RFLIX - ZENSE RECRUITMENT PROJECT 2022

EXPERIENCE THE SITE LIVE - RFLIX

HOW TO RUN

This project is divided into two parts –

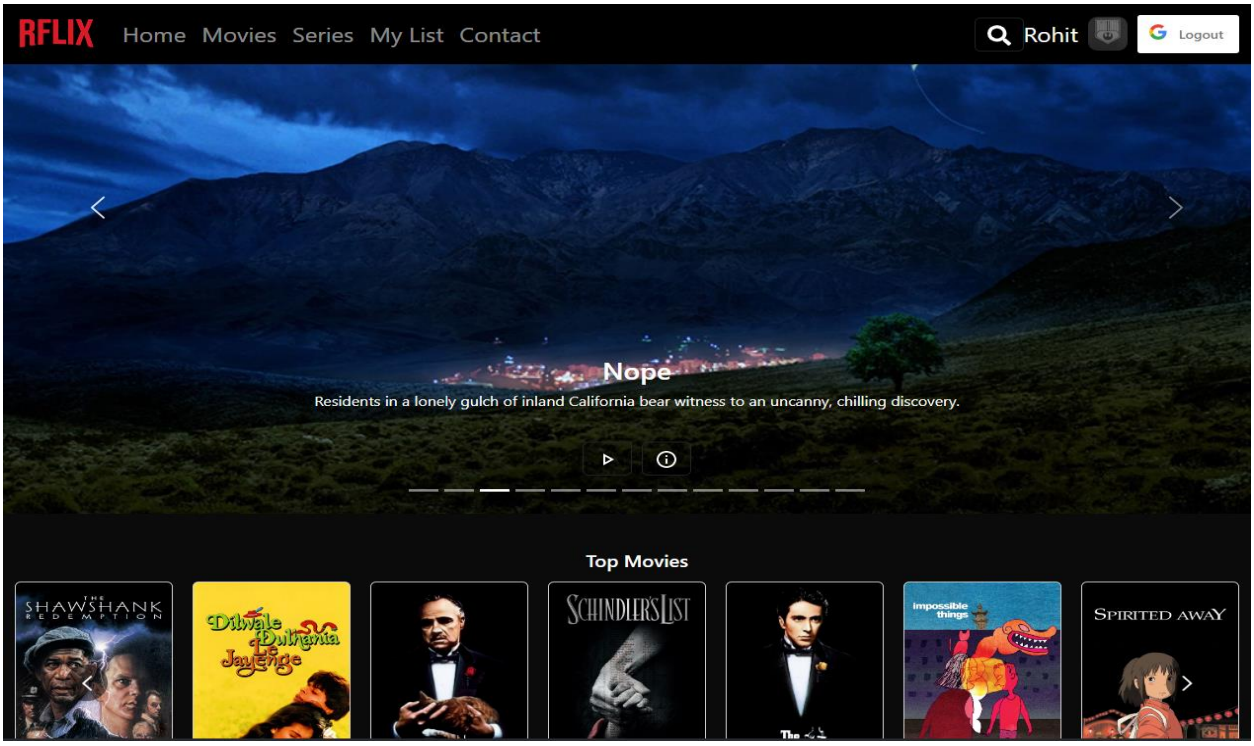
Backend = https://github.com/RohitShah1706/Rflix-backend-1706

Frontend = https://github.com/RohitShah1706/Rflix-frontend-1706

You can also find these hosted on respective links

https://rflix-backend-1706.herokuapp.com/ & https://rflix-frontend-1706.netlify.app/

NOTE : The backend and the frontend both require many API KEYS and sensitive environment variables to run. Please contact me at rohitlalit.shah@iiitb.ac.in and I'll send you the .env file that you need to paste in the root folder.



BACKEND – REST API BUILT USING NODE JS, EXPRESS, MONGO DB, FIREBASE

Clone the repository : https://github.com/RohitShah1706/Rflix-backend-1706

Run Locally

Clone the project

```
git clone https://github.com/RohitShah1706/Rflix-backend-1706.git
```

Go to the project directory

```
cd Rflix-backend-1706
```

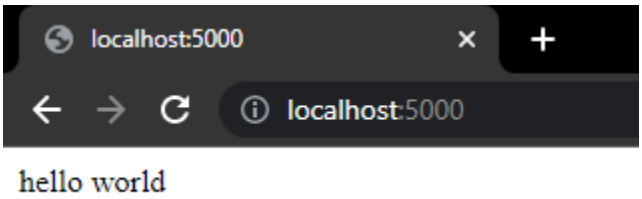
Install dependencies

```
npm install
```

Start the server

```
node server.js
```

This will start a local server on port 5000 – you should be able to confirm it by visiting localhost:5000 in your browser. If you see the below result then you are good to go.



Below is a walkthrough of the different endpoints you can hit to get the required data on movies and series, add a user to database, get a user's top movies & series from my list in mongo db database and much more.

API Reference

BASE_URL = localhost:5000

Get top movies

```
GET BASE_URL/api/top250movies/start/end
```

Parameter	Type	Description
`start`	`string`	**Optional**. Get top movies starting from this no.
`end`	`string`	**Optional**. Get top movies till this no.

Get top series

```
GET BASE_URL/api/top250series/start/end
```

Parameter	Type	Description
`start`	`string`	**Optional**. Get top series starting from this no.
`end`	`string`	**Optional**. Get top series till this no.

Get top movies in theatres

```
GET BASE_URL/api/inTheatres/start/end
```

Parameter	Type	Description
`start`	`string`	**Optional**. Get top series starting from this no.
`end`	`string`	**Optional**. Get top series till this no.

Get embed links for youtube trailer of specific movies

```
GET BASE_URL/api/trailer/movie/tmdb_id
```

Parameter	Type	Description
`tmdb_id`	`string`	**Required**. tmdb_id is specific to TMDB API.

Get embed links for youtube trailer of specific series

```
GET BASE_URL/api/trailer/series/tmdb_id
```

Parameter	Type	Description
`tmdb_id`	`string`	**Required**. tmdb_id is specific to TMDB API.

Get top movies in given genre

```
GET BASE_URL/api/genres/movies/genre_id/start/end
```

Parameter	Type	Description
`genre_id`	`string`	**Required**. genre_id is specific to TMDB API.
`start`	`string`	**Optional**. Get top movies of given genre_id starting from this no.
`end`	`string`	**Optional**. Get top movies of given genre_id till this no.

Get top series in given genre

```
GET BASE_URL/api/genres/series/genre_id/start/end
```

Parameter	Type	Description
`genre_id`	`string`	**Required**. genre_id is specific to TMDB API.
`start`	`string`	**Optional**. Get top series of given genre_id starting from this no.
`end`	`string`	**Optional**. Get top series of given genre_id till this no.

Get a new room for video chat

```
GET BASE_URL/api/meetingroom
```

Sign in a new user with google firebase

```
POST BASE_URL/api/users/signin
```

Data	Type	Description
`Google firebase user object`	`json`	**Required**

Get a user's list of stored movies and series

```
POST BASE_URL/api/users
```

Data	Type	Description
`Google firebase user object`	`json`	**Required**

Add a movie or series to user's my list

```
POST BASE_URL/api/users/addtomylist
```

Data	Type	Description
`Google firebase user object`	`json`	**Required**

FRONTEND – BUILT USING REACT, FIREBASE FOR AUTHENTICATION

Clone the repository : https://github.com/RohitShah1706/Rflix-frontend-1706

Clone the project

```
git clone https://github.com/RohitShah1706/Rflix-frontend-1706.git
```

Go to the project directory

```
cd Rflix-frontend-1706
```

Install dependencies

```
npm install
```

Start the server

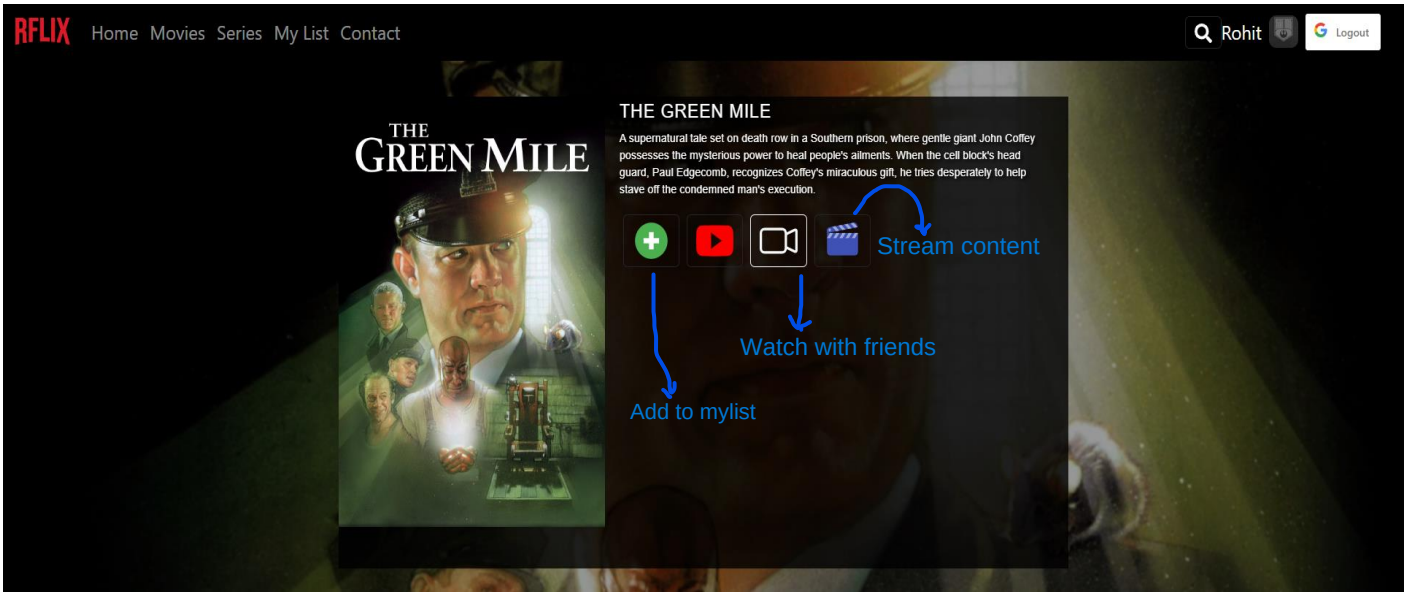
```
npm start
```

This will start a local server on port 3000 – you should be able to confirm it by visiting localhost:3000 in your browser.

MY IDEA FOR RFLIX

In current times you have many demanding OTT platforms but not a single one of them has all the movies and series at a single place. RFLIX brings all of them at a single place where you can stream them. Plus it also has an option to open up a new room in one click where you can stream the movies or series with your friends in realtime, chat and have fun. This meeting option saves you from the hassle of installing apps like zoom, google meet, and then creating a meet.

NOTE – You need to be logged in to create a room or add movies & series to my list.



PIRACY AND MY OPINION – For now the platform fetches the movie & series content to stream from (*hidden) for free. I don't in any way support piracy. This, is not my idea of how the website should provide the content to user. This is temporary and will be removed very soon. See FUTURE IDEAS at the end of the report for more details.

Also fetching content from such illicit ways has its cons and can be discussed in the interview.

IMPLEMENTATION, JOURNEY & TECHNOLOGIES USED

At the start of this project I had little knowledge of how websites work that mostly came from Network classes. But, I was genuinely interested in making a website that I could display and call my own. I started by refreshing what little knowledge I had of the basic HTML, CSS, JS etc.

FRONTEND – REACT, SASS, REACT BOOSTRAP, FIREBASE

I knew how to make static pages, so I started with React – I was fascinated by the concept of delivering a single page website and how to efficiently reuse the same component. This way React allowed my site to be dynamic. There were also great libraries available with React.

Some of my favorites are :

1. **React bootstrap** – provides inbuilt components – The navbar at the top is made using bootstrap.
2. **React device detect** – provides inbuilt classes (isMobile, isDesktop, isTablet etc.) that help in conditional rendering based on devices the user is in.
3. **React toastify** – provides beautiful notifications.
4. **Spline** – allows to import 3d model created using spline design. Checkout contact page background. Spline uses WebGL to render these models. I think of spline as the 3d - figma of the metaverse world.

By this time I had completed building the basic UI and home page of the site, learnt how to make API calls using axios on the client side. But the whole page was populated using the JSON file I had. This was static and I wanted the site to update in real time as new movies come out, popularity changes once I deploy it.

So I felt the need to create a dedicated backend server that does all the heavy lifting of fetching the data, aggregating it. This way the frontend would just need to make API calls to the server I create.

Other technologies used –

Axios – to make API calls to backend server

Firebase – google authentication on client side

Email JS – to create serverless email service on the frontend – checkout contact page.

Daily API – to create instant rooms to start meeting.

React Icons – for icons package.

BACKEND – NODE JS, EXPRESS, MONGODB, FIREBASE

So now, I learnt to create an express server using Node JS. This was where my interested peaked. I learnt about how to optimize API calls, async & await functions, what are promises. Most importantly I learnt why you need to wait for the promise to resolve before sending the response to the frontend. (P.S. – my frontend server crashed many times due to it reading undefined object.)

Then I thought of allowing users to store their favorite movies & series to some database. So I learnt MONGO DB (A NOSQL database – which is beginner friendly). Also I used the online mongo db atlas for hosting the database. I know this is not the most efficient way to do it – I got increased latency because the lifecycle of a get request to fetch content in a user's my list was this:

FRONTEND -> BACKEND -> (check if user is valid) -> MONGO DB ATLAS -> BACKEND -> FRONTEND.

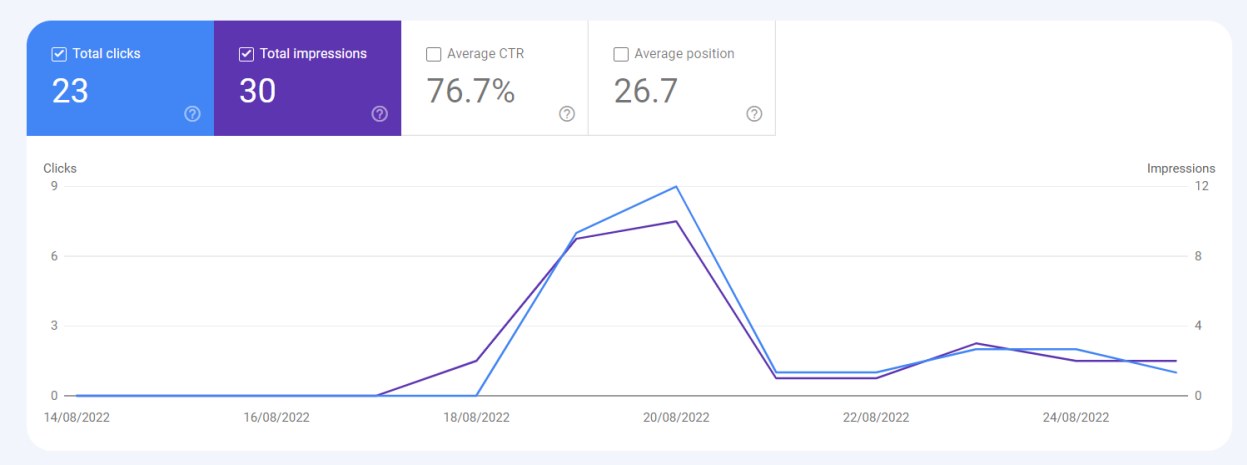
Conclusion: Overall I used the complete MERN stack and at last integrated the frontend with backend.

SEO – FOR INDEXNG MY SITE TO GOOGLE

I had deployed the frontend to netlify and backend on heroku. But nowhere could I find my site on google. Then, I started learning how google's SEO works and how to actually index my site.

First I added relevant headers, keywords to the index.html in react on which the whole content is rendered. This keywords are used by google to match search queries.

I used google search console to register my site and waited for more than 3 days. Then I received my first impressions (that my site had been indexed on google and visible to public – but no one had clicked it).



So I learnt how to build a sitemap and submitted that so that google can better understand my site and crawl throught it. (Sitemap = basically a guide to google that kind of provides direction on what to click, in what order and much more.)

From then on the site started receiving clicks and then I also forwarded the site to some friends.

If you search RFLIX on google you would find it on the second or third page with an average position of 27.

MY EXPERIENCE

I enjoyed a lot building this site. I am proud that the whole time I had not followed a single tutorial (apart from some styling tutorial, ☹) and built this from scratch. I had come a long way from building plain and boring html files to a fully automated site that you can even find on google. I am now comfortable reading complex documentation – thanks to some of the complex documented API'S I used.

PROBLEMS FACED - BACKEND

ECONNRESET

It means the other side of the TCP connection abruptly closed it's end of the connection. The backend worked fine on local system. The TMDB api - I guess is banned in America - so the backend sends a first request to the API and then the connection is closed off.

We need to restart the server and make a whole new fresh request to the API. Could not find any solution to this unless using a VPN which might not be possible once I host the backend on heroku. I changed my region to Europe and surprisingly it worked.

BEARER TOKEN AUTHENTICATION

This was not a problem but rather something that I didn't know. Basically, it is a HTTP authentication scheme and a security token (mostly private) is assigned to each user (in lamen terms – given access to this content to the owner of this token). You need to include this token in the authorization header of a get request.

Reference = (What are bearer tokens = https://www.youtube.com/watch?v=6BPEQU53HgA)
(How to set bearer token in axios = https://www.youtube.com/watch?v=-u04JD5Eo-c)

PROBLEMS FACED - FRONTEND

CORS ERROR

Got to know what this infamous error is about - Every browser blocks this by default as part of its security model. It can be allowed if the client (frontend side) request Origin header matches the server(backend side) response Access-Control-Allow-Origin header.

For now I am allowing all origins to access the backend, so the cors origins header is set to "*" in the backend. Check server.js file for the cors origin header.

Reference = https://www.youtube.com/watch?v=PNtFSVU-YTI

REQUEST DATA FROM BACKEND API

I have a function say fetchData that takes a URL and returns a list of object, which I import in a react component. fetchData makes request to backend which takes time and the state is not updated. So we render a blank list of cards. So fetchData needs to return a promise instead of data – when the promise resolves then we set the state and react renders it.

WRONG ONE – return data.

CORRECT ONE – returns promise – a subtle difference – see line 4

REACT IS ANNOYING

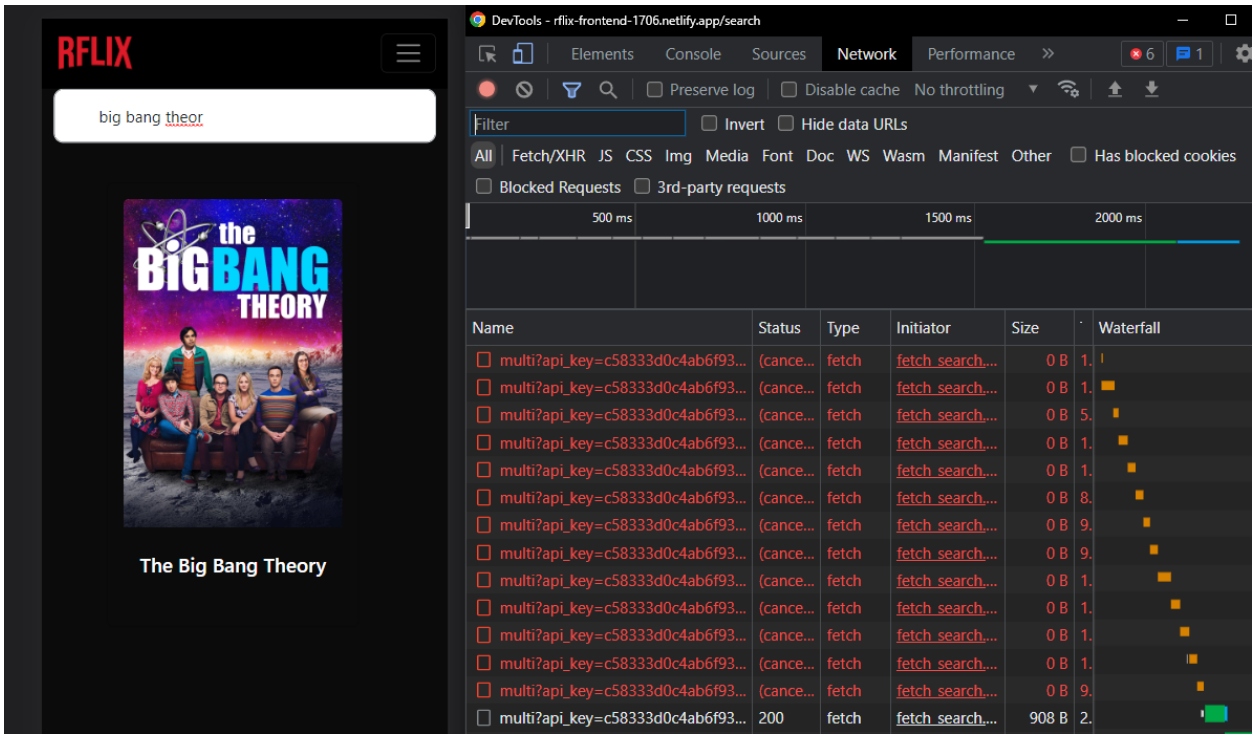
Yeah, sometimes in the development process I got stressed about it. Here's why - It auto compiles when some code is changed but when env files are added or changed it doesn't update itself to take on the new values. Its case sensitive - like allowfullscreen is actually allowFullScreen in youtube embed player.

MAKING THE SITE RESPONSIVE

Nothing to say here I guess. Nightmare for every frontend developer. But React's conditional rendering helped me a lot.

SOME NEW THINGS I LEARNT

1. CANCEL TOKEN - ABORT CONTROLLER FOR FETCH API - Basically the search bar makes request to backend for every change of character in search bar (google like) but sometimes we need to wait for the user to complete typing whole word or a keyword - See SearchPage.js - Watch - https://www.youtube.com/watch?v=0Be9Ez14e6M - Refer to https://developer.mozilla.org/en-US/docs/Web/API/AbortController/AbortController



As you can see the client made request to backend for every change of character but cancels the ongoing request if user types in next character in a given time limit. This is done to prevent the backend server from overloading with get requests. Also we can provide good results even if user types say only big – so client will only send request big.

2. How to extract url from tab = window.location.href
3. Implemented firebase google authentication - see firebaseAuth in components - REFERENCE (Basic tutorial = https://www.youtube.com/watch?v=vDT7EnUpEoo) (Protected routes = https://www.youtube.com/watch?v=6kgitEWTxac)
4. Optimizing for SEO - (Favicon generator, generate and paste the code in pubic folder and extract the icons there = https://www.favicon-generator.org/) (Compress static images to display = https://compressjpg.net/) (Provide helpful & specific description of images in alt attribute of image element) (What is SEO = https://www.youtube.com/watch?v=gmB_TC92I8w) (Use google search console = https://www.youtube.com/watch?v=HetaNOE5378)
5. Implementing serverless email service on the frontend via email js - (Reference = https://www.youtube.com/watch?v=NgWGllOjkbs)

DEPLOYING - WHY ?

The whole idea for my site was for it to be accessible everywhere. So from the starting it was my firm resolution that the site should be hosted online and viewable in about just a click.

However, the journey how deploying was not easy. Countless errors faced and solved all thanks to stack overflow.

FUTURE IDEAS

As I stated earlier, I don't support piracy. I plan to make a single subscription plan for all the OTT platforms so that users don't have to buy multiple plans. RFLIX would become the hub for entertainment.

My project is heavily dependent on API's and the minute they start removing their free plans – RFLIX would be down. I plan to make a stand alone video chat application with socket.io and integrate it with RFLIX – for now I use daily.js for video chats.