

Project Title:

Capstone Project: Develop a YouTube Clone Using the MERN Stack

Project Description

This project is a YouTube-like video sharing platform built using the MERN stack (MongoDB, Express, React, Node.js).

Users can authenticate, create channels, upload videos, browse videos by category, and interact through likes, dislikes, and comments.

The application follows RESTful API architecture, JWT-based authentication, and responsive UI design for mobile, tablet, and desktop devices.

Tech Stack

Frontend

- React.js
- Axios
- CSS (Responsive Design)

Backend

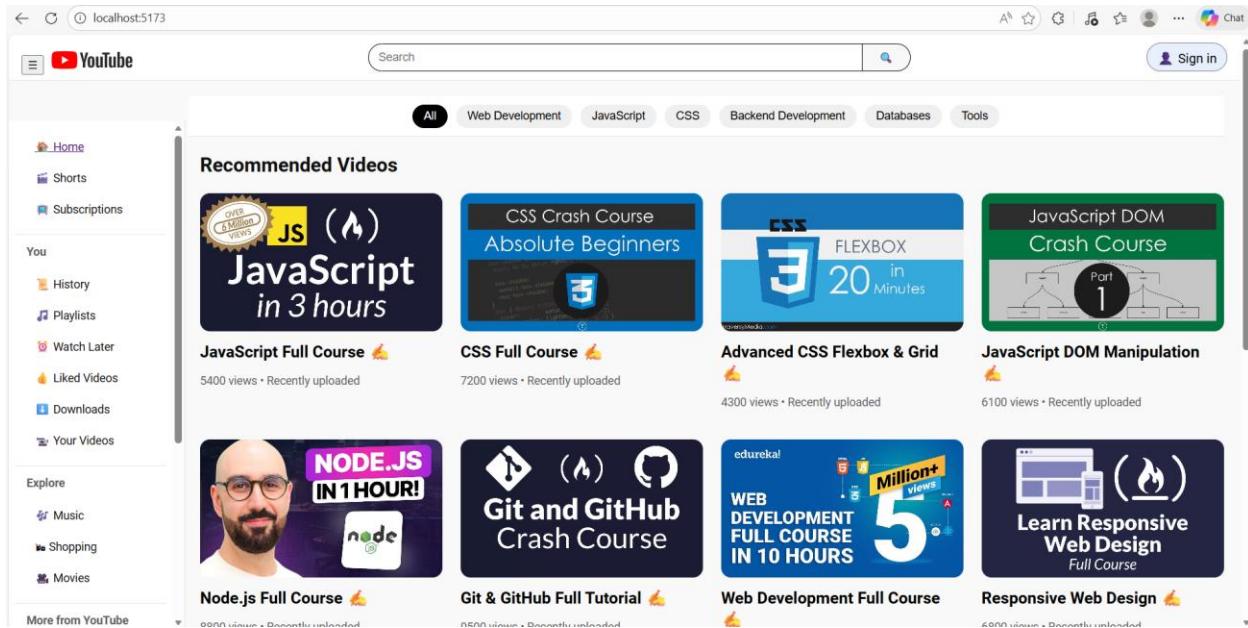
- Node.js
- Express.js
- MongoDB & Mongoose
- JWT Authentication

Tools

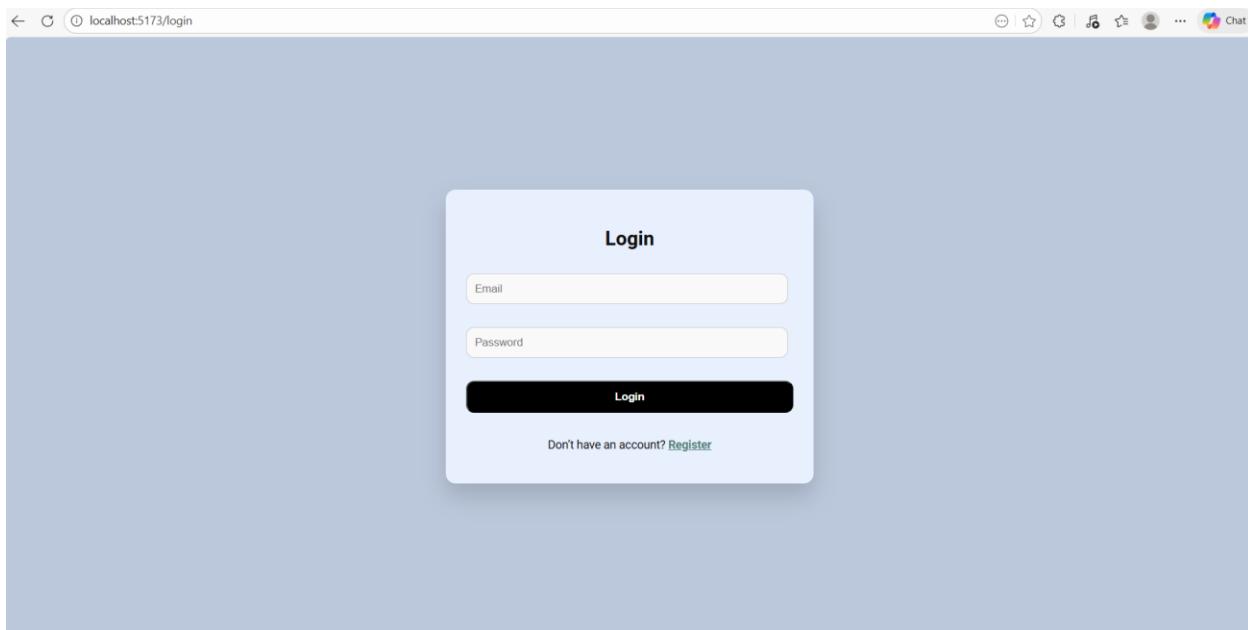
- Thunder Client (API testing)
- MongoDB Compass
- Git & GitHub

UI Screenshots:

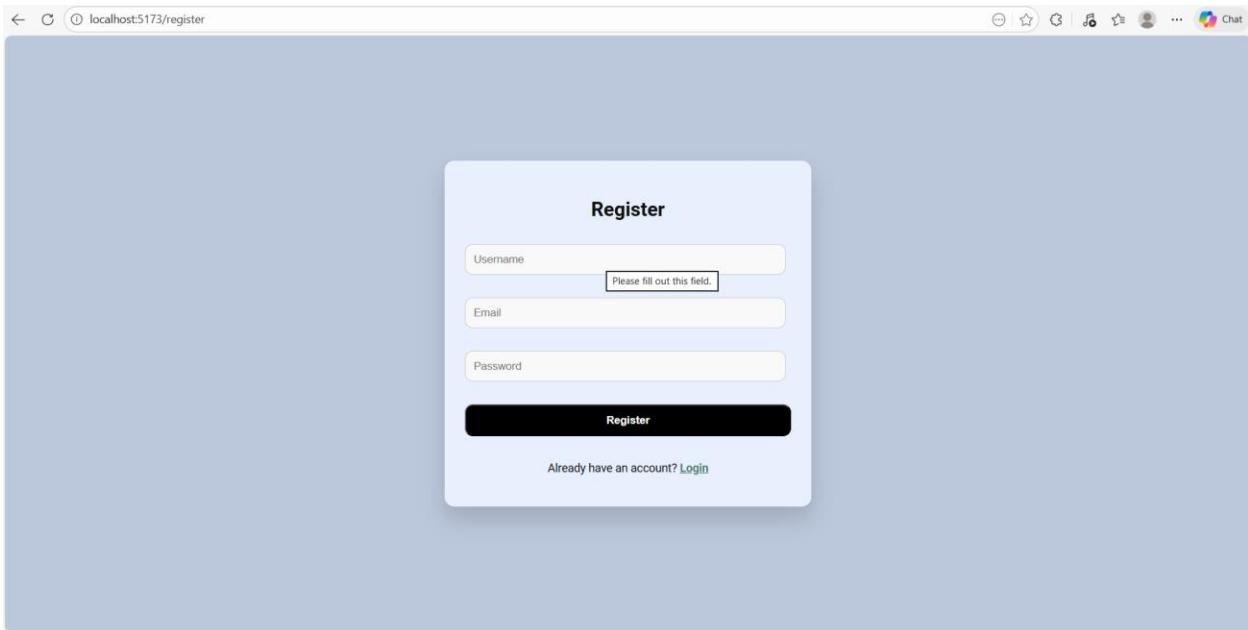
Homepage:



Login:



Register:



Uploaded Videos with Delete Button:

A screenshot of a YouTube-like application interface. On the left is a sidebar with links like Home, Shorts, Subscriptions, History, Playlists, Watch Later, Liked Videos, Downloads, and Your Videos. The main area shows a search bar at the top. Below it is a "Category (Web Dev, JS, CSS...)" dropdown and a "Description" input field. A large "Upload" button is centered. Underneath is a section titled "Your Videos" showing two video thumbnails. The first is for "RICK ASTLEY NEVER GONNA GIVE YOU UP" and the second is for "Full Stack Development Basics". Each video thumbnail has a "Delete" button below it.

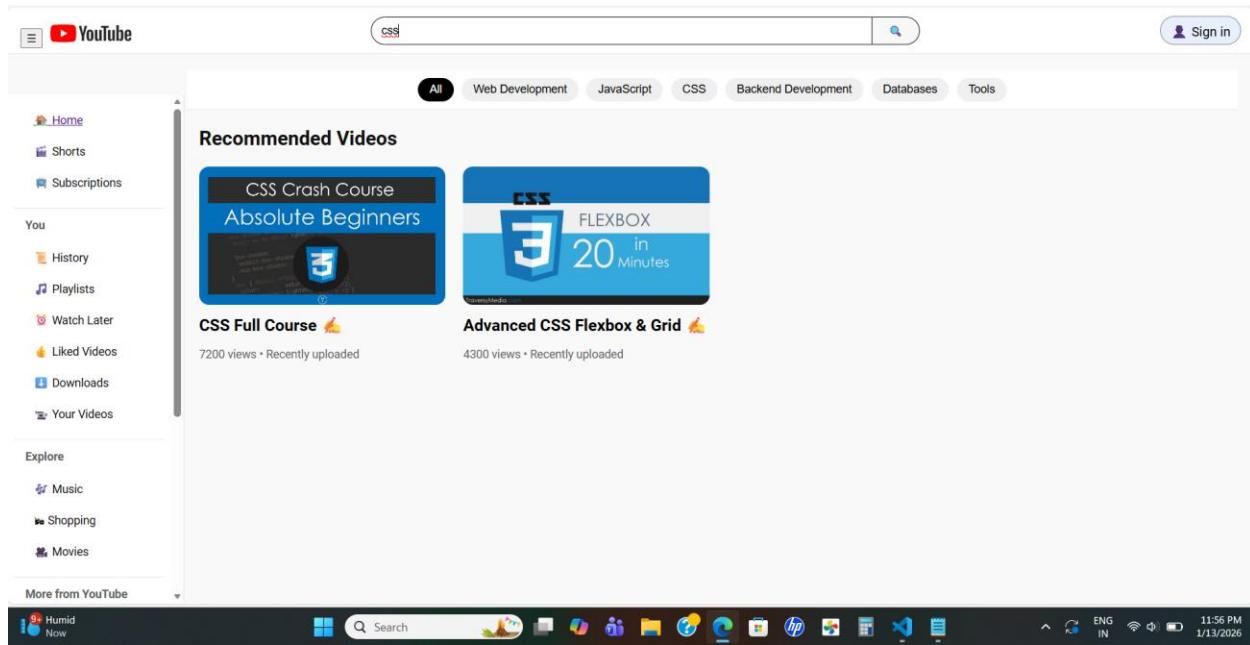
Filter by Category:

A screenshot of a YouTube channel page. The left sidebar shows navigation links like Home, Shorts, Subscriptions, and Your Videos. The main content area features a "Recommended Videos" section with two video thumbnails. The first video is titled "Web Development Full Course" by edureka!, with a thumbnail showing a large number "5" and text "WEB DEVELOPMENT FULL COURSE IN 10 HOURS". The second video is titled "Learn Responsive Web Design" by (火烧), with a thumbnail showing a smartphone and a laptop. Both videos have their view counts and upload dates below them.

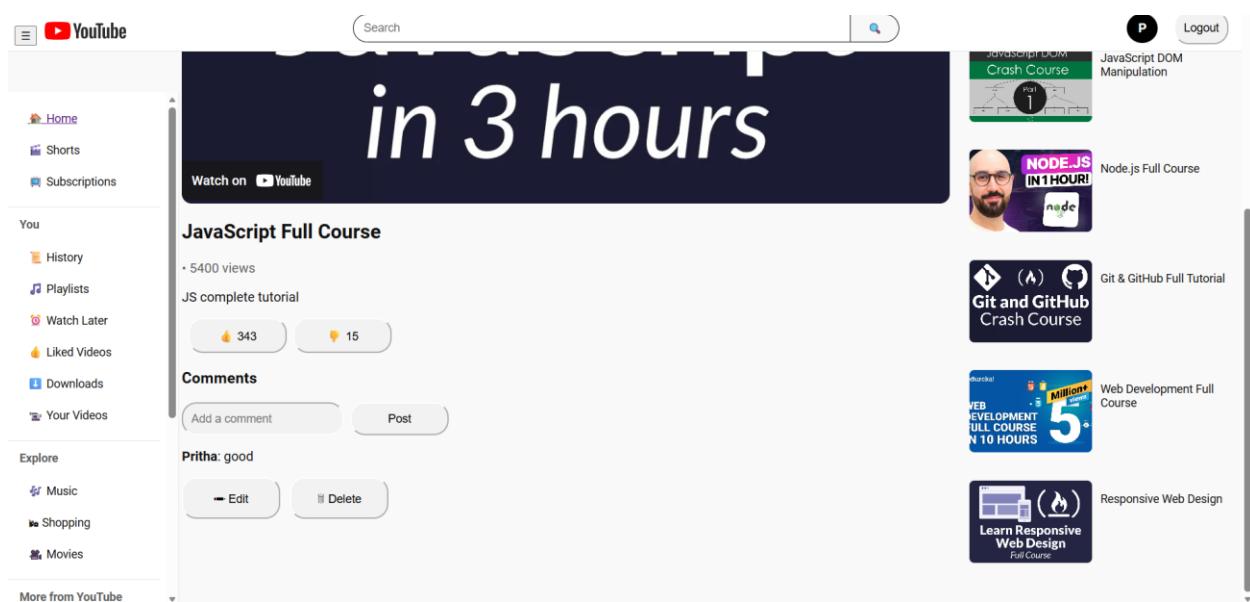
ChannelPage:

A screenshot of a YouTube channel page for "Code with Pritha". The channel's profile picture is a black circle with a white letter "C". The channel name "Code with Pritha" is displayed prominently in white text on a dark blue background. Below the channel name, it says "Learn Full Stack Development with hands-on tutorials on frontend, backend, databases, and tools to build real-world web applications." and "0 subscribers". At the bottom, there is a "Upload Video" section with fields for "Video Title" and "YouTube ID".

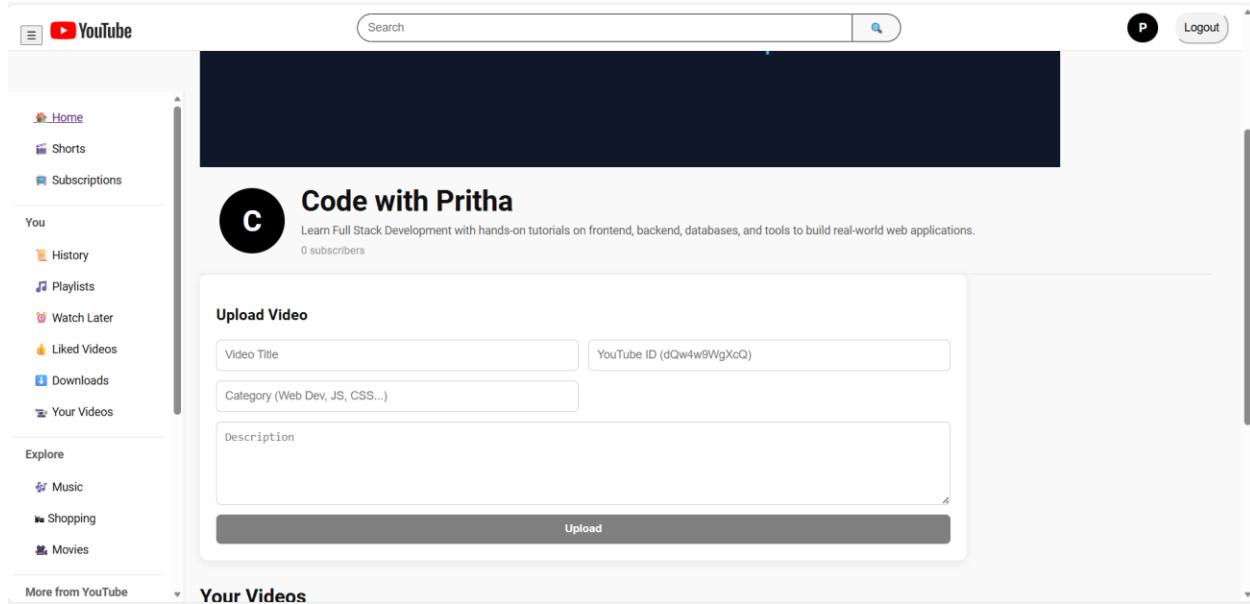
Search and filter:



Like,Dislike and Comment:



Upload video banner:



Terminal:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
>>
PS C:\Users\vendh\OneDrive\Desktop\youtube-clone\frontend\youtube-clone> npm
run dev

> youtube-clone@0.0.0 dev
> vite

VITE v7.3.1 ready in 1341 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help

[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server running on port 5000
MongoDB Connected
```

Description:

My Backend and Frontend is running Successful.

API Testing (Thunder Client)

All APIs were tested using **Thunder Client** inside VS Code.
Screenshots are included for verification.

User Registration API

Endpoint:

POST /api/auth/register

The screenshot shows a Postman request for a POST method to the URL `http://localhost:5000/api/auth/register`. The request body is a JSON object with fields `username`, `email`, and `password`. The response status is **201 Created**, size is **330 Bytes**, and time is **146 ms**. The response body contains a user object with additional fields like `id`, `createdAt`, `updatedAt`, and `v`.

```
1 {  
2   "message": "User registered successfully",  
3   "user": {  
4     "username": "cloneproject",  
5     "email": "cloneproject@gmail.com",  
6     "password": "$2b$10$MOTJ4uhrGzICs175Ld4iIOX3  
7       .Poquq8CmFjuVxEelE7QKG050TK8a",  
8     "avatar": "",  
9     "channel": null,  
10    "id": "6967a60cef91935982a61c1b",  
11    "createdAt": "2026-01-14T14:19:56.998Z",  
12    "updatedAt": "2026-01-14T14:19:56.998Z",  
13    "v": 0  
14  }  
15 }
```

Description:

Registers a new user by providing name, email, and password.

User Login API

Endpoint:

POST /api/auth/login

The screenshot shows a Postman request for a POST method to the URL `http://localhost:5000/api/auth/login`. The request body is a JSON object with fields `email` and `password`. The response status is **200 OK**, size is **283 Bytes**, and time is **212 ms**. The response body includes a JWT token and a user object with fields `id`, `username`, `email`, and `v`.

```
1 {  
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9  
3     .eyJpZC16Ijy5NjdhnBjZW5MTkzNTk4Mme2MwNyIisImhdCI6MTc20DQwMDU1  
4     .SwizXhwIjoxNzY5MDA1MzU5fQ.lSBtVe_W_tjECUqb4i7IAxSqEH8  
5     -EyewOMJH2WIWckM",  
6   "user": {  
7     "id": "6967a60cef91935982a61c1b",  
8     "username": "cloneproject",  
9     "email": "cloneproject@gmail.com"  
10    }  
11  }  
12 }
```

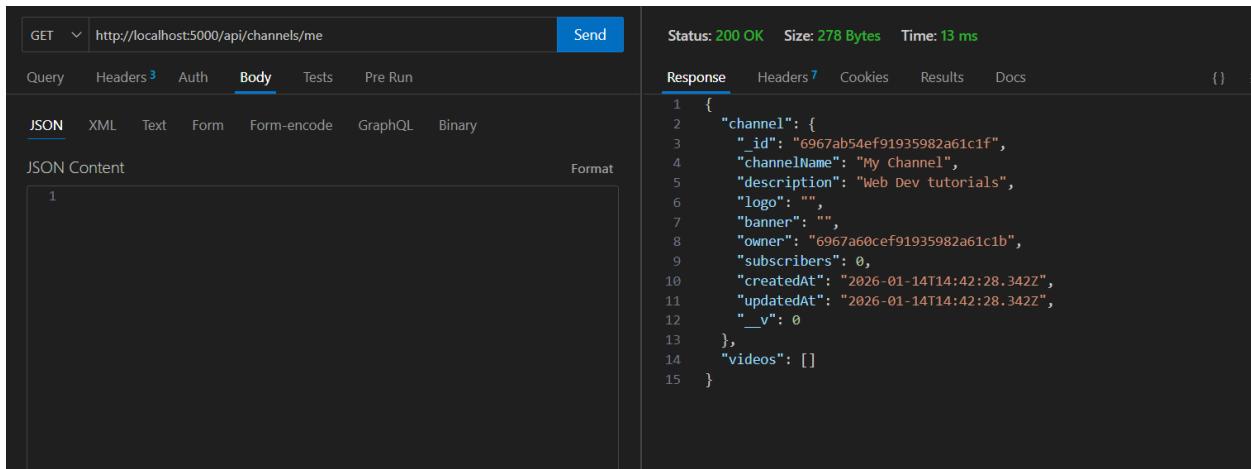
Description:

Authenticates the user and returns a JWT token used for protected routes.

Get My Channel

Endpoint:

GET /api/channels/me



The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:5000/api/channels/me`. The response status is `200 OK`, size is `278 Bytes`, and time taken is `13 ms`. The response body is a JSON object representing a channel:

```
1 {  
2   "channel": {  
3     "_id": "6967ab54ef91935982a61c1f",  
4     "channelName": "My Channel",  
5     "description": "Web Dev tutorials",  
6     "logo": "",  
7     "banner": "",  
8     "owner": "6967a60cef91935982a61c1b",  
9     "subscribers": 0,  
10    "createdAt": "2026-01-14T14:42:28.342Z",  
11    "updatedAt": "2026-01-14T14:42:28.342Z",  
12    "__v": 0  
13  },  
14  "videos": []  
15 }
```

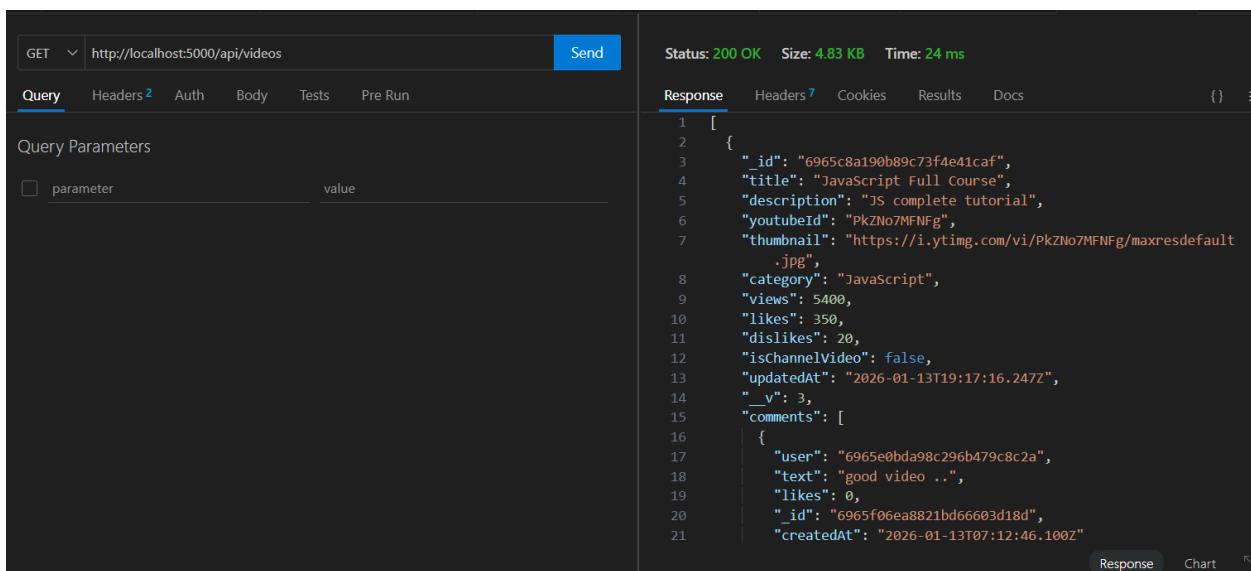
Description:

Fetches the logged-in user's channel and uploaded videos.

GET Uploaded Video API

Endpoint:

GET /api/videos



The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:5000/api/videos`. The response status is `200 OK`, size is `4.83 KB`, and time taken is `24 ms`. The response body is a JSON array of video objects:

```
1 [  
2   {  
3     "_id": "6965c8a190b89c73f4e41caf",  
4     "title": "JavaScript Full Course",  
5     "description": "JS complete tutorial",  
6     "youtubeID": "PkZN07MFNFg",  
7     "thumbnail": "https://i.ytimg.com/vi/PkZN07MFNFg/maxresdefault.jpg",  
8     "category": "JavaScript",  
9     "views": 5400,  
10    "likes": 350,  
11    "dislikes": 20,  
12    "isChannelVideo": false,  
13    "updatedAt": "2026-01-13T19:17:16.247Z",  
14    "__v": 3,  
15    "comments": [  
16      {  
17        "user": "6965e0bda98c296b479c8c2a",  
18        "text": "good video ..",  
19        "likes": 0,  
20        "id": "6965f06ea8821bd66603d18d",  
21        "createdAt": "2026-01-13T07:12:46.100Z"  
22      }  
23    ]  
24  ]
```

Description:

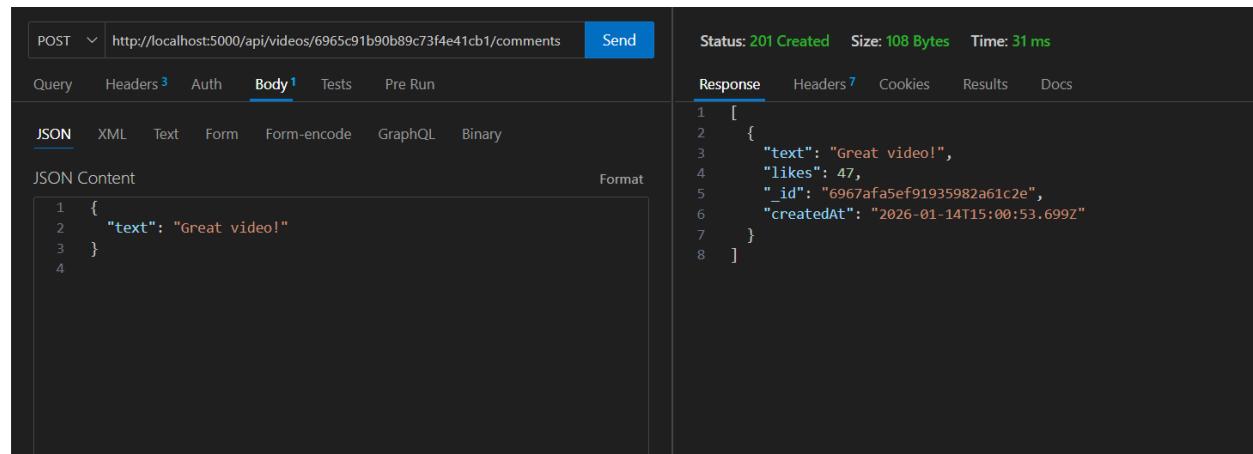
Get Uploaded video under the authenticated user's channel.

Channel ownership is derived from JWT (secure design).

Add Comment

Endpoint:

POST /api/comments/:videoId



The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: http://localhost:5000/api/videos/6965c91b90b89c73f4e41cb1/comments
- Body tab selected, showing JSON content:

```
1 {  
2   "text": "Great video!"  
3 }  
4
```
- Status: 201 Created
- Size: 108 Bytes
- Time: 31 ms
- Response tab selected, showing the JSON response:

```
1 [  
2   {  
3     "text": "Great video!",  
4     "likes": 47,  
5     "_id": "6967afa5ef91935982a61c2e",  
6     "createdAt": "2026-01-14T15:00:53.699Z"  
7   }  
8 ]
```

Description:

Adds a comment to a specific video.

Database (MongoDB)

- MongoDB Compass used locally
- Sample data exported and included in: db-exports/

Responsive Design

The application adapts to:

- Mobile screens
- Tablet screens
- Desktop screens

Media queries ensure layout consistency across devices.

Application Features

- User Authentication (JWT)
- Channel creation & management
- Video upload & listing
- Like / Dislike system
- Comment add / edit / delete
- Responsive UI (Mobile / Tablet / Desktop)
- Secure API access using middleware

Git & Commits

- Frontend and Backend developed separately
- Multiple meaningful commits
- Clear commit messages
- GitHub repository maintained properly