React - 5 IMDB Clone React Website

We will be creating an IMDB clone, where we will fetch Real-time trending movies data and will show it in grid format, we will be designing the whole application by using Tailwind CSS which we have already set up in our previous class

Features of the project

The following will be the features of our IMDB clone:

The user will be able to view the list of the latest & trending movies (IMDB API)
User can create their own separate watchlist
User can filter movies according to genre
User can sort movies according to ratings
Pagination will be implemented to move from one page to another to get updated data of other movies
Search feature will be there for movies.
We will be deploying this to netlify

title: Project setup and overview

undersanding Project structure

1. main.jsx: entry file for our project . It is the place from where our project will start executing
2. App.js : This is the react component that will represent the whole IMDB application

title: Adding Top level components

1. Let's create route for these two . Let's create a directory named Components [we will be keeping all the components of our application here]
2. inside that let's create components for our two routes
    a. Home.jsx
    b. WatchList.jsx

```
function Home() {
 return <div>Home</div>;
}

export default Home;
```

```
function WatchList() {
   return (
     <div>WatchList</div>
   )
 }
 export default WatchList
```

3. let's create common navbar - NavBar

```
function NavBar() {
   return (
     <div>NavBar</div>
   )
 }
 export default NavBar
```

4. let's now add these two pages and NavBar into our app.jsx.

```
import { useState } from "react";
import reactLogo from "./assets/react.svg";
import viteLogo from "/vite.svg";
import "./App.css";
import Home from "./components/Home";
import WatchList from "./components/WatchList";
import NavBar from "./components/Navbar";

function App() {
 return (
   <>
     <NavBar />
     <Home />
     <WatchList />
   </>
 );
}

export default App;
```

5. What happens if I include NavBar component in camel case?

```
import { useState } from "react";
import reactLogo from "./assets/react.svg";
import viteLogo from "/vite.svg";
import "./App.css";
import Home from "./components/Home";
import WatchList from "./components/WatchList";
```

```
import navBar from "./components/Navbar";

function App() {
 return (
   <>
     <navBar />
     <Home />
     <WatchList />
   </>
 );
}

export default App;
```

a. Uppercase names are treated as React components. This tells React to look for a component defined or imported with that name.
b. Lowercase names are treated as standard HTML tags. For example, or is recognized by JSX as HTML elements, not components.
c. React will interpret <navbar /> as an HTML tag, not a React component, and will look for an HTML element named navbar, which does not exist.

6. Ensure we have BrowserRouter wrapping our whole component
7. import browser router and wrap the whole application inside it
8. Update app.jsx with Routes

```
import { useState } from "react";
import reactLogo from "./assets/react.svg";
import viteLogo from "/vite.svg";
import "./App.css";
import Home from "./components/Home";
import WatchList from "./components/WatchList";
import NavBar from "./components/Navbar";
```

```
import { Routes, Route } from "react-router-dom";

function App() {
 return (
   <>
     <NavBar />
     <Routes>
       <Route path="/" element={<Home />} />
       <Route path="/watchlist" element={<WatchList />} />
     </Routes>
   </>
 );
}

export default App;
```

Navbar

1. Let's start with NavBar component first.
2. Navrbar will have hyperlinks. Which component do we use to create links in react-router-dom.
    a. answer : Link
3. Asset folder has the logo which we plan to use
4. As discussed before let's add these three links to our NavBar component

```
import { Link } from "react-router-dom";
import Logo from "../assets/Logo.png";
function NavBar() {
 return (
   <>
     <Link to="/">
```

```
      <img src={Logo} />
    </Link>

    <Link to="/">Movies</Link>

    <Link to="/watchlist">Watchlist</Link>
   </>
 );
}
export default NavBar;
```

styling Navbar

Cheatsheet- https://tailwindcomponents.com/cheatsheet/

1. First let us reduce the width of the image using tailwind css
    a. Adding a width of 50 px to the img tag
    b.
```
      <img className="w-[50px]" src={Logo} />
```
2. We will make the entire navbar as display- flex

```
import { Link } from "react-router-dom";
import Logo from "../assets/Logo.png";
function NavBar() {
 return (
   <div className="flex">
     <Link to="/">
     <img className="w-[50px]" src={Logo} />
     </Link>

     <Link to="/">Movies</Link>

     <Link to="/watchlist">Watchlist</Link>
   </div>
 );
}
export default NavBar;
```

3. Some margin-left for spacing b/w elements : space-x-8

```
function NavBar() {
 return (
   <div className="flex space-x-8">
```

4. Vertical - align: center -> items-center

```
<div className="flex space-x-8 items-center">
```

5. Adding padding to the left and top/bottom of our container ensures that our content isn't cramped against the edges of the container. The pl-3 class adds padding to the left side of the container, while the py-4 class adds padding to the top and bottom.

```
   <div className="flex space-x-8 items-center pl-3 py-4">
```

6. For the links, we use classes like text-blue-500, text-3xl, and font-bold to specify the color, font size, and font weight respectively.

```
<Link className="text-blue-500 text-3xl font-bold" to="/">Movies</Link>


     <Link className="text-blue-500 text-3xl font-bold"
to="/watchlist">Watchlist</Link>
```

7. If you find yourself repeating css for similar component, you can abstract them into a separate component or create a wrapper component if possible to apply style to all children

```
<div className="text-blue-500 text-3xl font-bold space-x-8">
       <Link to="/">Movies</Link>
```

```
        <Link to="/watchlist">Watchlist</Link>
      </div>
```

8. Final code for navbar so far

```
function NavBar() {
 return (
   <div className="flex space-x-8 items-center pl-3 py-4">
     <Link to="/">
       <img className="w-[50px]" src={Logo} />
     </Link>
     <div className="text-blue-500 text-3xl font-bold space-x-8">
       <Link to="/">Movies</Link>
       <Link to="/watchlist">Watchlist</Link>
     </div>
   </div>
 );
}
```

9. O/P

title: Banner and Movies component

We will first use placeholder image to design the component and then in the movies component we will introduce TMDB API

https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC_LeLBWr5LZMaAkWkF68

In banner we will have backgound image and on which bottom center we will have movie name.

so we will need two div's -> first one will used to show a background image and and an inner div that will be at the bottom ,Overall, this code snippet creates a responsive container with a background image, and a text element positioned at the bottom.

```
function Banner() {
   return (
     <div
       className="h-[20vh] md:h-[75vh] bg-cover bg-center flex items-end"
       style={{
         backgroundImage:
           `url(
https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC_LeLBWr5
LZMaAkWkF68)`,
       }}
     >
       <div className="text-white w-full text-center text-2xl">
        Placeholder movie
       </div>
     </div>
   );
 }
 export default Banner;
```

explaination of styling

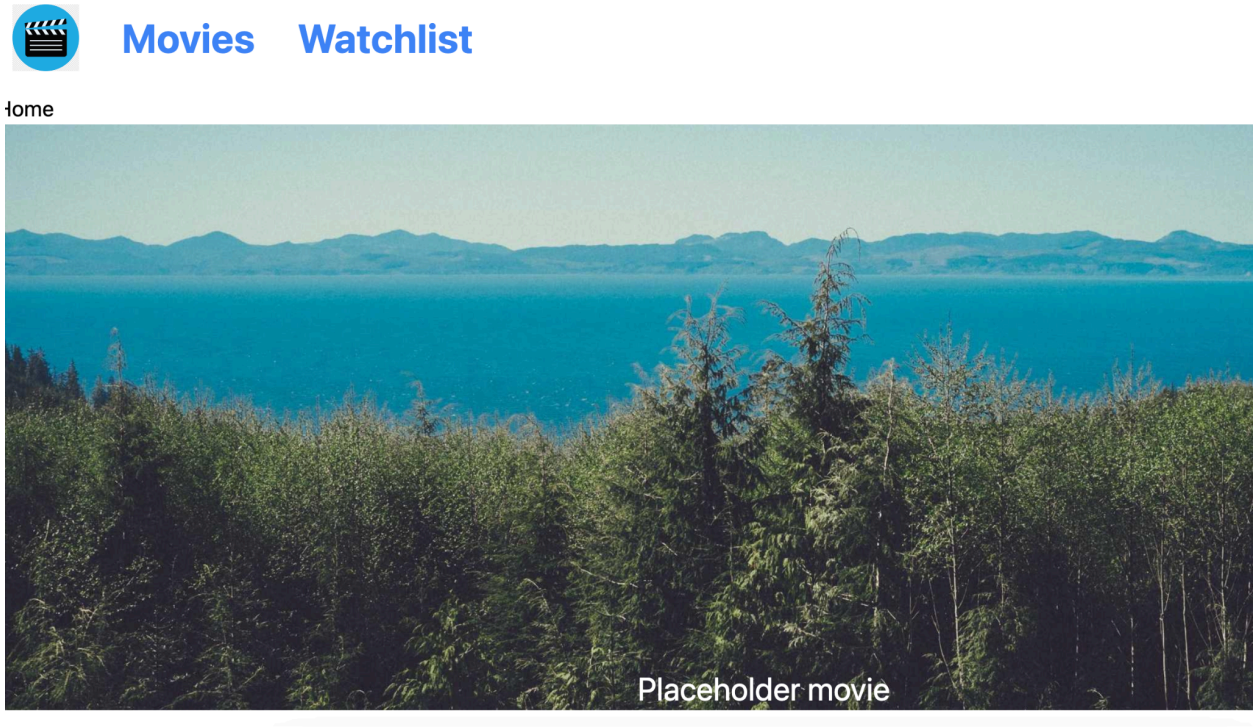
1. Inline Style: The style attribute is used to apply inline CSS styles to the div element. In this case, the backgroundImage property is

set to a URL that points to the background image. This image will be displayed as the background of the container.

2. Height: The h-[20vh] md:h-[75vh] classes are used to set the height of the div element. The first class, h-[20vh], sets the height to 20% of the viewport height (vh). This means the height of the container will be 20% of the viewport height, making it responsive to different screen sizes. The md:h-[75vh] class applies a height of 75% of the viewport height only on medium-sized screens and above (md breakpoint).
3. Flexbox Layout: The flex class is used to establish a flexbox layout for the container. This allows us to easily control the alignment and positioning of the child elements within the container.
4. Vertical Alignment: The items-end class is used to align the child elements to the bottom of the container. This ensures that the text inside the container is positioned at the bottom
5. Background Image: The bg-cover and bg-center classes are used to control the background image behavior. bg-cover ensures that the background image covers the entire container without distorting its aspect ratio, and bg-center ensures that the background image is centered within the container.
6. Child Element: Inside the container, there is a div element with the classes text-white w-full text-center text-2xl. This element contains the text "Placeholder movie". The classes are used to style the text, setting its color to white (text-white), making it take up the full width of the container (w-full), center-aligning the text

(text-center), and setting the font size to 2 times the base font size (text-2xl).

7. Import this in Home component



Movies component

1. to create the movie List component -> we will need to map it though the list of movies .
2. We will first use placeholder image the design the component and later on we will replace it TMDB API: https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC_LeLBWr5LZMaAkWkF68
3.

4. We will using placeholder image first then this will be replaced by actual API data.
5. Now the movie component is very simple -> it simply map through moviesArr

```
function Movies() {
 //     we will be using this static list of movies then we will replace it with
actual  data fetching logic
 const [movies, setMovies] = useState([
   {
     url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC_LeLBWr
5LZMaAkWkF68",
     title: "Movie 1",
   },
   {
     url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC_LeLBWr
5LZMaAkWkF68",
     title: "Movie 2",
   },
   {
     url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC_LeLBWr
5LZMaAkWkF68",
     title: "Movie 3",
   },
   {
     url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC_LeLBWr
5LZMaAkWkF68",
     title: "Movie 4",
   },
   {
     url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC_LeLBWr
5LZMaAkWkF68",
     title: "Movie 5",
   },
```

```
]);
return (
  <div>
    <div>
      <h1>Trending Movies</h1>
    </div>
    <div>
      {movies.map((movieObj) => {
        return (
          <div
            style={{
              backgroundImage: `url(${movieObj.url})`,
            }}
          >
            <div>{movieObj.title}</div>
          </div>
        );
      })}
    </div>
  </div>
);
}

export default Movies;
```

styling movies component

1. we will be placing
    a. trending movies to center of the screen
    b. want to make sure that all the movies are placed with the help of flex and are wrapped

```
import { useState } from "react";
function Movies() {
 //     we will be using this static list of movies then we will replace it with actual  data fetching logic
 const [movies, setMovies] = useState([
   {
```

```
    url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC
_LeLBWr5LZMaAkWkF68",
    title: "Movie 1",
  },
  {
    url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC
_LeLBWr5LZMaAkWkF68",
    title: "Movie 2",
  },
  {
    url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC
_LeLBWr5LZMaAkWkF68",
    title: "Movie 3",
  },
  {
    url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC
_LeLBWr5LZMaAkWkF68",
    title: "Movie 4",
  },
  {
    url:
"https://fastly.picsum.photos/id/10/2500/1667.jpg?hmac=J04WWC_ebchx3WwzbM-Z4_KC
_LeLBWr5LZMaAkWkF68",
    title: "Movie 5",
  },
]);
return (
  <div>
    <div className="text-2xl font-bold text-center m-5">
      <h1>Trending Movies</h1>
    </div>
    <div className="flex justify-evenly flex-wrap gap-8 ">
      {movies.map((movieObj) => {
        return (
          <div
            className="h-[40vh] w-[200px] bg-center bg-cover rounded-xl
hover:scale-110 duration-300 hover:cursor-pointer flex flex-col "
            style={{
```

```
              backgroundImage: `url(${movieObj.url})`,
            }}
          >
            <div className="text-white w-full text-center text-xl p-2
bg-gray-900/70 rounded-lg">
              {movieObj.title}
            </div>
          </div>
        );
      })}
    </div>
  </div>
 );
}

export default Movies;
```

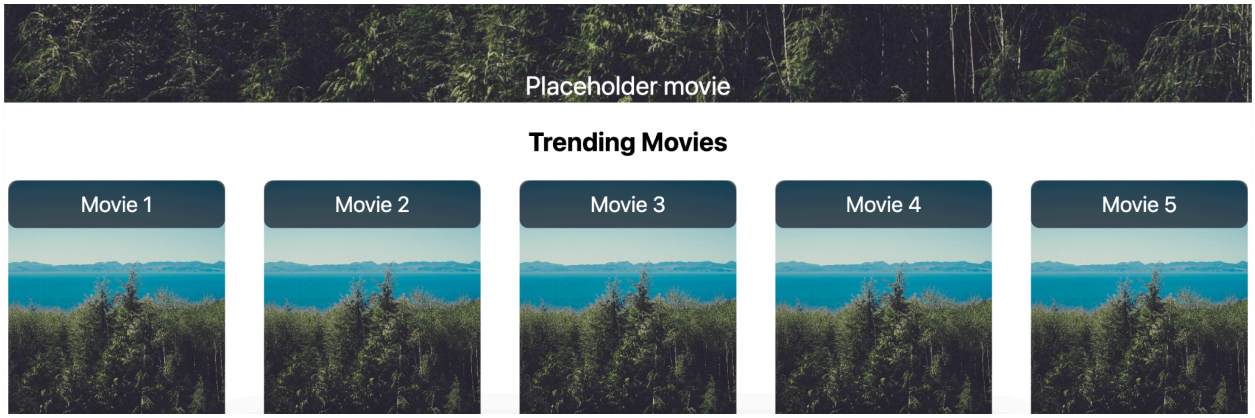
Explanation of styling

1. **Trending movies **:
    a. :
    b. text-2xl: Sets the text size to 2 times the original size.
    c. font-bold: Makes the text bold.
    d. text-center: Aligns the text to the center horizontally.
    e. m-5: Applies margin on all sides of the element.
2. Movie Cards:
    a. : This element serves as a container for displaying movie cards. It utilizes Flexbox for layout and responsiveness.

b. flex: Sets the display property to flex, allowing flexible layout options for child elements.
c. justify-evenly: Distributes the child elements evenly along the main axis (horizontal) with equal spacing between them.
d. flex-wrap: Allows the child elements to wrap onto multiple lines if they exceed the width of the container.
e. gap-8: Specifies the gap between each child element to be 8 units.

3. Movie Card:
    a. : This represents a single movie card.
    b. h-[40vh] w-[200px]: Sets the height to 40% of the viewport height and width to 200 pixels, ensuring consistent sizing for each card.
    c. bg-center bg-cover: Positions the background image at the center and covers the entire container.
    d. rounded-xl: Applies rounded corners to the card, giving it a softer appearance.
    e. hover:scale-110 duration-300: When hovered, the card scales to 110% of its original size with a smooth transition effect over 300 milliseconds.
    f. hover:cursor-pointer: Changes the cursor to a pointer when hovered, indicating that the card is clickable.

    g. flex flex-col justify-between items-end: Arranges the content inside the card in a column layout, aligning items to the end of the main axis (bottom in this case).
4. Movie Title:
    a. : This contains the title of the movie displayed on the card.
    b. text-white: Sets the text color to white.
    c. w-full: Sets the width of the to 100% of its parent container.
    d. text-center: Aligns the text to the center horizontally.
    e. text-xl: Sets the font size to extra-large.
    f. p-2: Applies padding of 2 units to all sides of the .
    g. bg-gray-900/70: Sets the background color to a semi-transparent dark gray (#444) with 70% opacity.
    h. rounded-lg: Applies rounded corners to the , matching the card's border radius.
5. Inline Styles:
    a. style={{ backgroundImage: url(${movieObj.url}) }}: Sets the background image of each movie card dynamically based on the url property of the current movieObj. This allows for individualized background images for each movie card.



title: Pagination and code refractoring

We will be using font-awesome to include icons in our application . Please add this script link to index.html

<script src="https://kit.fontawesome.com/589957875e.js" crossorigin="anonymous"></script>

```
<body>
   <div id="root"></div>
   <script type="module" src="/src/main.jsx"></script>
   <script src="https://kit.fontawesome.com/589957875e.js"
crossorigin="anonymous"></script>
 </body>
```

We will be needing two button -> back and forward and a text that reflects page number.Let's go inside our MovieList component and implement it

```
function Movies() {
 //     this state variable will be used to represent pagination
```

```
const [pageNo, setPageNo] = useState(1);

const handleNext = () => {
  setPageNo(pageNo + 1);
};
// go back handler
const handlePrevious = () => {
  if (pageNo == 1) {
    setPageNo(pageNo);
  }else {
      setPageNo(pageNo - 1);
  }
};
};
```

Add this div after movies array

```
<div>
      <div onClick={handlePrevious}>
        <i class="fa-solid fa-arrow-left"></i>
      </div>
      <div>{pageNo}</div>
      <div onClick={handleNext}>
        <i class="fa-solid fa-arrow-right"></i>
      </div>
    </div>
```

styling pagination

```
    <div className='bg-gray-400 p-4 h-[50px] w-full mt-8 flex justify-center
gap-2'>
      <div onClick={handlePrevious}>
```

This adds a color, padding of (.25 * 4) rem , height of 50px, full width, flex with justify center and a gap

Add some padding on the left and right of the handlers as well

```
<div className='bg-gray-400 p-4 h-[50px] w-full mt-8 flex justify-center gap-2'>
      <div onClick={handlePrevious} className='px-8'>
        <i class="fa-solid fa-arrow-left"></i>
      </div>
      <div>{pageNo}</div>
      <div onClick={handleNext} className='px-8'
```

Refactoring Pagination and Movie Card

Lets look at our Movies component. It has grown big

It is always a good practice to identify if you can break it in smaller and preferably independent components because it promotes reusability of code and improves code readability

Let's extract two components from our Movies Component

Pagination
MoviesCard

Pagination refactoring

1. create a file Pagination.jsx
2. intialize the component -> using rfce
3. import the component and pass required props to it
    a. our pagination will require setter function to go forward,backword and also to show the page number

```
import React from "react";
```

```
function Pagination({ nextPageFn, previosuPageFn, pageNumber }) {
 return (
   <div className="bg-gray-400 p-4 h-[50px] w-full mt-8 flex justify-center">
     <div onClick={previosuPageFn} className="px-8">
       <i class="fa-solid fa-arrow-left"></i>
     </div>
     <div>{pageNumber}</div>
     <div onClick={nextPageFn} className="px-8">
       <i class="fa-solid fa-arrow-right"></i>
     </div>
   </div>
 );
}

export default Pagination;
```

Updating Movies.jsx

Adding Pagination component

```
<Pagination
       nextPageFn={handleNext}
       previosuPageFn={handlePrevious}
       pageNumber={pageNo}
     />
```

Movie Card refactoring

1. create a file MovieCard.jsx
2. initialize the component -> using rfce
3. import the component and pass required props to it
    a. our movieCard will require movieObject

MovieCard.jsx

We can copy the markup being returned from the movies array loop

And create a separate component out of it

```
{movies.map((movieObj) => {
        return (
          <div
            className="h-[40vh] w-[200px] bg-center bg-cover rounded-xl
hover:scale-110 duration-300 hover:cursor-pointer flex flex-col "
            style={{
              backgroundImage: `url(${movieObj.url})`,
            }}
          >
            <div className="text-white w-full text-center text-xl p-2 bg-gray-900/70
rounded-lg">
              {movieObj.title}
            </div>
          </div>
        );
      })}
```

New component

```
function MovieCard({ movieObj }) {
 return (
   <div
     className="h-[40vh] w-[200px] bg-center bg-cover rounded-xl hover:scale-110
duration-300 hover:cursor-pointer flex flex-col justify-between items-end"
     style={{
       backgroundImage: `url(${movieObj.url})`,
     }}
   >
     <div className="text-white w-full text-center text-xl p-2 rounded-lg
bg-gray-900/70">
       {movieObj.title}
     </div>
   </div>
 );
```

```
}


export default MovieCard;
```

Call this component

```
{movies.map((movieObj) => {
          return <MovieCard movieObj={movieObj} />;
       })}
```