSS framework for fast and responsive styling.
- React Icons – For icons used in the UI (e.g., like/dislike, menu, account).
- Vite – Development server and bundler for fast React project bootstrapping.

## **API and Data Flow:**

- RESTful APIs built using Express and consumed using Axios on the client side.
- Protected routes ensure that certain pages (like uploading or editing videos) are only accessible to logged-in users.

## **Other Tools:**

- **MongoDB Compass** – For managing and viewing the database.
- **Postman** – API testing and development.

## Code

### Server.js

```
import express from "express"; import mongoose from
"mongoose"; import dotenv from "dotenv"; import cors from "cors";
import authRoutes from "./Routes/authRoutes.js"; import
commentRoutes from "./Routes/commentRoutes.js"; import
channelRoutes from "./Routes/channelRoutes.js"; import
videoRoutes from "./Routes/videoRoutes.js";

//Load environment variable dotenv.config();

const app = express(); const PORT =
process.env.PORT || 5000;

//Middleware app.use(express.json()); app.use(cors());

//API Routes app.use("/api/auth", authRoutes);
app.use("/api/comments", commentRoutes);
app.use("/api/channels", channelRoutes); app.use("/api/videos",
videoRoutes);
```



```

//Routes placeholder

// app.get("/", (req, res) => {
//   res.send("Youtube Clone API is running");
// });

// MongoDB Connection mongoose
.connect(process.env.MONGO_URI)
.then(() => { console.log("Connected to MongoDB");
app.listen(PORT, () => console.log(`Server running on http://localhost:
${PORT}`)
});
})
.catch((err) => console.log("MongoDB connection error:", err));

```

### Models/User.js

```

import mongoose from "mongoose";

const userSchema = new mongoose.Schema(
  {
    username: { type:
String,
required: true,
}, email: { type:
String, required:
true, unique: true,
},
password: { type:
String,
required: true,
},
avatar: { type:
String,
default: "",

```

```

    },
    channels: [
      {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Channel",
      },
    ],
  },
  { timestamps:
true,
  }
);

const User = mongoose.model("User", userSchema); export default
User;

```

### Models/Video.js

```

import mongoose from "mongoose";

const videoSchema = new mongoose.Schema(
  {
    videoId: { type:
String, required:
true, unique: true,
    },
    title: { type:
String, required:
true, trim: true,
    },
    thumbnailUrl: {
type: String,
required: true,
    },
    description: { type:
String, required: true,
    },
    videoUrl: { type:
String,

```

```

required: true,

    },    channelId: {
type:
mongoose.Schema.Types.ObjectId,
    ref: "Channel",    required:
true,

    },    uploader: {
type:
mongoose.Schema.Types.ObjectId,
    ref: "User",    required:
true,

    },
category: {
type: String,
default: "Other",
enum: [
    "Movies",
    "Music",
    "TV Show",
    "Gaming",
    "Education",
    "News",
    "Sports",
    "Travel",
    "Food",
    "Other",
    ],    },    views:
{    type: Number,
default:
0,
    },    likes:
[
    {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User",
    },
],
dislikes: [

```

```

    {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
    },
  ],
  uploadDate: {
    type: Date,    default:
    Date.now,
  },
},
{ timestamps: true }
);

```

```

const Video = mongoose.model("Video", videoSchema); export default
Video;

```

### Models/Comment.js

```

import mongoose from "mongoose";

const commentSchema = new mongoose.Schema(
  {
    videoId: {
      type:
      mongoose.Schema.Types.ObjectId,
      ref: "Video",    required:
      true,
    },
    userId: {
      type:
      mongoose.Schema.Types.ObjectId,
      ref: "User",    required:
      true,
    },
    text: {
      type:
      String,    required:
      true,
    },
    timestamp: {
      type: Date,
      default: Date.now,
    },
  },

```

```

    },
    { timestamps:
true,
    }
);

const Comment = mongoose.model("Comment", commentSchema); export default
Comment;

```

### Models/Channel.js

```

import mongoose from "mongoose";

const channelSchema = new mongoose.Schema(
  {
    channelName: {
      type: String,    required:
true,
    },    owner: {
type:
mongoose.Schema.Types.ObjectId,
      ref: "User",    required:
true,
    },
    description: {
type: String,
required: true,
    },
    channelBanner: {    type:
String,    default:
"",
    },
    subscribers: {    type:
Number,
default: 0,
    },
    videos: [
    {

```

```

    type: mongoose.Schema.Types.ObjectId,
    ref: "Video",
  },
],
},
{ timestamps:
true,
}
);

const Channel = mongoose.model("Channel", channelSchema); export default
Channel;

```

### **Middleware/authMiddleware.js**

```

import jwt from "jsonwebtoken";

const authMiddleware = (req, res, next) => {  const authHeader
= req.headers.authorization;

  if (!authHeader || !authHeader.startsWith("JWT")) {    return
res
    .status(401)
    .json({ error: "Authorization header missing or malformed" });
  }
  const token = authHeader.split(" ")[1];
  try {
    const decoded = jwt.verify(token, "secretKey");    req.user
= decoded;    next();  } catch (error) {
res.status(403).json({ error:
"Invalid or expired token" });
  }
};

export default authMiddleware;

```

### **Controllers/authController.js**

```

import User from "../Models/User.js";
import bcrypt from "bcryptjs"; import jwt
from "jsonwebtoken";

//Register User export const registerUser = async
(req, res) => { try { const { username, email, password } =
req.body; const existingUser = await User.findOne({ email });
if (existingUser) { return res.status(400).json({ error: "User
already exists" });
}
const hashedPassword = await bcrypt.hash(password, 10);
const newUser = new User({ username, email,
password: hashedPassword,
});
await newUser.save(); res.status(201).json({ message: "User
registered successfully" });
} catch (error) { res.status(500).json({ error:
"Registration failed" });
}
};

//Login User export const loginUser = async (req,
res) => { try { const { email, password } =
req.body; const user = await User.findOne({
email });
if (!user) { return res.status(400).json({ error: "Invalid credentials"
});
}
const isMatch = await bcrypt.compare(password, user.password); if (!isMatch) { return res.status(400).json({
error: "Invalid credentials" });
}
const token = jwt.sign({ id: user._id }, "secretKey", { expiresIn: "1h" });
res.json({ token, user:
{ id: user._id, username:
user.username, email:
user.email, avatar: user.avatar,
},
});

```

```

    } catch (error) { res.status(500).json({ error:
"Login failed" });
    }
};

```

### **Routes/authRoutes.js**

```

import express from "express"; import { loginUser, registerUser } from
"../Controllers/authController.js"; import authMiddleware from
"../Middleware/authMiddleware.js"; import User from "../Models/User.js";

const router = express.Router();

//Register new user router.post("/register", registerUser);

//Login user and return JWT router.post("/login", loginUser);

// Get currently logged in user router.get("/me", authMiddleware,
async (req, res) => {
    try { const user = await User.findById(req.user.id).select("-
password"); if (!user) return res.status(404).json({ error: "User not
found" }); res.json(user); } catch (err) { console.error("/me
error:", err); res.status(500).json({ error: "Failed to fetch user
info" });
    }
});

export default router;

```

### **Frontend Code:-**

App.jsx

```

import { Route, BrowserRouter as Router, Routes } from "react-router-dom"; import
Header from "../components/Header"; import Home from "../pages/Home"; import
VideoPlayer from "../pages/VideoPlayer"; import Channel from

```



```

"./pages/Channel"; import Login from "./pages/Login"; import Register from
"./pages/Register"; import NotFound from "./pages/NotFound"; import { useEffect, useState
} from "react"; import Sidebar from "./components/Sidebar"; import
ProtectedRoute from "./routes/ProtectedRoute"; import UploadVideo from
"./pages/UploadVideo"; import EditVideo from "./pages/EditVideo";

```

```

function App() { const [isSidebarOpen, setIsSidebarOpen] = useState(true);

```

```

    useEffect(() => { const handleResize
= () => { if (window.innerWidth <
768) { setIsSidebarOpen(false);
    } else {
setIsSidebarOpen(true);
    }
  };

```

```

    handleResize(); // run once on mount
window.addEventListener("resize", handleResize); return () =>
window.removeEventListener("resize", handleResize);
}, []);

```

```

return (
  <Router>
    <Header toggleSidebar={() => setIsSidebarOpen((prev) => !prev)} />
    <div className="flex relative">
      <Sidebar isOpen={isSidebarOpen} />
      <div className="flex-1 transition-all duration-300">
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/video/:id"
element={
            <ProtectedRoute>
              {" "}
              <VideoPlayer />
            </ProtectedRoute>
          }
        />

```

```

        <Route
path="/channel/:channelId"          element={
    <ProtectedRoute>
      <Channel />
    </ProtectedRoute>
  }
/>

<Route path="/login" element={<Login />} />
<Route path="/register" element={<Register />} />
<Route path="/upload" element={<UploadVideo />} />
<Route path="/edit-video/:videoId" element={<EditVideo />} />
<Route path="*" element={<NotFound />} />

</Routes>
</div>
</div>
</Router>
);
}

```

```
export default App;
```

### Header.jsx

```

import { HiMenu } from "react-icons/hi"; import { Link }
from "react-router-dom"; import { IoLogoYoutube } from
"react-icons/io"; import { FiSearch } from "reacticons/fi";
import { MdAccountCircle } from "reacticons/md";
import { useContext } from "react"; import { AuthContext
} from "../context/AuthContext"; import { useVideo }
from "../context/VideoContext";

function Header({ toggleSidebar }) { const { user, logout } =
useContext(AuthContext); const { search, setSearch, handleSearch
} = useVideo();

return (
  <header className="flex items-center justify-between px-6 py-3 bg-white shadow-md sticky top-0 z100">

```

```

<div className="flex item-center gap-4">
  <button onClick={toggleSidebar} className="text-2xl">
    <HiMenu />
  </button>
  <Link to="/" className="flex items-center gap-1 text-xl font-bold text-shadow-lg">
    >
    <IoLogoYoutube className="text-2xl text-red-600" />
    YouTube
  </Link>
</div>

{/* Search Bar */}
<div className="hidden md:flex justify-center flex-1 mx-4">
  <input type="text" placeholder="Search..."
value={search} onChange={(e) => setSearch(e.target.value)}
className="w-1/2 border rounded-l-xl py-1 px-4"
  />
  <button onClick={handleSearch} className="bg-gray-200
px-4 py-1 rounded-r-xl"
  >
    <FiSearch />
  </button>
</div>

{/* Auth Section */}
<div className="flex items-center gap-3">
  <!user ? (
<Link to="/login"
  className="flex items-center gap-1 text-blue-500 text-sm font-semibold bg-gray-100 hover:bggray-300 px-3
py-1 rounded-md"
  >
    <MdAccountCircle className="text-2xl" /> Sign In
  </Link>
) : (
  <
    <span className="flex items-center gap-1 text-sm font-medium text-gray-700 bg-zinc-300 py-1 px-4 rounded">
      <MdAccountCircle className="text-2xl" /> {user.username}

```

```

        </span>        <button        onClick={logout}        className="text-sm font-semibold bg-red-500
hover:bg-red-700 text-white px-3 py-1 rounded-md"
        >
            Logout
        </button>
    </>
    )}
</div>
</header>
);
}

```

```
export default Header;
```

### Home.jsx

```

import { useEffect, useState } from "react"; import
VideoCard from "../components/VideoCard"; import {
useLocation } from "react-router-dom"; import { useVideo
} from "../context/VideoContext";

function Home() {  const categories
= [
    "All",
    "Movies",
    "Music",
    "TV Show",
    "Gaming",
    "Education",
    "News",
    "Sports",
    "Travel",
    "Food",
    "Other",
];

    const [activeCategory, setActiveCategory] = useState("All");  const { videos,
filteredVideos, setFilteredVideos } = useVideo();

```

```

const location = useLocation(); const searchParams = new
URLSearchParams(location.search); const search = searchParams.get("search")?.toLowerCase()
|| "";

```

```

useEffect(() => { let filtered
= [...videos];

```

```

    if (activeCategory !== "All") { filtered = filtered.filter( (video) =>
video.category?.toLowerCase() === activeCategory.toLowerCase()
    );
}

```

```

    if (search) { filtered = filtered.filter((video)
=> video.title.toLowerCase().includes(search)
    );
}

```

```

    setFilteredVideos(filtered);
}, [videos, activeCategory, search]);

```

```

return (
<div className="flex">
  <main className="flex-1 p-4 ml-0">
    {/* Category Filters */}
    <div className="flex justify-center flex-wrap gap-2 mb-4">
      {categories.map((cat) => (
        <button key={cat} onClick={() =>
setActiveCategory(cat)} className={`px-4 py-1
textsm rounded-full ${activeCategory === cat
? "bg-zinc-900 text-white font-semibold"
: "bg-gray-200 hover:bg-gray-300"}
        `}
      >
        {cat}
      </button>
      )
    )}
  </div>
  <div>
    {cat}
  </div>
</div>
)}

```

```

</div>

    {/* Video Grid */}
    {filteredVideos.length === 0 ? (
      <p className="text-center text-gray-500">No videos found.</p>
    ) : (
      <div className="grid m-6 sm:m-0 gap-4 grid-cols-1 sm:grid-cols-2 md:grid-cols-2 lg:grid-cols-3">
        {filteredVideos.map((video) => (
          <VideoCard key={video._id} video={video} />
        ))}
      </div>
    )}
  </main>
</div>
);
}

export default Home;

```

### UploadVideo.jsx

```

import { useContext, useState } from "react"; import {
  useNavigate } from "react-router-dom"; import {
  AuthContext } from "../context/AuthContext"; import axios
  from "../api/axios";

function UploadVideo() {  const { user } =
  useContext(AuthContext);  const navigate = useNavigate();

  const [formData, setFormData] = useState({
    title: "",  description:
    "",  category: "",
    videoUrl: "",
    thumbnailUrl: "",
  });

  const [error, setError] = useState("");

```

```

const handleChange = (e) => {  setFormData((prev) => ({ ...prev,
[e.target.name]: e.target.value }));
};

const generateVideoId = () => {  return
"vid_" + Date.now().toString(36);
};

const handleSubmit = async (e) => {
  e.preventDefault();

  const token = localStorage.getItem("token");  const channelId = localStorage.getItem("channelId");
// stored after login/channel creation  console.log("Uploading to channelId:", channelId);  if
(!channelId) {  setError("No channel found. Please create a channel
first.");
  return;
}

  const payload = {  videoId:
generateVideoId(),  title: formData.title,
description: formData.description,  thumbnailUrl:
formData.thumbnailUrl,  videoUrl:
formData.videoUrl,  channelId,  category:
formData.category || "Other",
};  try {  await
axios.post("/videos", payload, {
headers: {
  Authorization: `JWT ${token}`,
},
});
  navigate("/channel/my");
} catch (err) {  setError(err.response?.data?.error ||
"Upload failed");
}
  console.log("Uploading:", payload);
};

if (!user) return <p className="text-center mt-10">Please login first.</p>;

```

```

return (
  <div className="max-w-2xl mx-auto mt-10 p-6 bg-white rounded shadow">
    <h2 className="text-2xl font-bold mb-6">Upload New Video</h2>
    {error && <p className="text-red-500 mb-4">{error}</p>}

    <form onSubmit={handleSubmit} className="space-y-4">
      <input
        name="title"
        placeholder="Video Title"
        value={formData.title}
        onChange={handleChange}
        required
        className="w-full border p-2
rounded"
        />
      <textarea
        name="description"
        placeholder="Video Description"
        value={formData.description}
        onChange={handleChange}
        required
        className="w-full border p-2
rounded"
        />
      { /* Category Dropdown */ }
      <select
        name="category"
        value={formData.category}
        onChange={handleChange}
        required
        className="w-full border p-2 rounded bg-white"
        >
        <option value="">Select a Category</option>
        <option value="Movies">Movies</option>
        <option value="Music">Music</option>
        <option value="TV Show">TV Show</option>
        <option value="Gaming">Gaming</option>
        <option value="Education">Education</option>
        <option value="News">News</option>
        <option value="Sports">Sports</option>
        <option value="Travel">Travel</option>
        <option value="Food">Food</option>
        <option value="Other">Other</option>
      </select>
      <input
        name="videoUrl"
        placeholder="Video

```



```

URL"      value={formData.videoUrl}
onChange={handleChange}
      required      className="w-full border p-2
rounded"
    />
    <input      name="thumbnailUrl"
placeholder="Thumbnail URL"
value={formData.thumbnailUrl}      onChange={handleChange}
      required      className="w-full border p-2
rounded"

    />
    <button      type="submit"      className="w-full bg-blue-600 text-white py-2 rounded
hover:bg-blue-800"
    >
      Upload Video
    </button>
  </form>
</div>
);
}

export default UploadVideo;

```

### Sidebar.jsx

```

import { Link } from "react-router-dom"; import { IoMdHome
} from "react-icons/io"; import { MdExplore } from "react-
icons/md"; import { RiWechatChannelsLine } from "react-
icons/ri";

function Sidebar({ isOpen }) { return (  <aside      className={`      bg-gray-100 min-hscreen
top-[55px] z-50 overflow-hidden transition-all duration-300      fixed md:static
      ${isOpen ? "w-48" : "w-0"}
      ${isOpen && window.innerWidth < 768 ? "left-0 absolute shadow-lg" : ""}
    `}
    >
      {" "}    <nav      className={`flex flexcol
gap-4 p-4 ${      isOpen ? "opacity-100"

```

```

: "opacity-0"
  } transition-opacity duration-300`}
  > <Link to="/" className="hover:bg-gray-300 p-1
rounded flex items-center gap-2"
  >
    <IoMdHome className="text-2xl" /> Home
  </Link> <Link to="/channel/my" className="hover:bggray-300
p-1 rounded flex items-center gap-2"
  >
    <RiWechatChannelsLine className="text-2xl" /> My Channel
  </Link>
  <Link className="hover:bg-gray-300 p-1 rounded flex items-center gap-2">
    <MdExplore className="text-2xl" /> Explore
  </Link>
</nav>
</aside>
);
}

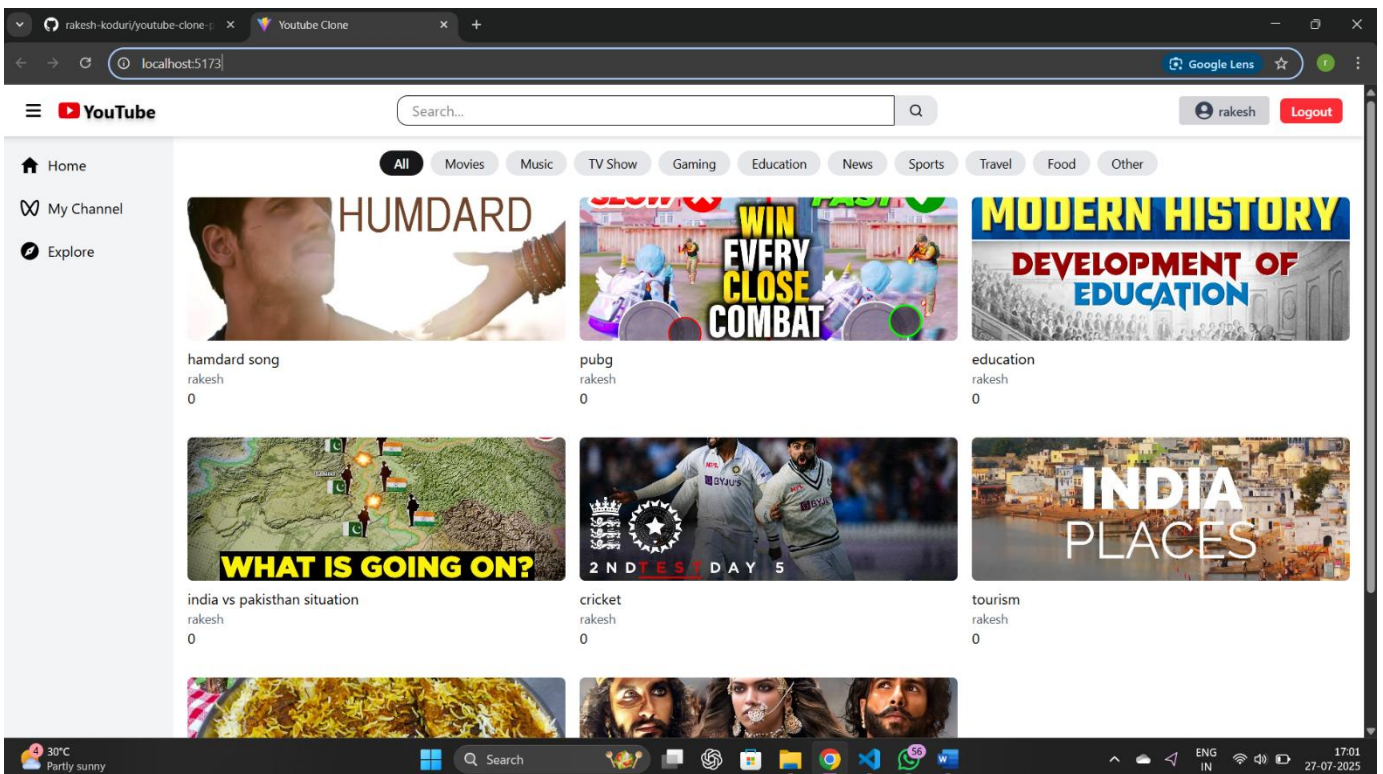
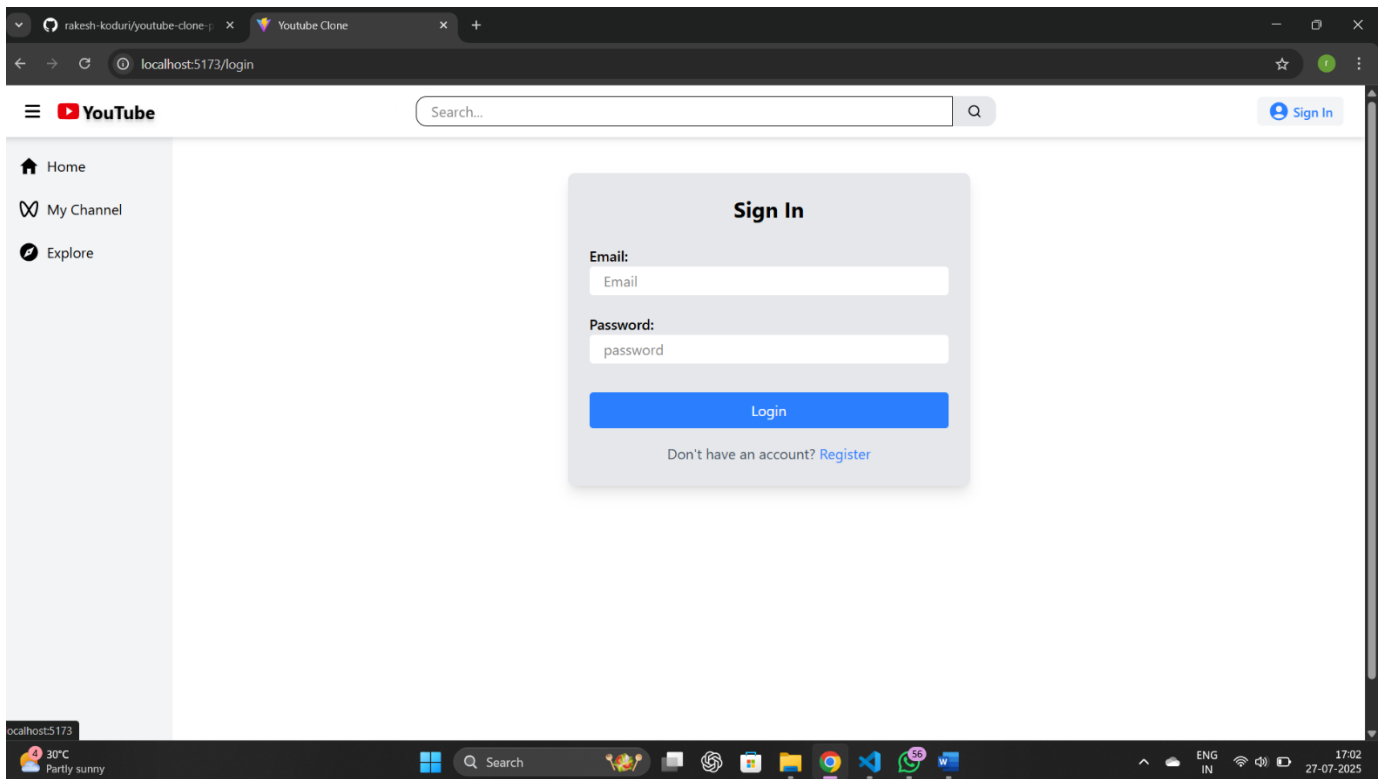
```

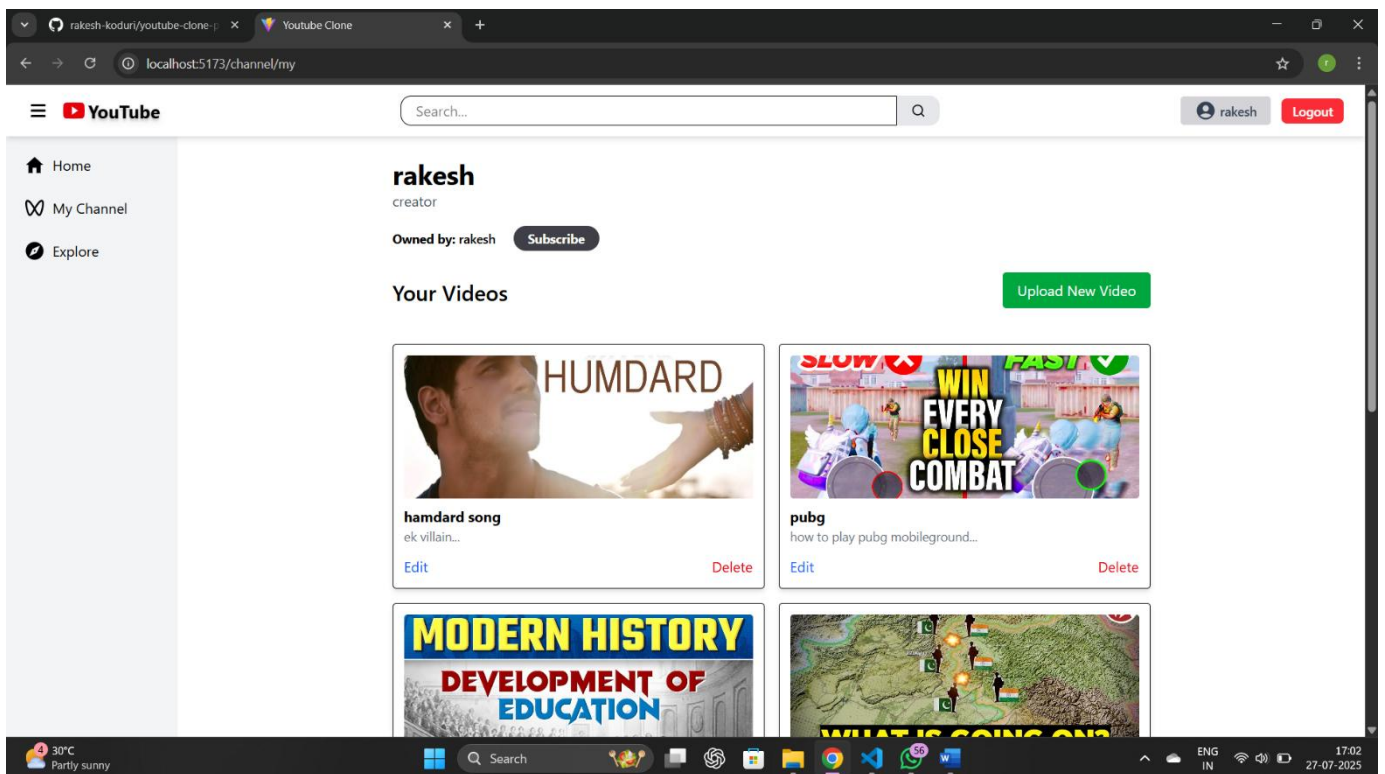
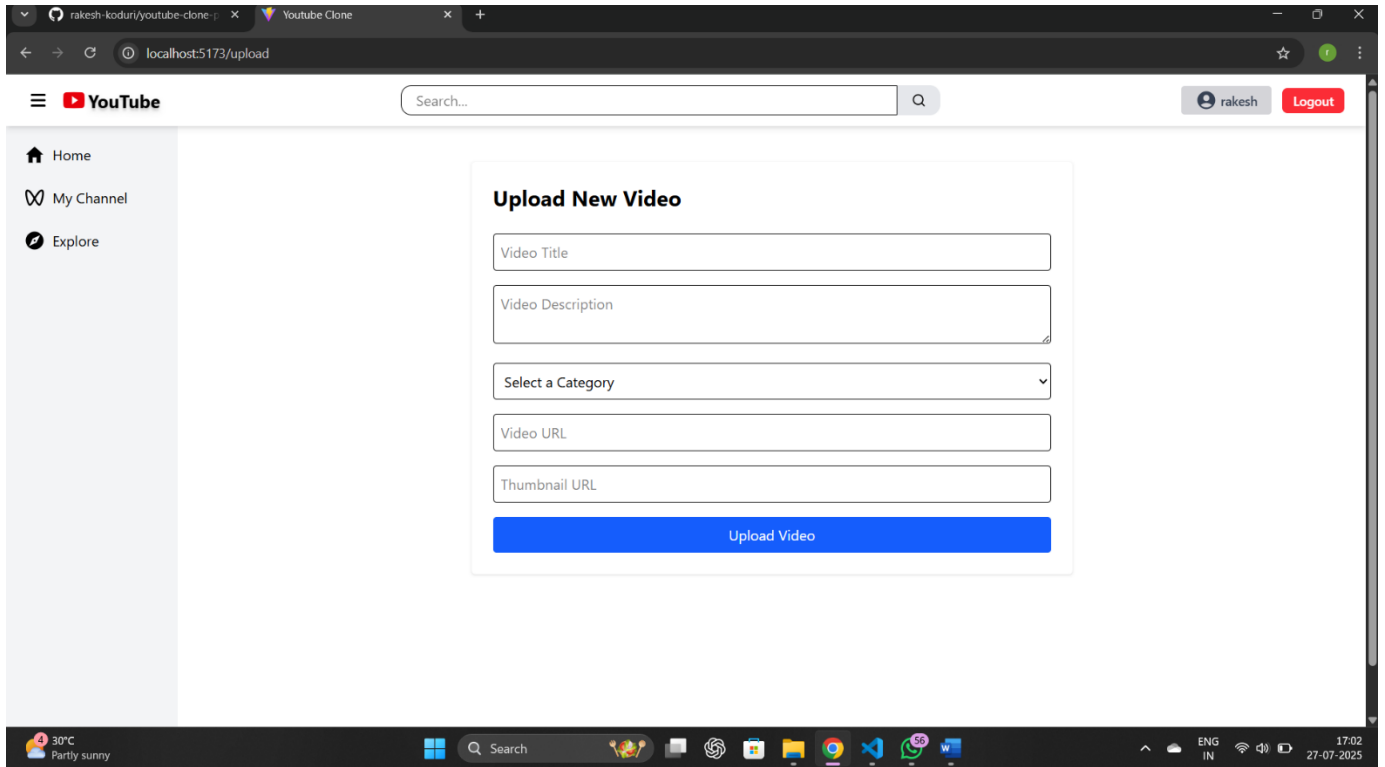
```

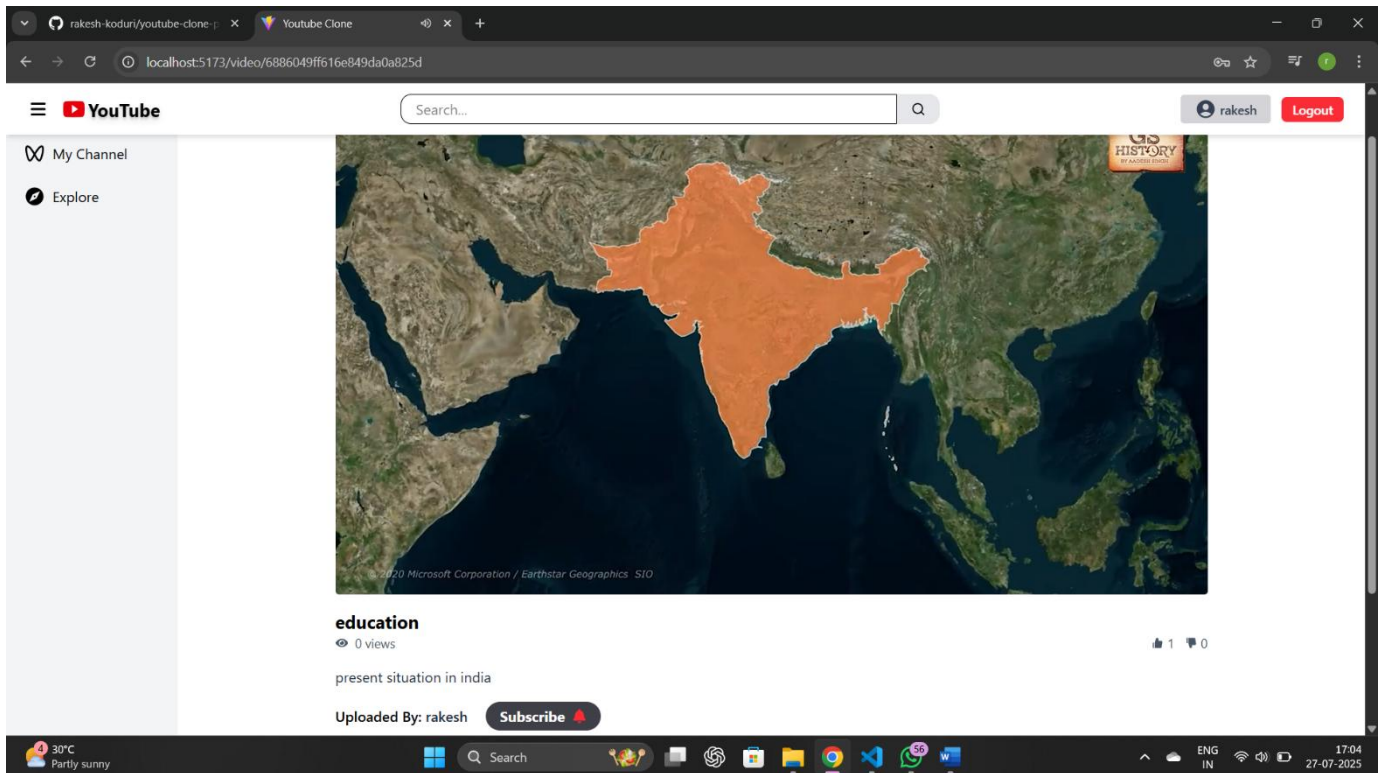
export default Sidebar;

```

## Screenshot :-







GitHub Link:- <https://github.com/rakesh-koduri/youtube-clone-project>