

Capstone Project:

Youtube Clone using MERN Stack

Submitted By :
Chaitali Mahato

Github : <https://github.com/chaitali9497/youtube-clone>

Video Demo :

<https://www.loom.com/share/10a99b9ca3094995a65c7d84fe2db01d>

<https://www.loom.com/share/e775e00266124035bc9dd ef53ac83980>

PROJECT OVERVIEW :

The **YouTube Clone Application** is a full-stack web application developed using the **MERN stack (MongoDB, Express.js, React.js, and Node.js)**. The project aims to replicate the core features and user experience of YouTube, allowing users to browse, watch, upload, and interact with video content in a scalable and responsive environment.

This application demonstrates the implementation of real-world full-stack concepts such as user authentication, RESTful APIs, database design, state management, and responsive UI development. It is designed to provide hands-on experience in building modern web applications with secure authentication and dynamic data handling.

Objective

The primary objective of this project is to build a functional video-sharing platform where users can:

- Register and authenticate securely
 - Browse and search for videos
 - Watch videos and interact through likes, dislikes, and comments
 - Create and manage their own channels
 - Upload, edit, and delete videos
-

Key Features

Frontend (React)

- Home Page**

- YouTube-style header with search bar
- Toggleable sidebar navigation
- Category-based filter buttons
- Grid layout displaying video thumbnails with title, channel name, views, and upload details

- User Authentication**

- User registration and login using email, username, and password
- JWT-based authentication
- Conditional UI rendering based on login status

- Video Player Page**

- Embedded video player
- Video title and description
- Comment section with full CRUD operations (add, edit, delete comments)

- Channel Page**

- Channel creation available only for authenticated users
- Display all videos uploaded by the channel
- Video management features (edit and delete videos)

- **Responsive Design**

- Fully responsive layout for desktop, tablet, and mobile devices

Backend (Node.js & Express)

- **RESTful APIs**

- User authentication (signup, login, JWT validation)
- Channel management APIs
- Video management APIs (CRUD operations)
- Comment management APIs

- **Authentication & Security**

- JSON Web Tokens (JWT) for secure authentication
- Protected routes for authorized users

Database (MongoDB)

- Stores structured data for:

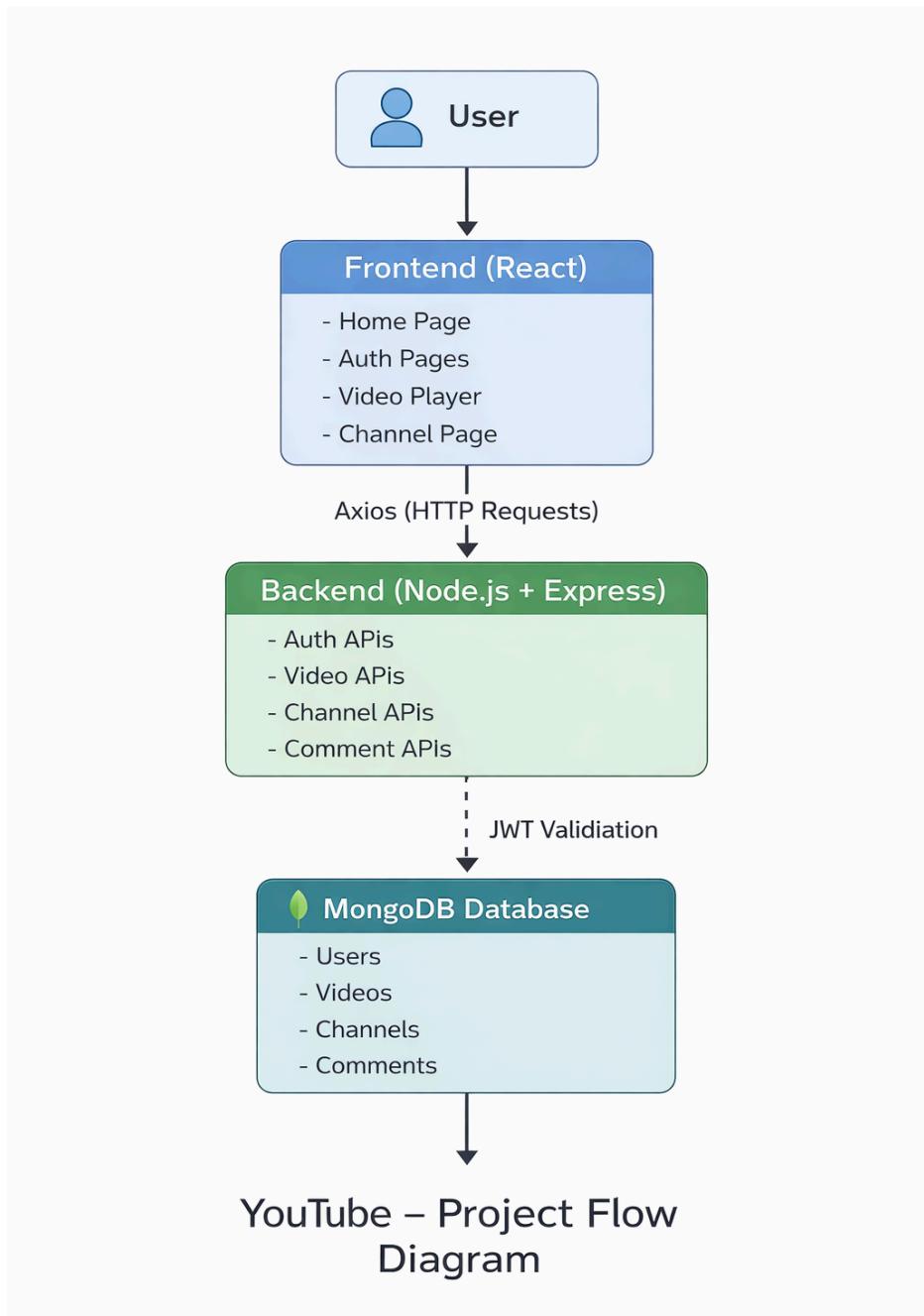
- Users
- Videos
- Channels
- Comments

- Handles video metadata such as video URLs, thumbnails, views, likes, and timestamps
-

Technologies Used

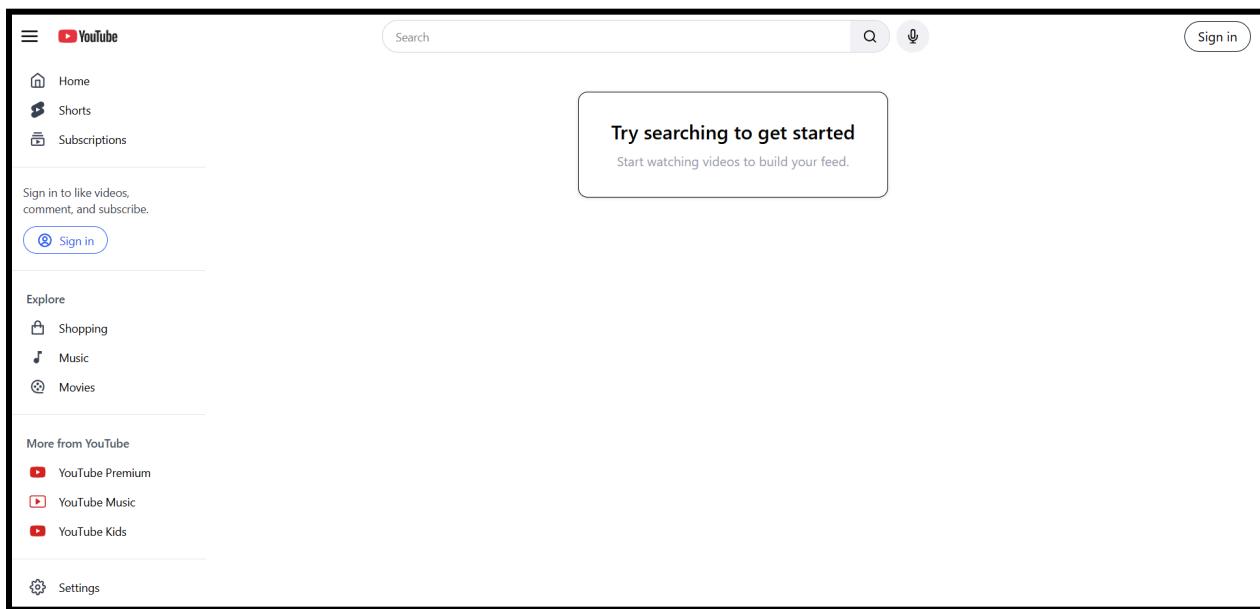
- **Frontend:** React, React Router, Axios
- **Backend:** Node.js, Express.js
- **Database:** MongoDB (MongoDB Atlas / Local)
- **Authentication:** JWT (JSON Web Tokens)
- **Version Control:** Git & GitHub
- **Development Tools:** Vite, Thunderclient

PROJECT FLOW :

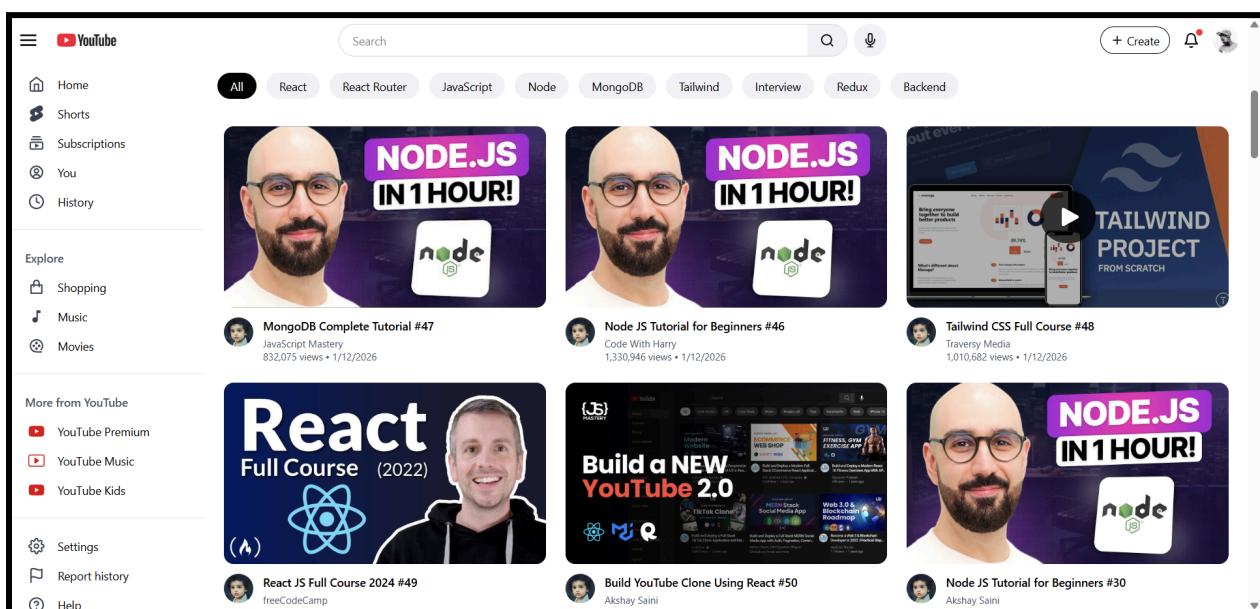


FrontEnd UI sneak peek:

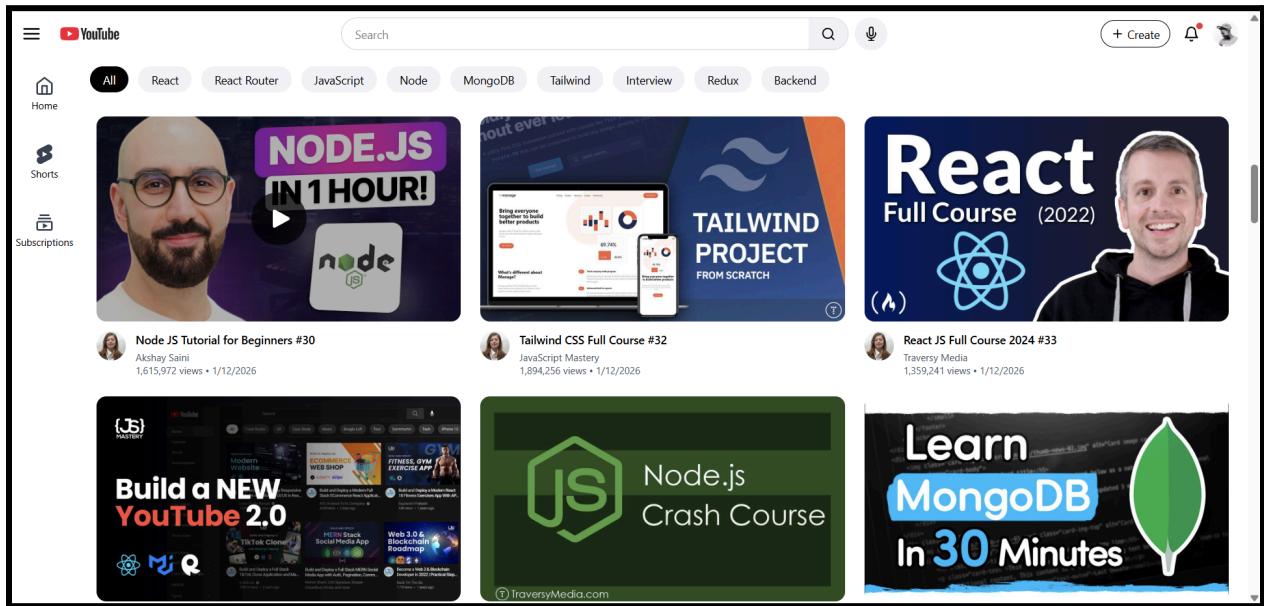
Unauthenticated user:



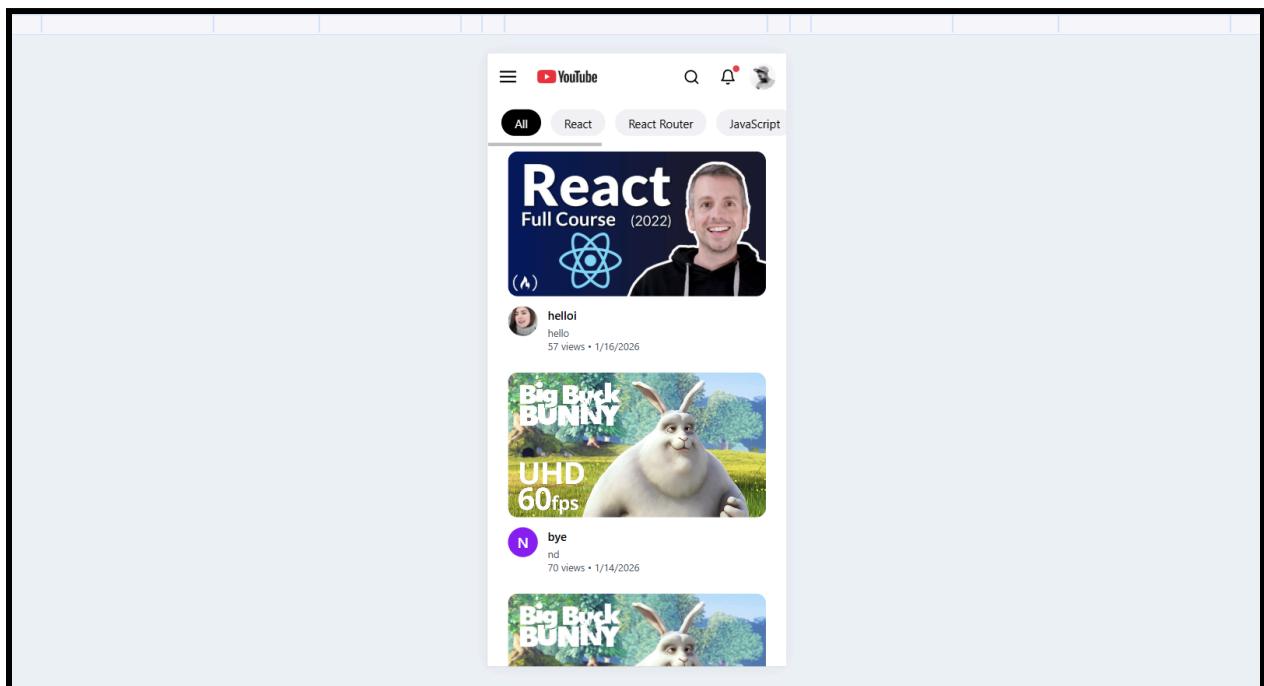
Authenticated user:

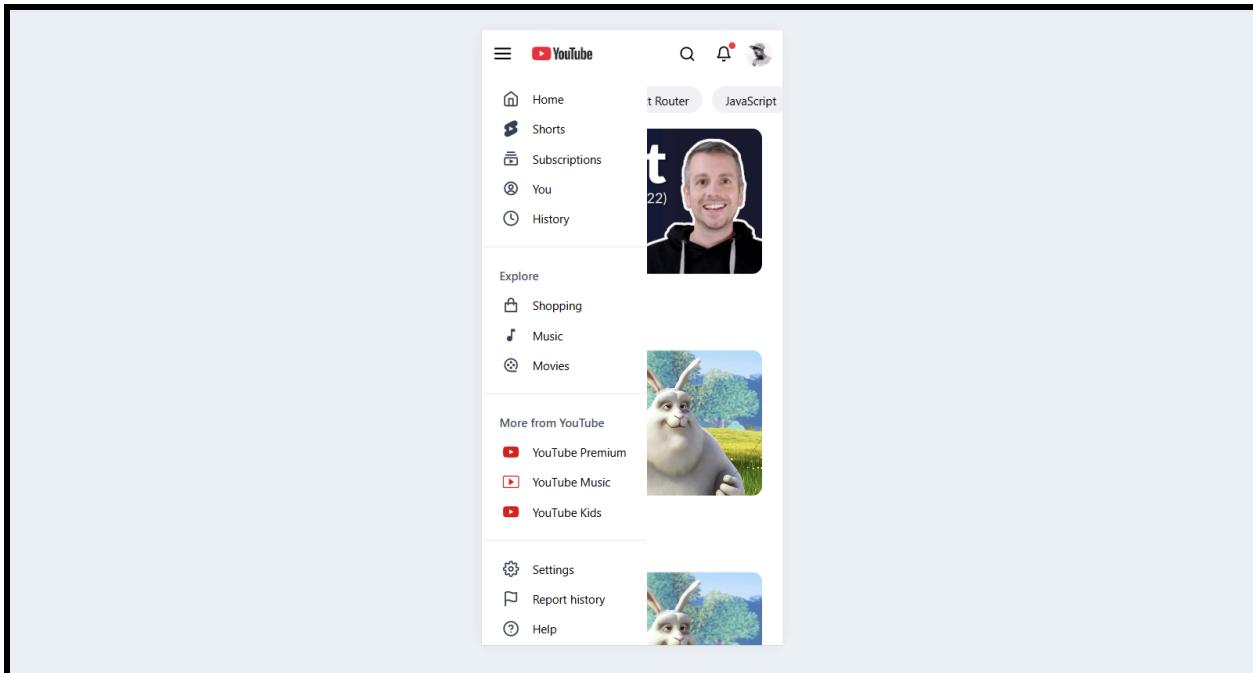


Toggled Menu:

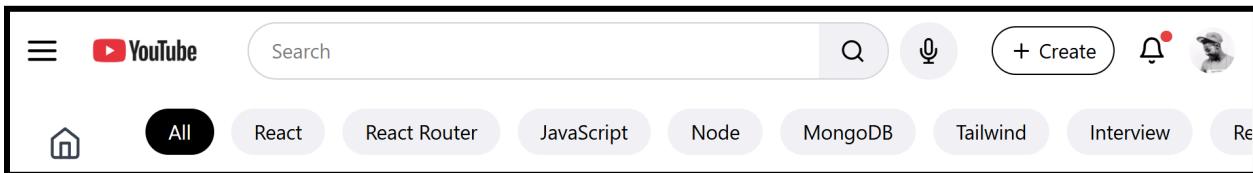


Responsive UI:

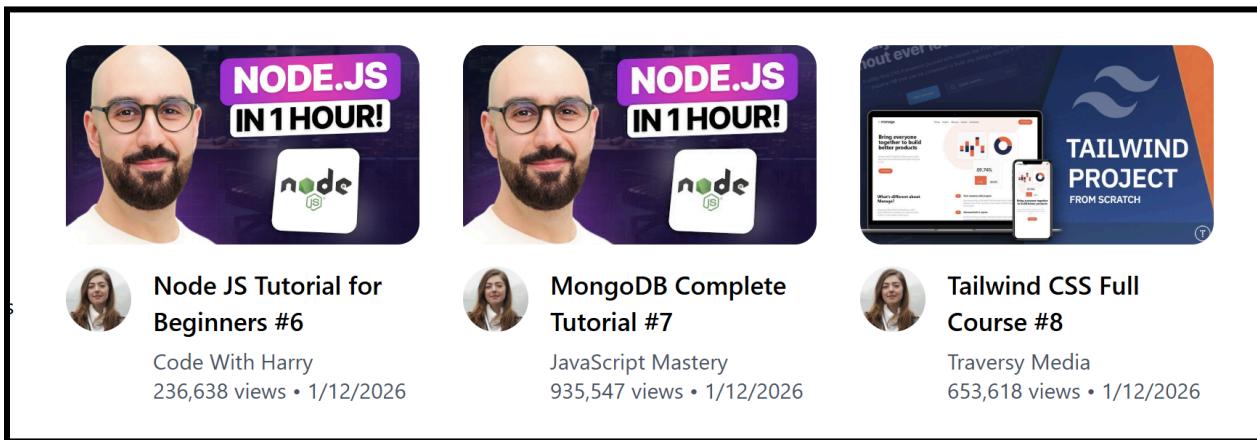




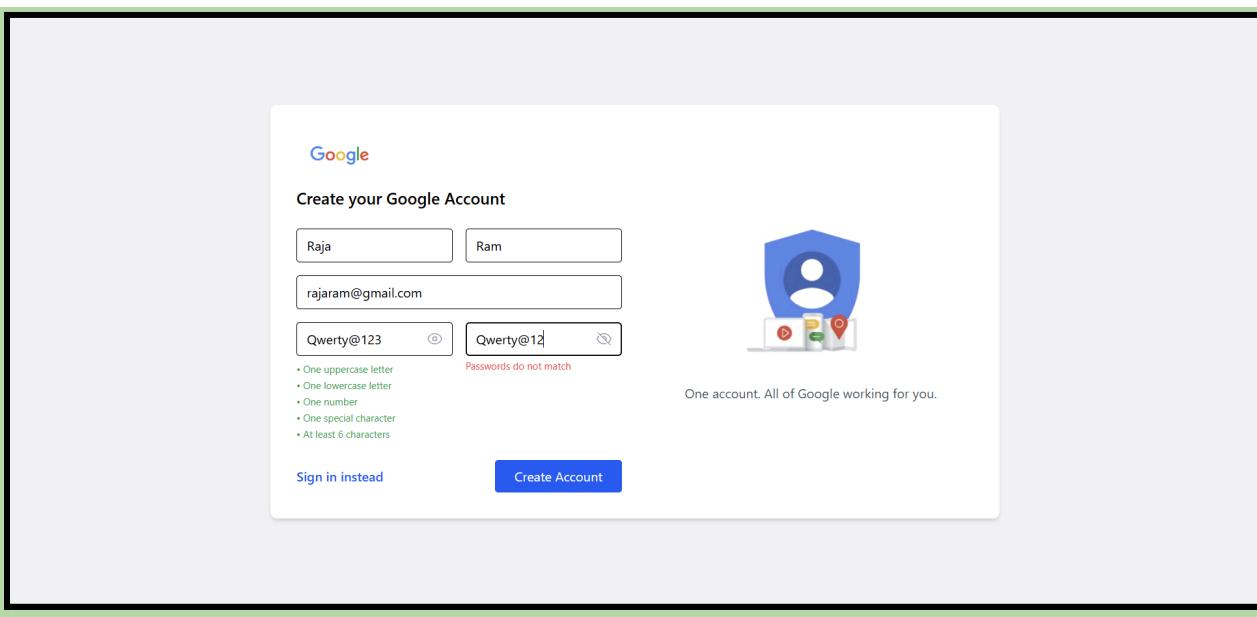
Header/ Hamburger/ Filter Buttons:



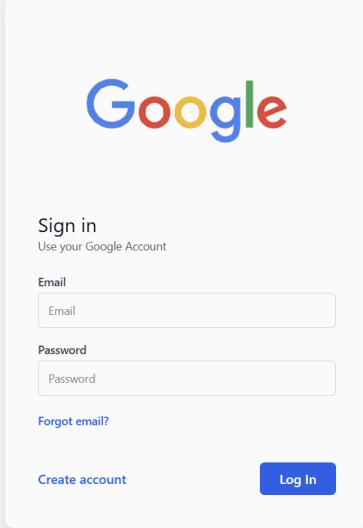
Thumbnail / Image/ Title/ Channel/ Views/ Date:



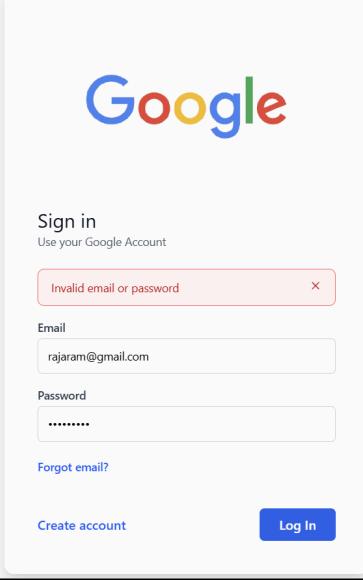
Create account page with validation:



Sign In Form:

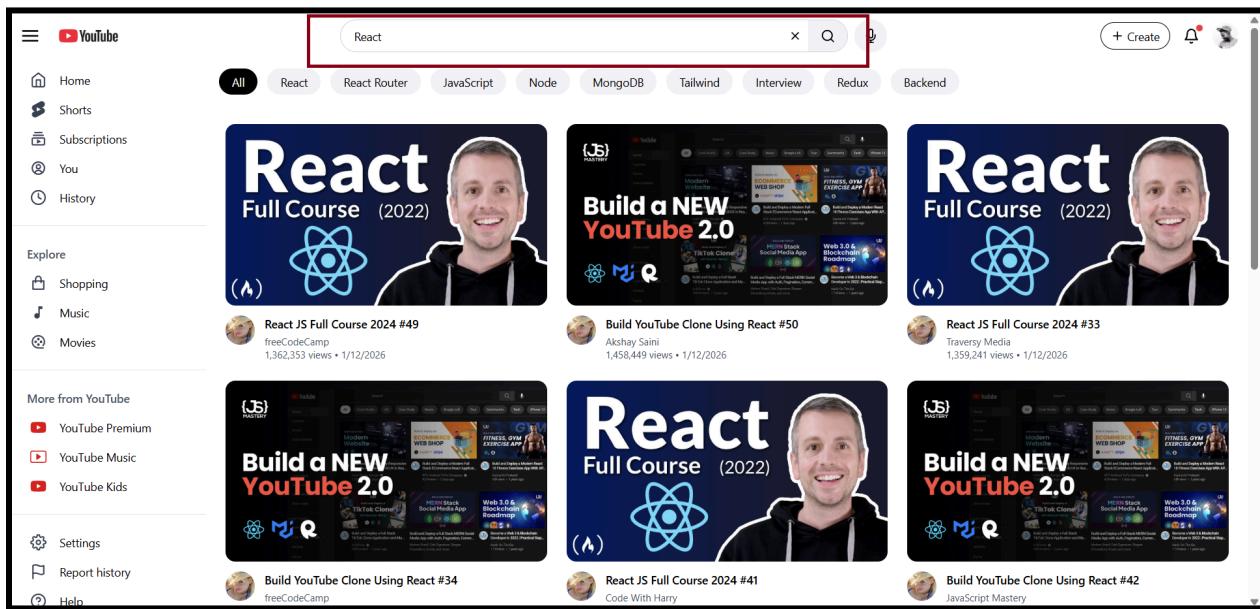


The image shows a standard Google sign-in page. At the top is the Google logo. Below it is the "Sign in" heading and a sub-instruction "Use your Google Account". There are two input fields: one for "Email" containing the placeholder "Email" and another for "Password" containing the placeholder "Password". Below these fields are links for "Forgot email?" and "Create account". To the right of the password field is a blue "Log In" button.

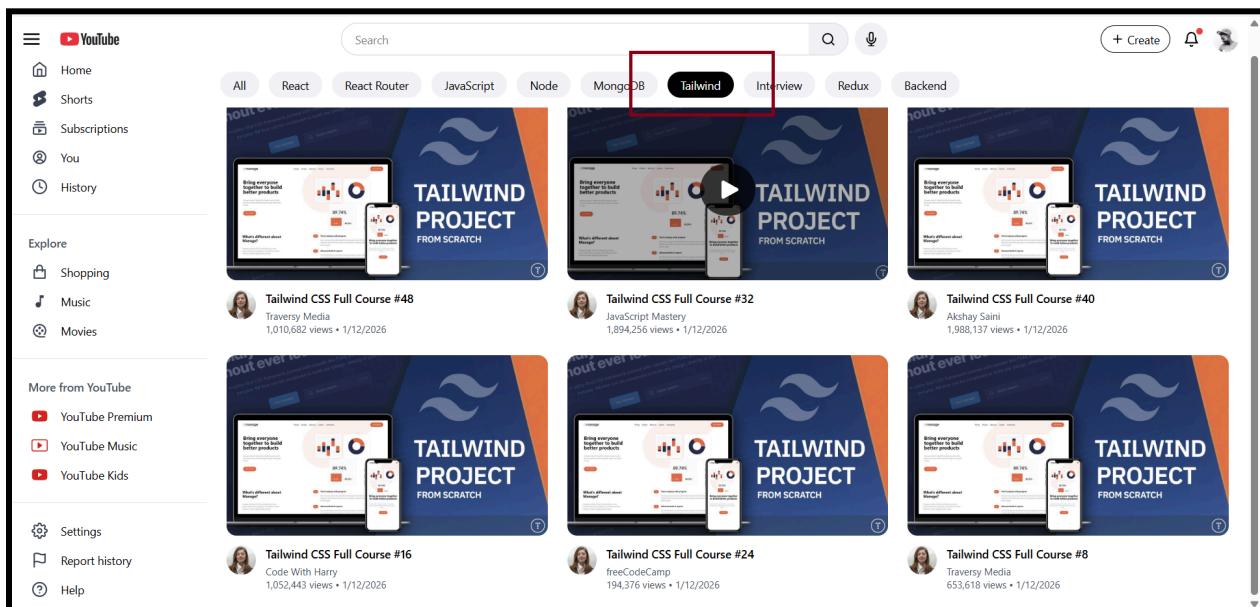


The image shows the same Google sign-in page as above, but with an error message. The "Email" field now contains the text "Invalid email or password" and has a red border, indicating an input error. The "Password" field contains the placeholder "*****". The rest of the page, including the "Log In" button, remains the same.

Search functionality:



Filter Functionality:



Video Player/ Title / Description / Channel name / Like / Comments / Side Section



React JS Full Course 2024 #49

freeCodeCamp 1125878 subscribers

1,362,355 views • 1/12/2026

Learn React JS from scratch in this comprehensive course.

0 Share ...

Node.js & Express Crash Course #43
Traversy Media 670,830 views

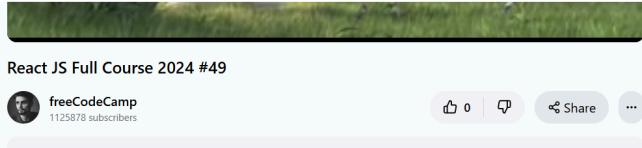
Tailwind CSS Full Course #16
Code With Harry 1,052,443 views

React JS Full Course 2024 #33
Traversy Media 1,359,241 views

JavaScript Crash Course #13
Traversy Media 22,341 views

hello nd 69 views

MongoDB Complete Tutorial #23



React JS Full Course 2024 #49

freeCodeCamp 1125878 subscribers

1,362,355 views • 1/12/2026

Learn React JS from scratch in this comprehensive course.

0 Share ...

3 Comments

R Add a comment...

may sen 1/17/2026 Great lecture
0 Reply

freeCodeCamp 1/14/2026 Let us know your thoughts in the comments ❤️
0 Reply

freeCodeCamp 1/14/2026 Welcome to this video 🎉
0 Reply

JavaScript Crash Course #13
Traversy Media 22,341 views

hello nd 69 views

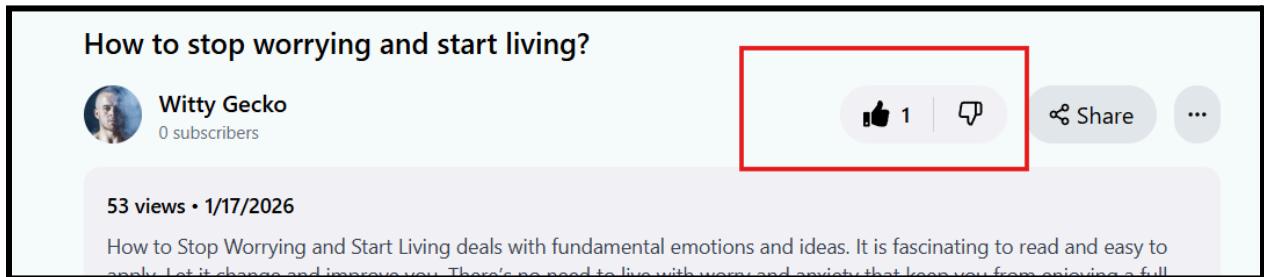
MongoDB Complete Tutorial #23
Traversy Media 997,120 views

Build YouTube Clone Using React #42
JavaScript Mastery 707,338 views

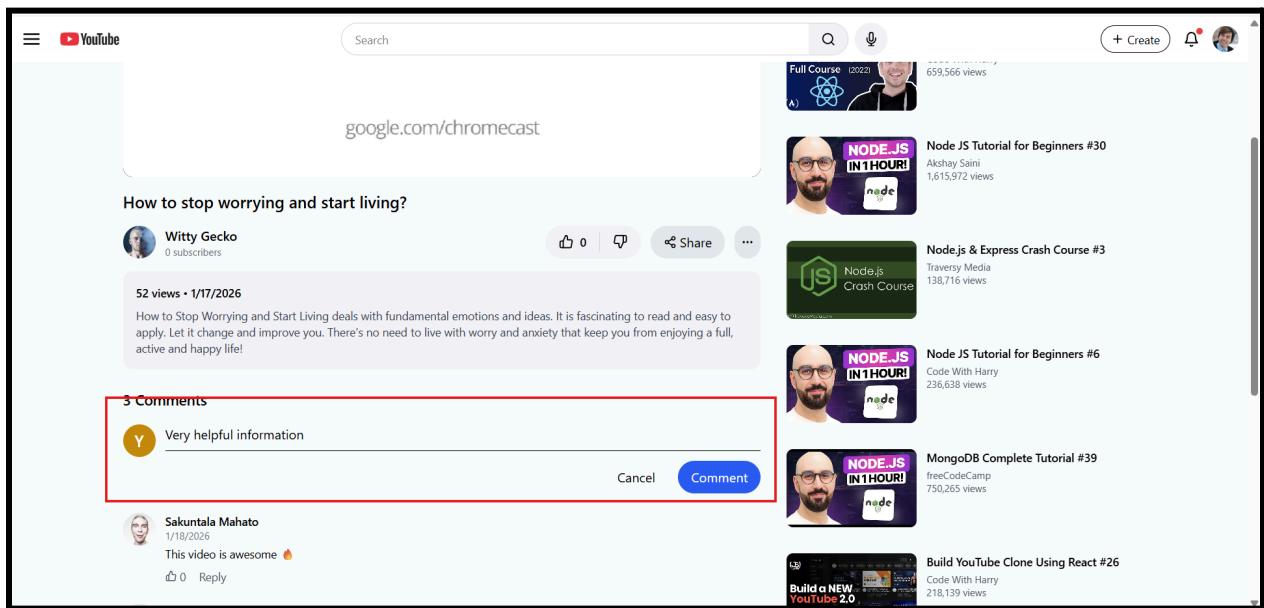
Node JS Tutorial for Beginners #38
Traversy Media 636,529 views

bye

Add/Remove Like:



Add comment:



Edit/Delete comment:

4 Comments

 Add a comment...

 **you me**
1/18/2026
Very helpful information
 0 Reply

 **Sakuntala Mahato**


 Edit
 Delete

 Add a comment...

 **you me**
1/18/2026
 Very helpful information! Thanks
 Save 

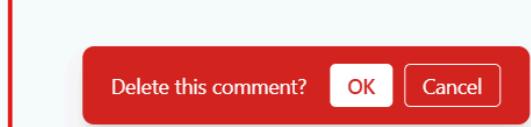
 0 Reply

 Add a comment...

 **you me**
1/18/2026
 Very helpful information! Thanks
 0 Reply

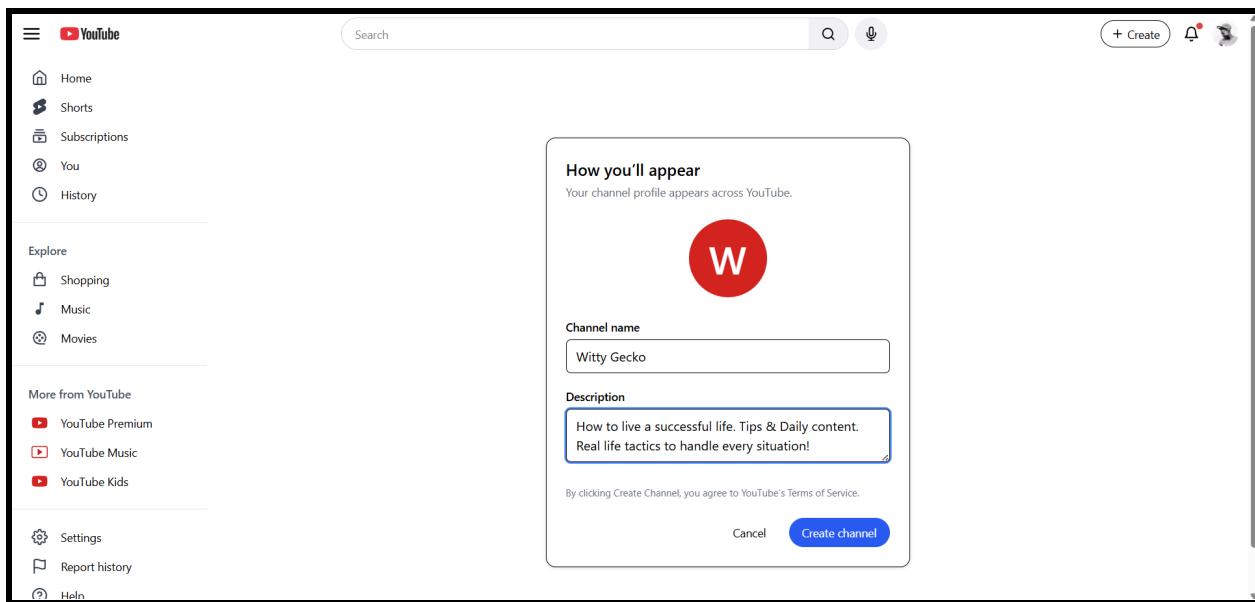
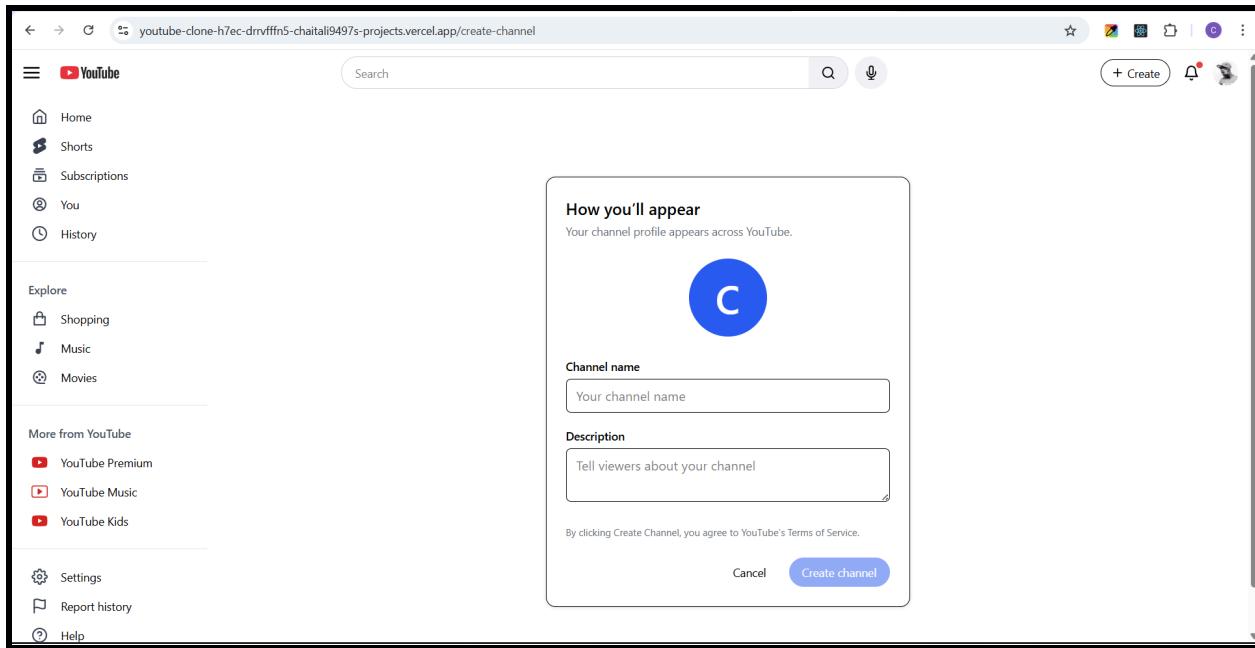
 **Sakuntala Mahato**
1/18/2026
This video is awesome 🔥
 0 Reply

 **Raja Ram**
1/17/2026

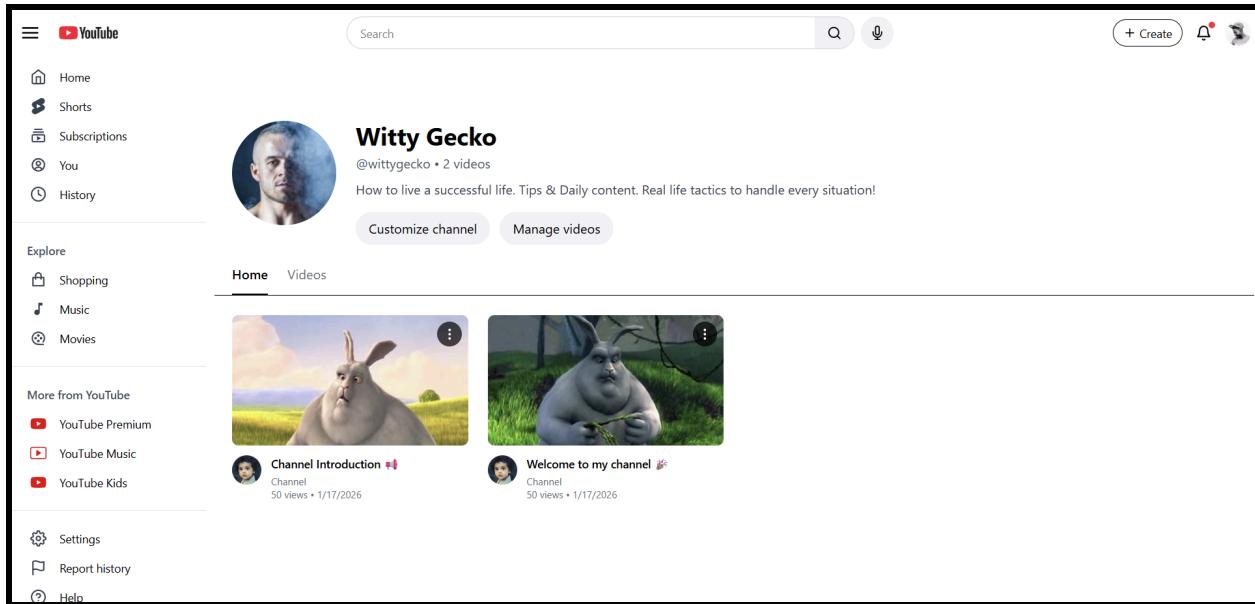
 Delete this comment?  OK 

Channel Page:

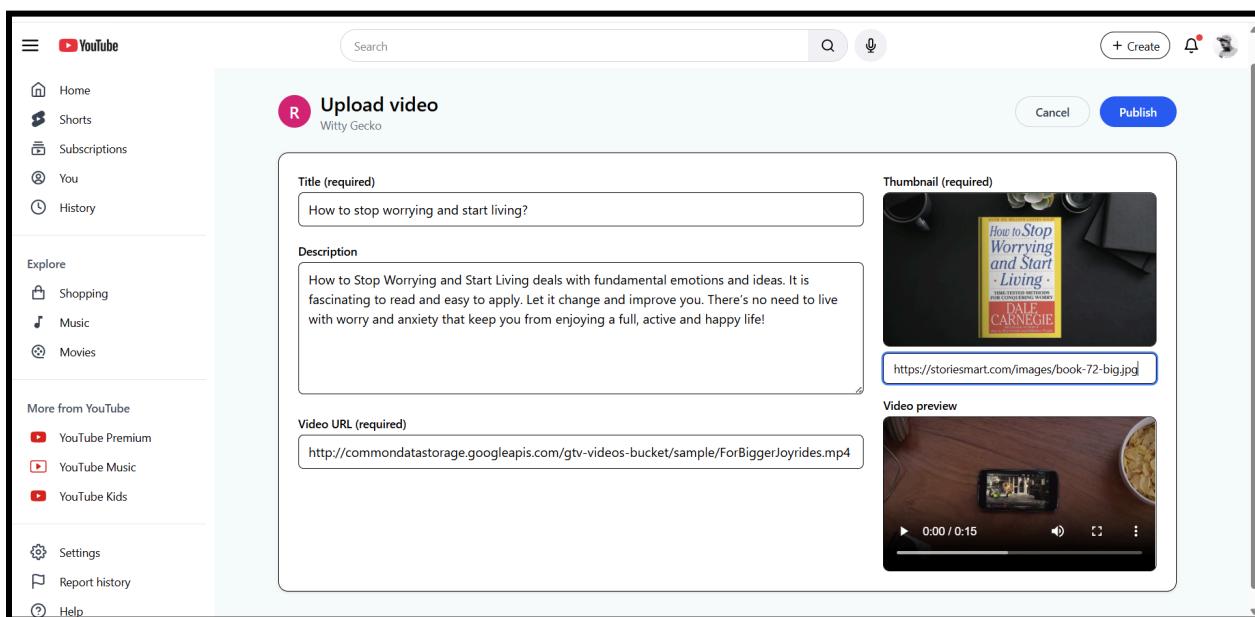
Create Channel by giving Channel Name & Description



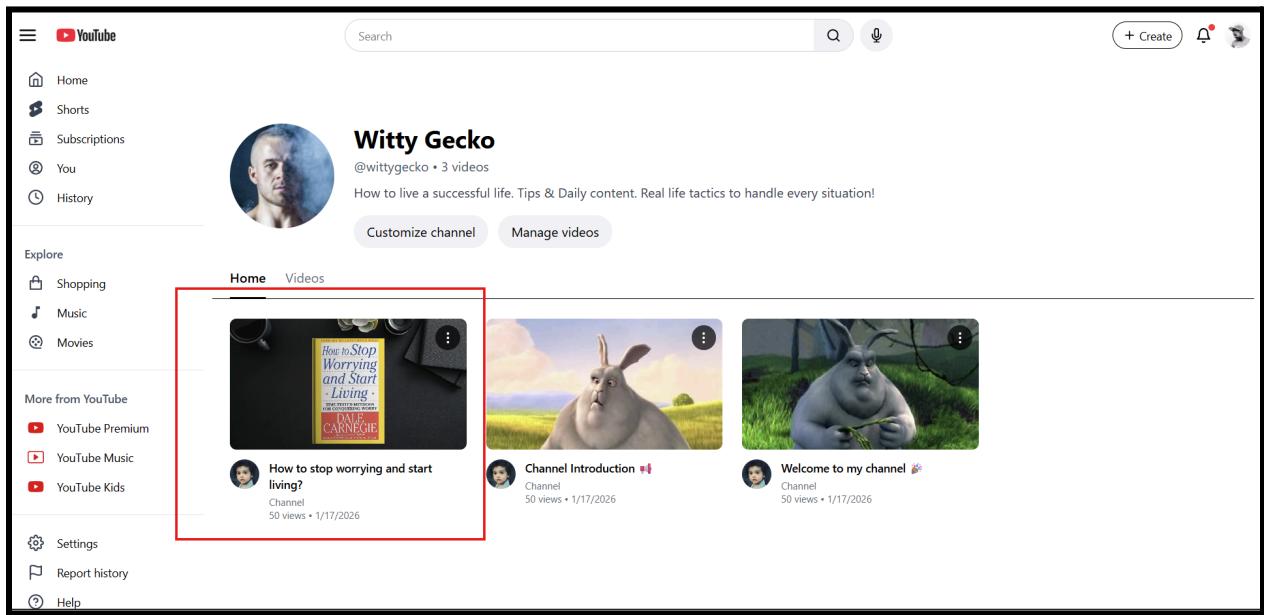
Created Channel



Upload video and image (Provide hosted image url & video url)



We can see the uploaded image in the Channel page



Test Results :

1) POST register users

Request:

Method: POST

URL: <http://localhost:5000/api/auth/register>

Body:

```
{  
    "name": "Sakuntala Mahato",  
    "email": "sakuntala@gmail.com",  
    "password": "sak-UN-@2025"  
}
```

Headers:

Content-Type: application/json

Response:

```
{  
  "status": "success",  
  "token":  
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY5NmM4YmE2  
    MWIwNjk2YzdmNjU4MDJiMCIsImIhdCI6MTc2ODcyMTMxOCwiZXhwI  
    joxNzY5MzI2MTE4fQ.cMG2mW2CAVLcOij5yQQKkvYmTq4pi6idNI48  
    CXV3FZw",  
  "user": {  
    "id": "696c8ba61b0696c7f65802b0",  
    "name": "Sakuntala Mahato",  
    "email": "sakuntala@gmail.com",  
    "channel": null  
  }  
}
```

The screenshot shows a POST request to `http://localhost:5000/api/auth/register`. The response status is `201 Created`, size is `314 Bytes`, and time is `321 ms`. The response body is a JSON object containing the token and user details.

```
POST ▾ http://localhost:5000/api/auth/register Send  
Query Headers 3 Auth Body 1 Tests Pre Run  
JSON XML Text Form Form-encode GraphQL Binary  
JSON Content Format  
1 {  
2   "name": "Sakuntala Mahato",  
3   "email": "sakuntala@gmail.com",  
4   "password": "sak-UN-@2025"  
5 }  
6  
Status: 201 Created Size: 314 Bytes Time: 321 ms  
Response Headers 8 Cookies Results Docs { }  
1 {  
2   "status": "success",  
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
        .eyJpZCI6IjY5NmM4YmE2MWIwNjk2YzdmNjU4MDJiMCIsImIhdCI6MTc2ODcyM  
        TMxOCwiZXhwIjoxNzY5MzI2MTE4fQ.  
        .cMG2mW2CAVLcOij5yQQKkvYmTq4pi6idNI48CXV3FZw",  
4   "user": {  
5     "id": "696c8ba61b0696c7f65802b0",  
6     "name": "Sakuntala Mahato",  
7     "email": "sakuntala@gmail.com",  
8     "channel": null  
9   }  
10 }
```

Data is inserted in MongoDB Database

```
_id: ObjectId('696c8ba61b0696c7f65802b0')
name: "Sakuntala Mahato"
email: "sakuntala@gmail.com"
password: "$2b$12$neJdkmK8weGJf9r2XPluk.FUaZhQRgjaWhzXPoG2RbGz4IN1aqdBq"
avatar: "https://i.pravatar.cc/150?img=6"
subscribers: 0
createdAt: 2026-01-18T07:28:38.688+00:00
updatedAt: 2026-01-18T07:28:38.688+00:00
__v: 0
```

Validation if user tries to signup using same email

The screenshot shows a Postman request to `http://localhost:5000/api/auth/register`. The request method is POST. The JSON body contains:

```
{  
  "name": "Sakuntala Mahato",  
  "email": "sakuntala@gmail.com",  
  "password": "sak-UN-@2025"}  
}
```

The response status is 400 Bad Request, with a message: "User already exists".

2) POST login user

Request:

Method: POST

URL: <http://localhost:5000/api/auth/login>

Body:

```
{  
    "email": "sakuntala@gmail.com",  
    "password": "sak-UN-@2025"  
}
```

Headers:

Content-Type: application/json

Response:

```
{  
    "status": "success",  
    "token":  
        "eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJpZCI6IjY5NmM4YmE2  
        MWIwNjk2YzdmNjU4MDJiMCIsImhdCI6MTc2ODcyMjEzMjI2OTM4fQ.  
        rJrKgHIAkv6rKz93kbvPYCHV_xLMGrz2oYf0yR  
        c7rXc",  
    "user": {  
        "id": "696c8ba61b0696c7f65802b0",  
        "name": "Sakuntala Mahato",  
        "email": "sakuntala@gmail.com",  
        "channel": null  
    }  
}
```

The screenshot shows a Postman request for a POST API endpoint at `http://localhost:5000/api/auth/login`. The request body is JSON, containing:

```
1 {
2   "email": "sakuntala@gmail.com",
3   "password": "sak-UN-@2025"
4 }
```

The response status is 200 OK, with a size of 314 bytes and a time of 289 ms. The response body is a JSON object:

```
1 {
2   "status": "success",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
4 .eyJpZCI6IjY5NmM4YmE2MWIwNjk2YzdmNjU4MDJiMCIsImhlhdCI6MTc2ODcyM
5 jEzOCwizXhwIjoxNzY5MzI2OTM4fQ
6 .rJrkghIAkv6rKz93kbvPYCHV_xLMGrz2oYf0yRc7rXc",
7 "user": {
8   "id": "696c8ba61b0696c7f65802b0",
9   "name": "Sakuntala Mahato",
10  "email": "sakuntala@gmail.com",
11  "channel": null
12 }
13 }
```

Validation for wrong email/password

The screenshot shows a Postman request for a POST API endpoint at `http://localhost:5000/api/auth/login`. The request body is JSON, containing:

```
1 {
2   "email": "sakuntala@gmail.com",
3   "password": "sak-UN-@202"
4 }
```

The response status is 401 Unauthorized, with a size of 55 bytes and a time of 274 ms. The response body is a JSON object:

```
1 {
2   "status": "fail",
3   "message": "invalid email or password"
4 }
```

3) CREATE channel

Request:

Method: POST

URL: <http://localhost:5000/api/channels>

Headers:

Content-Type: application/json

Authorization: Bearer

eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJpZCI6IjY5NmM4YmE2M
WIwNjk2YzdmNjU4MDJiMCIsImIhdCI6MTc2ODcyMjEzMjOCwiZXhwIjox
NzY5MzI2OTM4fQ.rJrKgHIAkv6rKz93kbvPYCHV_xLMGrz2oYf0yRc7r
Xc

Body :

{

```
"name": "Chaiti Vlogs",
"description": "Daily lifestyle and tech videos",
"avatar": "https://i.pravatar.cc/150?img=3",
"banner": "https://picsum.photos/1200/300"
```

}

Response:

{

```
"status": "success",
"channel": {
  "name": "Chaiti Vlogs",
  "description": "Daily lifestyle and tech videos",
  "banner": "https://picsum.photos/1200/300",
  "avatar": "https://i.pravatar.cc/150?img=3",
  "owner": "696c8ba61b0696c7f65802b0",
  "subscribers": 0,
```

```
        "_id": "696c91351b0696c7f65802bc",
        "createdAt": "2026-01-18T07:52:21.668Z",
        "updatedAt": "2026-01-18T07:52:21.668Z",
        "__v": 0
    },
    "videos": [
        {
            "title": "Welcome to my channel 🎉",
            "description": "This is my first video on this channel.",
            "videoUrl":
                "https://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4",
            "thumbnail":
                "https://healthin2nds.com/wp-content/uploads/2026/01/big_bunny_2.jpg",
            "channel": "696c91351b0696c7f65802bc",
            "views": 50,
            "likes": [],
            "dislikes": [],
            "commentsCount": 0,
            "_id": "696c91351b0696c7f65802be",
            "__v": 0,
            "createdAt": "2026-01-18T07:52:21.693Z",
            "updatedAt": "2026-01-18T07:52:21.693Z"
        },
        {
            "title": "Channel Introduction 🎤",
            "description": "Let me introduce what this channel is about.",
            "videoUrl":
                "https://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4",
            "channel": "696c91351b0696c7f65802bc",
            "views": 50,
            "likes": [],
            "dislikes": [],
            "commentsCount": 0,
            "_id": "696c91351b0696c7f65802cf",
            "__v": 0,
            "createdAt": "2026-01-18T07:52:21.718Z",
            "updatedAt": "2026-01-18T07:52:21.718Z"
        }
    ]
}
```

```



```

POST <http://localhost:5000/api/channels> Send

Query	Headers 4	Auth	<u>Body 1</u>	Tests	Pre Run	
JSON	XML	Text	Form	Form-encode	GraphQL	Binary
JSON Content				Format		
<pre> 1 { 2 "name": "Chaiti Vlogs", 3 "description": "Daily lifestyle and tech videos", 4 "avatar": "https://i.pravatar.cc/150?img=3", 5 "banner": "https://picsum.photos/1200/300" 6 } 7 </pre>						

Status: 201 Created Size: 1.3 KB Time: 76 ms

Response	Headers 8	Cookies	Results	Docs
<pre> 1 { 2 "status": "success", 3 "channel": { 4 "name": "Chaiti Vlogs", 5 "description": "Daily lifestyle and tech videos", 6 "banner": "https://picsum.photos/1200/300", 7 "avatar": "https://i.pravatar.cc/150?img=3", 8 "owner": "696c8ba61b0696c7f65802b0", 9 "subscribers": 0, 10 "_id": "696c91351b0696c7f65802bc", 11 "createdAt": "2026-01-18T07:52:21.668Z", 12 "updatedAt": "2026-01-18T07:52:21.668Z", 13 "__v": 0 14 }, 15 "videos": [16 { 17 "title": "Welcome to my channel 🎉", 18 "description": "This is my first video on this channel.", 19 "videoUrl": "https://commondatastorage.googleapis.com/gtv- 19 -videos-bucket/sample/BigBuckBunny.mp4", 20 "thumbnail": "https://healthin2nds.com/wp-content/uploads/ 20 /2026/01/big_bunny_2.jpg", 21 "channel": "696c91351b0696c7f65802bc", 22 "views": 50, 23 "likes": [], 24 "dislikes": [], 25 "commentsCount": 0, 26 "_id": "696c91351b0696c7f65802be", 27 "__v": 0, 28 "createdAt": "2026-01-18T07:52:21.693Z", 29 "updatedAt": "2026-01-18T07:52:21.693Z" </pre>	Copy			

Data is inserted in MongoDB Database

```
_id: ObjectId('696c91351b0696c7f65802bc')
name : "Chaiti Vlogs"
description : "Daily lifestyle and tech videos"
banner : "https://picsum.photos/1200/300"
avatar : "https://i.pravatar.cc/150?img=3"
owner : ObjectId('696c8ba61b0696c7f65802b0')
subscribers : 0
createdAt : 2026-01-18T07:52:21.668+00:00
updatedAt : 2026-01-18T07:52:21.668+00:00
__v : 0
```

Validation if user tries to create the channel again, shows “Channel Already Exists”

The screenshot shows two API requests in Postman.

Request 1: A successful POST request to `http://localhost:5000/api/channels`. The response status is 201 Created, and the response body is:

```
{ "_id": "696c91351b0696c7f65802bc", "name": "Chaiti Vlogs", "description": "Daily lifestyle and tech videos", "banner": "https://picsum.photos/1200/300", "avatar": "https://i.pravatar.cc/150?img=3", "owner": "696c8ba61b0696c7f65802b0", "subscribers": 0, "createdAt": "2026-01-18T07:52:21.668+00:00", "updatedAt": "2026-01-18T07:52:21.668+00:00", "__v": 0 }
```

Request 2: An attempt to create the same channel again, resulting in a 400 Bad Request error. The response status is 400 Bad Request, and the response body is:

```
{ "message": "Channel already exists" }
```

4) GET CHANNEL BY ID (+ VIDEOS)

Request:

Method: GET

URL: <http://localhost:5000/api/channels/696c91351b0696c7f65802bc>

Body:

Headers:

Content-Type: application/json

Authorization: Bearer

Response:

```
{  
  "channel": {  
    "_id": "696c91351b0696c7f65802bc",  
    "name": "Chaiti Vlogs",  
    "description": "Daily lifestyle and tech videos",  
    "banner": "https://picsum.photos/1200/300",  
    "avatar": "https://i.pravatar.cc/150?img=3",  
    "owner": {  
      "_id": "696c8ba61b0696c7f65802b0"  
    },  
    "subscribers": 0,  
    "createdAt": "2026-01-18T07:52:21.668Z",  
    "updatedAt": "2026-01-18T07:52:21.668Z",  
    "v": 0  
  }  
}
```

```
},
"videos": [
{
  "_id": "696c91351b0696c7f65802be",
  "title": "Welcome to my channel 🎉",
  "description": "This is my first video on this channel.",
  "videoUrl":
  "https://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4",
  "thumbnail":
  "https://healthin2nds.com/wp-content/uploads/2026/01/big_bunny_2.jpg",
  "channel": "696c91351b0696c7f65802bc",
  "views": 50,
  "likes": [],
  "dislikes": [],
  "commentsCount": 0,
  "__v": 0,
  "createdAt": "2026-01-18T07:52:21.693Z",
  "updatedAt": "2026-01-18T07:52:21.693Z"
},
{
  "_id": "696c91351b0696c7f65802bf",
  "title": "Channel Introduction 🎤",
  "description": "Let me introduce what this channel is about.",
  "videoUrl":
  "https://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4",
  "thumbnail":
  "https://healthin2nds.com/wp-content/uploads/2026/01/big_bunny.jpg",
  "channel": "696c91351b0696c7f65802bc",
  "views": 50,
```

```

    "likes": [],
    "dislikes": [],
    "commentsCount": 0,
    "__v": 0,
    "createdAt": "2026-01-18T07:52:21.693Z",
    "updatedAt": "2026-01-18T07:52:21.693Z"
}
]
}

```

The screenshot shows a Postman request and its corresponding response.

Request (Left):

- Method: GET
- URL: `http://localhost:5000/api/channels/696c91351b0696c7f65802bc`
- Headers: 4
- Auth: None
- Body: Body (selected)
- Tests: None
- Pre Run: None

Response (Right):

- Status: 200 OK
- Size: 1.29 KB
- Time: 35 ms

Response	Headers 8	Cookies	Results	Docs
<pre> 1 { 2 "channel": { 3 "_id": "696c91351b0696c7f65802bc", 4 "name": "Chaiti Vlogs", 5 "description": "Daily lifestyle and tech videos", 6 "banner": "https://picsum.photos/1200/300", 7 "avatar": "https://i.pravatar.cc/150?img=3", 8 "owner": { 9 "_id": "696c8ba61b0696c7f65802b0" 10 }, 11 "subscribers": 0, 12 "createdAt": "2026-01-18T07:52:21.668Z", 13 "updatedAt": "2026-01-18T07:52:21.668Z", 14 "__v": 0 15 }, 16 "videos": [17 { 18 "_id": "696c91351b0696c7f65802be", 19 "title": "Welcome to my channel 🎥", 20 "description": "This is my first video on this channel.", 21 "videoUrl": "https://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4", 22 "thumbnail": "https://healthin2nds.com/wp-content/uploads/2026/01/big_bunny_2.jpg", 23 "channel": "696c91351b0696c7f65802bc", 24 "views": 50, 25 "likes": [], 26 "dislikes": [], 27 "commentsCount": 0, 28 "__v": 0, 29 "createdAt": "2026-01-18T07:52:21.693Z", </pre>				

5) GET MY CHANNEL (Logged-in User)

Request:

Method: GET

URL: <http://localhost:5000/api/channels/me>

Body:

Headers:

Content-Type: application/json

Authorization: Bearer

eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJpZCI6IjY5NmM4YmE2M
WIwNjk2YzdmNjU4MDJiMCIsImIhdCI6MTc2ODcyMjEzMjEzOCwiZXhwIjox
NzY5MzI2OTM4fQ.rJrKgHIAkv6rKz93kbvPYCHV_xLMGrz2oYf0yRc7r
Xc

Response:

```
{  
  "_id": "696c91351b0696c7f65802bc",  
  "name": "Chaiti Vlogs",  
  "description": "Daily lifestyle and tech videos",  
  "banner": "https://picsum.photos/1200/300",  
  "avatar": "https://i.pravatar.cc/150?img=3",  
  "owner": "696c8ba61b0696c7f65802b0",  
  "subscribers": 0,  
  "createdAt": "2026-01-18T07:52:21.668Z",  
  "updatedAt": "2026-01-18T07:52:21.668Z",  
  "__v": 0  
}
```

GET Send

Query Headers **4** Auth Body Tests Pre Run

HTTP Headers Raw

<input checked="" type="checkbox"/> Accept	/*
<input checked="" type="checkbox"/> User-Agent	Thunder Client (https://www.thunderclient.com)
<input checked="" type="checkbox"/> Content-Type	application/json
<input checked="" type="checkbox"/> Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC...
<input type="checkbox"/> header	value

Status: 200 OK Size: 326 Bytes Time: 64 ms

Response Headers **8** Cookies Results Docs

```
1 {  
2   "_id": "696c91351b0696c7f65802bc",  
3   "name": "Chaiti Vlogs",  
4   "description": "Daily lifestyle and tech videos",  
5   "banner": "https://picsum.photos/1200/300",  
6   "avatar": "https://i.pravatar.cc/150?img=3",  
7   "owner": "696c8ba61b0696c7f65802b0",  
8   "subscribers": 0,  
9   "createdAt": "2026-01-18T07:52:21.668Z",  
10  "updatedAt": "2026-01-18T07:52:21.668Z",  
11  "__v": 0  
12 }
```

Unauthorised user wont see channel

GET Send

Query Headers **4** Auth Body Tests Pre Run

HTTP Headers Raw

<input checked="" type="checkbox"/> Accept	/*
<input checked="" type="checkbox"/> User-Agent	Thunder Client (https://www.thunderclient.com)
<input checked="" type="checkbox"/> Content-Type	application/json
<input checked="" type="checkbox"/> Authorization	value
<input type="checkbox"/> header	value

Status: 401 Unauthorized Size: 28 Bytes Time: 3 ms

Response Headers **8** Cookies Results Docs

```
1 {  
2   "message": "Not authorized"  
3 }
```

6) GET ALL VIDEOS (Home Feed)

Request:

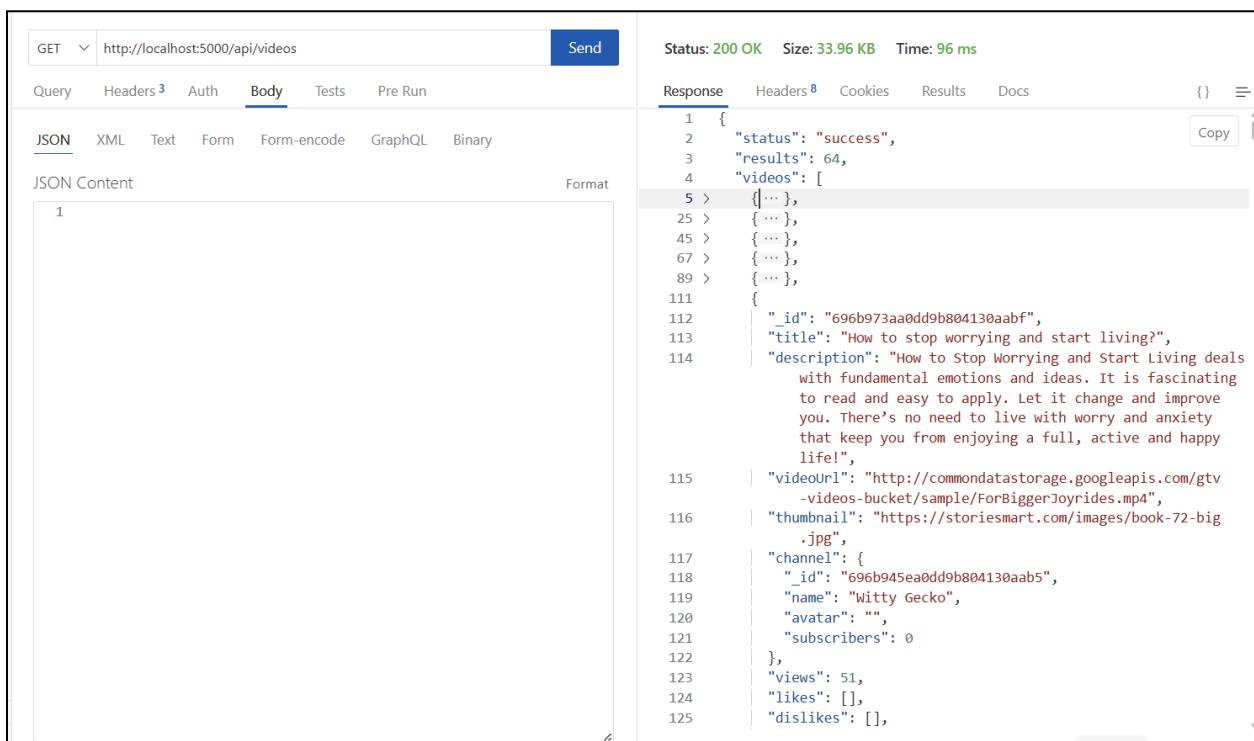
Method: GET

URL: <http://localhost:5000/api/videos>

Body:

Header:

Response: [Attached the screenshot due to large dataset]



GET <http://localhost:5000/api/videos> Send

Status: 200 OK Size: 33.96 KB Time: 96 ms

Response	Headers 8	Cookies	Results	Docs
<pre>1 { 2 "status": "success", 3 "results": 64, 4 "videos": [5 {...}, 6 {...}, 7 {...}, 8 {...}, 9 {...}, 10 { 11 "_id": "696b973aa0dd9b804130aabf", 12 "title": "How to stop worrying and start living?", 13 "description": "How to Stop Worrying and Start Living deals with fundamental emotions and ideas. It is fascinating to read and easy to apply. Let it change and improve you. There's no need to live with worry and anxiety that keep you from enjoying a full, active and happy life!", 14 "videourl": "http://commondatastorage.googleapis.com/gtv-videos-bucket/sample/ForBiggerJoyrides.mp4", 15 "thumbnail": "https://storiesmart.com/images/book-72-big.jpg", 16 "channel": { 17 "_id": "696b945ea0dd9b804130aab5", 18 "name": "Witty Gecko", 19 "avatar": "", 20 "subscribers": 0 21 }, 22 "views": 51, 23 "likes": [], 24 "dislikes": [] 25 } 26] 27 }</pre>				

Query Headers 3 Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1
```

7) GET VIDEO BY ID (Watch Page)

Request:

Method: GET

URL: <http://localhost:5000/api/videos/696b973aa0dd9b804130aabf>

Body:

Headers:

Response:

```
{  
  "status": "success",  
  "video": {  
    "_id": "696b973aa0dd9b804130aabf",  
    "title": "How to stop worrying and start living?",  
    "description": "How to Stop Worrying and Start Living deals with fundamental emotions and ideas. It is fascinating to read and easy to apply. Let it change and improve you. There's no need to live with worry and anxiety that keep you from enjoying a full, active and happy life!",  
    "videoUrl":  
      "http://commondatastorage.googleapis.com/gtv-videos-bucket/sample/ForBiggerJoyrides.mp4",  
    "thumbnail": "https://storiesmart.com/images/book-72-big.jpg",  
    "channel": {  
      "_id": "696b945ea0dd9b804130aab5",  
      "name": "Witty Gecko",  
      "avatar": "",  
      "subscribers": 0  
    },  
    "views": 51,  
  }  
}
```

```

    "likes": [],
    "dislikes": [],
    "commentsCount": 2,
    "createdAt": "2026-01-17T14:05:46.223Z",
    "updatedAt": "2026-01-17T16:29:47.884Z",
    "__v": 0
},
"likes": 0,
"dislikes": 0,
"likedByUser": false,
"dislikedByUser": false
}

```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:5000/api/videos/696b973aa0dd9b804130aabf
- Status:** 200 OK
- Size:** 854 Bytes
- Time:** 55 ms
- Response Headers:** Headers 8
- Response Body:**

```

2   "status": "success",
3     "video": {
4       "_id": "696b973aa0dd9b804130aabf",
5       "title": "How to stop worrying and start living?",
6       "description": "How to Stop Worrying and Start Living deals with fundamental emotions and ideas. It is fascinating to read and easy to apply. Let it change and improve you. There's no need to live with worry and anxiety that keep you from enjoying a full, active and happy life!",
7       "videoUrl": "http://commondatastorage.googleapis.com/gtv-videos-bucket/sample/ForBiggerJoyrides.mp4",
8       "thumbnail": "https://storiesmart.com/images/book-72-big.jpg",
9       "channel": {
10         "_id": "696b945ea0dd9b804130aab5",
11         "name": "Witty Gecko",
12         "avatar": "",
13         "subscribers": 0
14       },
15       "views": 51,
16       "likes": [],
17       "dislikes": [],
18       "commentsCount": 2,
19       "createdAt": "2026-01-17T14:05:46.223Z",
20       "updatedAt": "2026-01-17T16:29:47.884Z",
21       "__v": 0
22     },
23     "likes": 0,
24     "dislikes": 0,
25     "likedByUser": false,
26     "dislikedByUser": false
27   }

```

8) LIKE VIDEO

Request:

Method: POST

URL: <http://localhost:5000/api/videos/:id/like>

<http://localhost:5000/api/videos/696b973aa0dd9b804130aabf/like>

Body:

Headers:

Content-Type: application/json

Authorization: Bearer

eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJpZCI6IjY5NmM4YmE2M

WlwNjk2YzdmNjU4MDJiMCIsImhdCl6MTc2ODcyMjEzOCwiZXhwIjox

NzY5Mzl2OTM4fQ.rJrKgHIAkv6rKz93kbvPYCHV_xLMGrz2oYf0yRc7r

Xc

Response:

```
{
  "status": "success",
  "likes": 1,
  "dislikes": 0,
  "likedByUser": true,
  "dislikedByUser": false
}
```

The screenshot shows the Thunder Client interface with the following details:

- Method:** POST
- URL:** http://localhost:5000/api/videos/696b973aa0dd9b804130aabf/like
- Status:** 200 OK
- Size:** 85 Bytes
- Time:** 64 ms
- Headers:** 4 (Accept, User-Agent, Content-Type, Authorization)
- HTTP Headers:** Raw
- Response:**

```
1  {
2    "status": "success",
3    "likes": 1,
4    "dislikes": 0,
5    "likedByUser": true,
6    "dislikedByUser": false
7 }
```

9) DISLIKE VIDEO

Request:

Method: POST

URL: <http://localhost:5000/api/videos/:id/dislike>

<http://localhost:5000/api/videos/696b973aa0dd9b804130aabf/dislike>

Body:

Headers:

Content-Type: application/json

Authorization: Bearer

eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJpZCI6IjY5NmM4YmE2M
WIwNjk2YzdmNjU4MDJiMCIsImIhdCI6MTc2ODcyMjEzMjOCwiZXhwIjox
NzY5MzI2OTM4fQ.rJrKgHIAkv6rKz93kbvPYCHV_xLMGrz2oYf0yRc7r
Xc

Response:

```
{  
  "status": "success",  
  "likes": 0,  
  "dislikes": 1,  
  "likedByUser": false,  
  "dislikedByUser": true  
}
```

POST http://localhost:5000/api/videos/696b973aa0dd9b804130aabf/dislike Send

HTTP Headers

Header	Value
Accept	*/*
User-Agent	Thunder Client (https://www.thunderclient.io)
Content-Type	application/json
Authorization	Bearer eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVC...
header	value

Status: 200 OK Size: 85 Bytes Time: 48 ms

Response Headers 8 Cookies Results Docs

```
1 {
2   "status": "success",
3   "likes": 0,
4   "dislikes": 1,
5   "likedByUser": false,
6   "dislikedByUser": true
7 }
```

10) COMMENT VIDEO

Request:

Method: POST

URL: <http://localhost:5000/api/comments/:id>

http://localhost:5000/api/comments/696b973aa0dd9b804130aabf

Body:

```
{  
  "text": "This video is awesome 🔥"  
}
```

Headers:

Content-Type: application/json

Authorization: Bearer

eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJpZCI6IjY5NmM4YmE2M
WIwNjk2YzdmNjU4MDJiMCIsImhdCI6MTc2ODcyMjEzOCwiZXhwIjox
NzY5MzI2OTM4fQ.rJrKgHIAkv6rKz93kbvPYCHV_xLMGrz2oYf0yRc7r
Xc

Response:

```
{  
  "text": "This video is awesome 🔥",  
  "video": "696b973aa0dd9b804130aabf",  
  "user": {  
    "_id": "696c8ba61b0696c7f65802b0",  
    "name": "Sakuntala Mahato",  
    "avatar": "https://i.pravatar.cc/150?img=6"  
  },  
  "likes": [],  
  "parentComment": null,  
  "_id": "696ca80f1b0696c7f65802df",  
  "createdAt": "2026-01-18T09:29:51.595Z",  
  "updatedAt": "2026-01-18T09:29:51.595Z",  
  "__v": 0  
}
```

----- Thank you -----