

Netflix Clone

①

Steps

- Ⓐ - Get TMDB API key
- Ⓑ - Create a react app
- Ⓒ - Get all the Movies
- Ⓓ - Build the Rows
- Ⓔ - Build the Banner
- Ⓕ - Build the nav bars
- Ⓖ - Add Trailers popups.

Step Ⓐ

- * Make a Account
- * Make a request to TMDB, → they will give a bunch of information about different film.

↓
Eg: netflix Original
Trending now

Top Rated movie etc.

We just pull information & display it.

Step Ⓑ

* npx Create-react-app netflix-clone

↓
has to be in lower case.

Tip → When ever, we write a Command in terminal with **-g**, it means, we need to write **Sudo**.

Sudo → gives admin privileges

Axios : Something that we can use to play around with API's

* promise based HTTP client for the browsers & node.js

Step ③

⇒

Create a file Request.js **request.js**

* Component → Should be upper case

* functional Module → lower case.

⇒

Once you have Created Request, we are going to pair with another file → **axios.js**

Every single request, from request.js, is going to have same starting URL

→ Every single Row in our clone will be One Component. ②

Eg: Netflix Original is One Component.

Trending Now is another Component & So on.

* So we build Each Row in `app.js`
To build each row, we need a Component `Row.js` → Create that in Src file the add `<Row>` in `app.js`.

Step ①

*

`Row.js`

`file`

* We prop to grab those title from `app.js`

* We state for movies = to store information about movies

*

`App.js`

In row, we add `fetchUrl`, because each row will have a different request then trending now etc.

So we request URL from `request.js`

Row.js

we add the Effect.

Because when a row loads, we need to make a request to TMDB,

Basically we pull information, right when the row loads,

so based on fetchURL prop, our effect will load information from particular URL

- * If the bracket is blank in useEffect, we are basically saying, run once when the row loads, & don't run again.

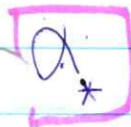
- * If we pass something in bracket for eg [movies], that means, run once when the row loads & load every time when movie changes.

- * import axios from our file './axios'
& not directly from dependency, this is because,
we export as default, we can use instance with whatever name I want to use.

* we make a async call, because we are sending request outside to a 3rd party Service,

UseEffect doesnot directly allow async, So we got to do in a special way. write little internal function to fetch ξ , then call it.

Awaity Says — when you make a request wait for promise to come back.

 If there is any Variable used in useeffect. that (eg fetchurl) pull data from outside, ξ is used in useeffect, then we have to include it in [].

If I dont pass Variable (for eg fetch URL), then it wouldn't re-render or Load back the URL

* Once we get the information from API, we render out the movies.

* **BEM** = Naming Convention for React

* we have to add movies/Row inside a Container because

we kind of wanted to scroll through them.

- * when you have array & you want to see data in a Tabular form, use `console.table()`

`console.table` is a lifesaver for arrays & objects.

- * Our `poster_path` should append to the base URL.
to append in js → we got to do string interpretation

- * `Use prettier Extension`

- Goto prettier → Install
- then click on setting beside prettier & go to Extension setting
- clear Everything in Search bar & type format
- check Format on Save

- * `Row.css`

`row-poster`
`row-poster.`

CSS



transition: transform 450ms;



this will make our CSS hit after some time assigned.

transition: transform 450ms; → main body



transform: scale(1.08); → hover body.

will make transition Super Smooth



• row-posters :: -webkit-scrollbar {
display: none;



keep the functionality of scrollbar, but don't display the scrollbar.

*

Row.js

When rendering the component which has more than few elements, the pass a **key** Value, so that react can identify each element uniquely.

Note:

Netflix Original thumbnail is bigger than rest of the row thumbnail.

*

Since we use `LargeRow` → insted of poster path, we will use backdrop-path.

Row.css

we can add universal css along with local css.

Eg: Everything gets a row-poster but if it is Large Row, give additional class called row-poster large

```
class Name = { 'row-poster' $ {LargeRow} & &  
  "row-poster large" }
```

Step ③

Banner :

Create a file Banner.js

To pick one item at random from array do
`Math.floor(Math.random() * request.data.results.length - 1)`

? → Optional Chaining

If something is undefined. Just don't crash the app

→ Truncate function add (:) 3 dot (...) when there is lot of text present.

truncate function takes a string & number, the number of text (characters) after which it needs to be truncated.

→ Add an empty div, so that you scroll & can have dimming ~~hover~~ effect.

Nav.js

* we use flexbox to bring avatar & logo side by side.

* When ever we are dealing with images, In order to keep the aspect ratio (ie it doesnot stretch, Or go wide) → we have to use → Object fit contain.

Nav.css

position: fixed;
will not work well with display: flex;
justify-content: space-between

step 6

Nav.js

Attach a listener to window & say, when you scroll down, do something.

for that use → UseEffect. Load Once when nav bar loads.

Our Listener listens to scroll & when scroll on y-axis is greater than 100px, set a piece of state, to `handle show = true`, else don't show.

& Every time we effect gets fixed, before you fix listener again, just remove it & then fix.

In that way you don't have more than one listener.

* `className = 'nav $ {show ? 'nav-black' : ''}`

always have/apply nav property & when show is true then also apply nav-black property.

Nav.CSS

transition-timing-function: ease in;
transition: all 0.5s;



add a transition to everything you do here.

Movie Trailer

Step 1) Trailer popup. - Row.js

We use 2 dependency.

① React YT (YouTube)

② react-youtube

③ movie-trailer

* Options for react-YT is present in documentation.

* One state to capture trailer URL.

This state will grab Video Id of each option - we click from youtube.

* If there is movie name, movie title, then return an URL. [promise]

* URLSearchParams → allow us to make get request on a variable/parameter.

It will go upto & v=(abc) & b=

↓
this will be extracted

Hosting Git hub

- ① npm start \Rightarrow Check if things are working fine locally & confirm
- ② npm run build \Rightarrow build production app
- ③ npm install gh-pages --save-dev \Rightarrow Save as dev dependency.
- ④ In package.json : first line.
"homepage" : "http://Ratbod-shubham.github.io/
repository name",

Now add 2 lines to script

```
"predeploy": "npm run build",  
"deploy": "gh-pages -d build",
```

- ⑤ git init
git remote add origin https:// \rightarrow .git

- ⑥ git status

- ⑦ git add .

- ⑧ git Commit -m "commit"

- ⑨ npm run deploy [log in Cred]

- 10) git push -U origin master

\rightarrow Go to github >> ^{repo}Setting >> Github pages