**Master React and OMBD API: The Complete Guide to Building a Movie App -** [Willochs Ojigbo](https://dev.to/willochs316) 4/12/2023

**Table of Contents**

1. [Introduction](https://dev.to/willochs316/building-a-movie-app-with-react-and-ombd-api-a-step-by-step-guide-2p33#introduction)
2. [How to Get an API Key](https://dev.to/willochs316/building-a-movie-app-with-react-and-ombd-api-a-step-by-step-guide-2p33#get-api)
3. [How to Set Up a React Project](https://dev.to/willochs316/building-a-movie-app-with-react-and-ombd-api-a-step-by-step-guide-2p33#setting-up)
4. [How to Safely Store API Keys using Environment Variables](https://dev.to/willochs316/building-a-movie-app-with-react-and-ombd-api-a-step-by-step-guide-2p33#storing-api)
5. [How to Implement the 'Add to Favorites' and 'Remove from Favorites' Feature](https://dev.to/willochs316/building-a-movie-app-with-react-and-ombd-api-a-step-by-step-guide-2p33#favorites)

**Introduction**

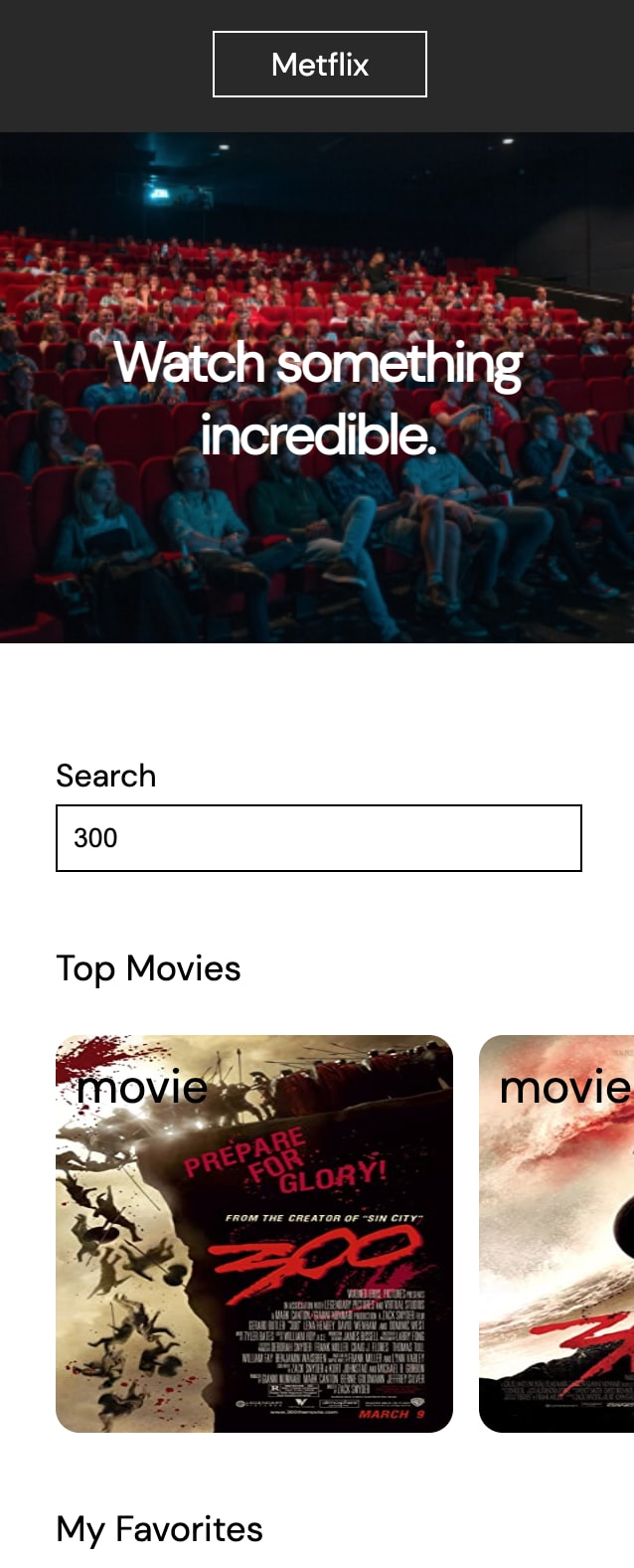
Developers can learn how to create a movie app by following the tutorial **"Build A Fun Movie App using React and the OMBD API."** Developers who wish to discover how to use APIs to construct web applications and fetch data will find this article to be a very useful resource.

The tutorial starts with setting up the development environment and creating a React project, then guides developers through each step of building a movie app. The next section describes how to retrieve movie data using the OMDb API and display it within the app.

The article also explains how to add further features to the app, such as movie search, movie details display, and adding a favorites list. The lesson is well-written, simple to follow, and contains code snippets and pictures to aid developers in understanding the topics. You'll make API requests and deal with the results received in this tutorial using the [Axios library](https://github.com/axios/axios/blob/main/README.md) because Axios automatically converts JSON data into JavaScript objects and supports Promises, the resulting code is simpler to read and debug.

Additionally, this tutorial will cover how to secure your API key by storing it in an environmental variable and also saving favorites movie to local storage.

Here's an example of what our project structure might look like:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--7H8DVBGO--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_800/https:/dev-to-uploads.s3.amazonaws.com/uploads/articles/dahifa7bco2tq5pju4n4.png)

For developers who want to learn how to create web applications using React and APIs, this article is a great resource. It offers a useful illustration of how to retrieve movie data using the OMDb API, which may be used in other API-based projects.

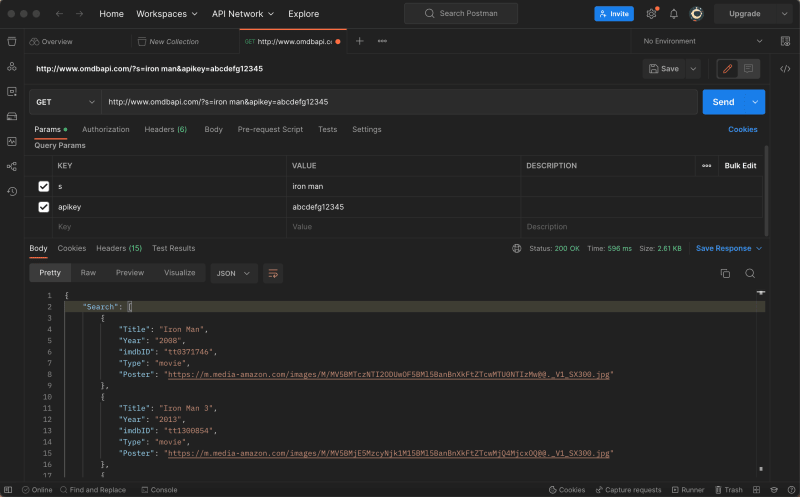
**How to Get an API Key**

To get an API Key for the OMDb API, follow these steps:

1. Go to the OMDb API website at <http://www.omdbapi.com/> and click on the "API Key" button in the top menu.
2. Fill out the required information in the form, including your name, email address, and the intended use of the API. Choose whether you are using the API for personal or commercial use.
3. After filling out the form, click on the "Submit" button to proceed.
4. You will receive an email from OMDb API with your API key. Keep your API key secure, as it is unique to your account and required to make API requests.
5. To use the API key, append it to the end of the API endpoint URL. For example, if your API key is "abcdefg12345", the URL to search for a movie would be: <http://www.omdbapi.com/?s=movie%20title&apikey=abcdefg12345>

Instead of using the "i" parameter to search for movies by their ID, we will replace it with the search "s" parameter. This will enable us to search for movies by title instead.

As demonstrated in the figure below using Postman [(If you haven't downloaded it yet, you can download it here.)](https://www.postman.com/downloads/).

[](https://res.cloudinary.com/practicaldev/image/fetch/s--ldJZJeKL--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_800/https:/dev-to-uploads.s3.amazonaws.com/uploads/articles/azqjwd4kz4948t9t1qlp.png)

**How to Set Up a React Project**

Type the following command in your terminal:

npx create-react-app metflix

This will create a new React project named "metflix" in the current directory.

Once the command finishes executing, navigate to the project directory by typing the following command:

cd metflix

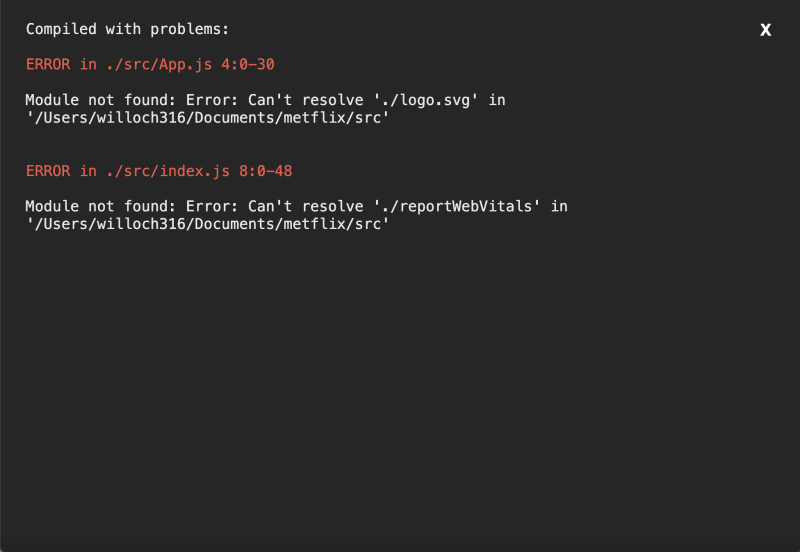
npm i axios

Start the development server by running the following command:

npm start

Remove the default App.test.js, logo.svg, reportWebVitals.js, setUpTests.js and index.css you have already created.

Notice, you'll come across an error on your local server:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--mkvRxfAQ--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_800/https:/dev-to-uploads.s3.amazonaws.com/uploads/articles/xi41vvnefjrcsjnzeel4.png)

A simple way to handle this error is basically to navigate to the index.js file and remove the index.css and reportWebVitals(). You will notice another compilation error at App.js because the compiler is trying to import logo.svg which we no longer have.

We will add a font CDN to the index.html file, you can use the following steps:

Open the index.html file in the public folder of your React project.  
Look for the <head> section of the file.  
Add the following code inside the <head> section:

<!--DM FONTS-->

<link

href="https://fonts.googleapis.com/css2?family=DM+Sans&display=swap"

rel="stylesheet"

/>

Navigate to **App.js** and add the following lines of code to **App.js**:

import "./App.css";

import NavBar from "./components/navbar/navBar";

const App = () => {

return (

<div className="App">

<div className="movie-container">

<NavBar />

</div>

</div>

);

};

export default App;

Create a new directory named **commons** inside the **components** directory located in the **src** folder. Inside the **commons** directory, add two new files named 'input.js' and 'typography.js'.

Inside **input.js** add the following codes;

// passed type, className, name, placeholder, onChange = () => {}, onBlur = () => {}, onFocus = () => {} as props.

const Input = ({

type,

className,

name,

placeholder,

onChange = () => {},

onBlur = () => {},

onFocus = () => {},

value,

}) => {

return (

<>

<input

type={type}

className={className}

name={name}

placeholder={placeholder}

required

onChange={onChange}

onBlur={onBlur}

onFocus={onFocus}

value={value}

/>

</>

);

};

export default Input;

Inside **typography.js** add the following codes;

// pass className, title as props

const Typography = ({ className, title }) => {

return (

<>

<p className={className}>{title}</p>

</>

);

};

export default Typography;

Navigate to the **components** directory, and create a new subdirectory named **navbar**. Within the **navbar** directory, include two files: **navBar.js** and **navBar.css**.

Add the following codes to **navBar.js**;

import React from "react";

import "./navBar.css";

import Typography from "../commons/typography";

import SearchInput from "../search/searchInput";

const NavBar = ({ searchValue, setSearchValue }) => {

return (

<header>

<div className="navbar">

<div className="main-navigation-container">

<Typography className="navbar-title" title="Metflix" />

</div>

</div>

<div className="navigation-image-container">

<div className="watch-something">

<Typography className="movie-text" title="Watch something" />

<Typography className="movie-text" title="incredible." />

</div>

</div>

<SearchInput searchValue={searchValue} setSearchValue={setSearchValue} />

</header>

);

};

export default NavBar;

Inside **navBar.css** add the following styles:

.navbar {

display: flex;

align-items: center;

justify-content: center;

height: 140px;

width: 100%;

color: #ffffff;

background: #292929;

position: absolute;

top: 0;

left: 0;

}

.main-navigation-container {

display: flex;

align-items: center;

justify-content: center;

border: 1px solid #ffffff;

font-size: 16.001px;

height: 60px;

width: 193px;

position: absolute;

left: 77px;

top: 40px;

border-radius: 0px;

}

.navbar-title {

font-size: 16.001px;

color: #ffffff;

}

.navigation-image-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

color: white;

height: 550px;

width: 100%;

position: absolute;

left: 0px;

top: 138px;

background: url("https://dev-to-uploads.s3.amazonaws.com/uploads/articles/lidwxp4yh2qttgw2eduv.png");

background-repeat: no-repeat;

background-position: center;

background-size: 100%;

}

.watch-something {

display: flex;

flex-direction: column;

align-items: baseline;

justify-content: center;

height: 282px;

width: 490px;

position: absolute;

left: 77px;

top: 109px;

}

.movie-text {

color: #ffffff;

font-family: DM Sans;

font-size: 72px;

font-weight: 700;

line-height: 94px;

letter-spacing: -0.05em;

text-align: left;

}

/\* Media Query iPad Pro 11 \*/

@media screen and (max-width: 834px) {

.navbar {

display: flex;

align-items: center;

justify-content: center;

height: 140px;

width: 100%;

color: #ffffff;

background: #292929;

position: absolute;

top: 0;

left: 0;

}

.main-navigation-container {

display: flex;

align-items: center;

justify-content: center;

border: 1px solid #ffffff;

font-size: 16.001px;

height: 60px;

width: 193px;

position: absolute;

left: 321px;

top: 40px;

border-radius: 0px;

}

.navbar-title {

font-size: 16.001px;

color: #ffffff;

}

.navigation-image-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

color: white;

height: 550px;

width: 100%;

position: absolute;

left: 0px;

top: 138px;

background: url("https://dev-to-uploads.s3.amazonaws.com/uploads/articles/lidwxp4yh2qttgw2eduv.png");

background-repeat: no-repeat;

background-position: center;

background-size: 100%;

}

.watch-something {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 282px;

width: 490px;

position: absolute;

left: 172px;

top: 109px;

}

.movie-text {

color: #ffffff;

font-family: DM Sans;

font-size: 72px;

font-weight: 700;

line-height: 94px;

letter-spacing: -0.05em;

text-align: center;

}

}

/\* Media Query Mobile Devices \*/

@media only screen and (min-width: 321px) and (max-width: 679px) {

.navbar {

display: flex;

align-items: center;

justify-content: center;

height: 67px;

width: 100%;

color: #ffffff;

background: #292929;

position: absolute;

top: 0;

left: 0;

}

.main-navigation-container {

display: flex;

align-items: center;

justify-content: center;

border: 1px solid #ffffff;

font-size: 16.001px;

height: 33.58px;

width: 108px;

position: absolute;

left: 107px;

top: 16px;

border-radius: 0px;

}

.navbar-title {

font-size: 16.001px;

color: #ffffff;

}

.navigation-image-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

color: white;

height: 257px;

width: 100%;

position: absolute;

left: 0px;

top: 67px;

background: url("https://dev-to-uploads.s3.amazonaws.com/uploads/articles/lidwxp4yh2qttgw2eduv.png");

background-repeat: no-repeat;

background-position: center;

background-size: 100%;

}

.watch-something {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 72px;

width: 273px;

position: absolute;

left: 23px;

top: 98px;

}

.movie-text {

color: #ffffff;

font-family: DM Sans;

font-size: 28px;

font-weight: 700;

line-height: 36.46px;

letter-spacing: -0.05em;

text-align: center;

}

}

We will create a search input that will enable us to implement our search functionality. In order to do this, we must first make a brand-new subdirectory called **search** in the **components** folder. We will add two new files called **searchInput.js** and **searchInput.css** to the **search** folder.

Open the **searchInput.js** and add the following lines of code:

import Input from "../commons/input";

import "./searchInput.css";

const SearchInput = ({ searchValue, setSearchValue }) => {

return (

<div className="searchbar-container">

<label id="search-label" htmlFor="search">

Search

</label>

{/\* The setSearchValue function updates the state of the component by setting the value of searchValue to the value of the search input field \*/}

<Input

className="search-input"

value={searchValue}

onChange={(event) => setSearchValue(event.target.value)}

placeholder="type to search..."

/>

</div>

);

};

export default SearchInput;

Inside **searchInput.css** add:

.searchbar-container {

display: flex;

flex-direction: column;

align-items: flex-start;

justify-content: center;

padding: 0px;

width: 90.6944444444%;

height: 89px;

gap: 4px;

position: absolute;

top: 751px;

left: 77px;

}

#search-label {

display: flex;

align-items: center;

font-family: DM Sans;

font-size: 24px;

font-weight: 400;

line-height: 31px;

letter-spacing: 0em;

text-align: left;

}

.search-input {

display: flex;

width: 100%;

height: 54px;

border: 1px solid #000000;

outline: 0;

flex: none;

order: 1;

flex-grow: 0;

padding-left: 12px;

}

::placeholder {

font-size: 16px;

font-style: italic;

text-transform: none;

}

/\* Media Query iPad Pro 11 \*/

@media screen and (max-width: 834px) {

.searchbar-container {

display: flex;

flex-direction: column;

align-items: flex-start;

justify-content: center;

padding: 0px;

height: 89px;

width: 81.5347721823%;

gap: 4px;

position: absolute;

left: 77px;

top: 751px;

border-radius: 0px;

}

#search-label {

display: flex;

align-items: center;

font-family: DM Sans;

font-size: 24px;

font-weight: 400;

line-height: 31px;

letter-spacing: 0em;

text-align: left;

}

.search-input {

display: flex;

box-sizing: border-box;

height: 54px;

width: 100%;

border: 1px solid #000000;

outline: 0;

flex: none;

order: 1;

flex-grow: 0;

padding-left: 8px;

}

::placeholder {

font-style: italic;

text-transform: none;

}

}

/\* Media Query Mobile Devices \*/

@media only screen and (min-width: 321px) and (max-width: 679px) {

.searchbar-container {

display: flex;

flex-direction: column;

align-items: flex-start;

justify-content: center;

padding: 0px;

width: 82.554517134%;

height: 59px;

gap: 4px;

position: absolute;

top: 380px;

left: 28px;

}

#search-label {

display: flex;

align-items: center;

font-size: 16px;

font-weight: 400;

line-height: 21px;

letter-spacing: 0em;

text-align: left;

}

.search-input {

display: flex;

box-sizing: border-box;

width: 100%;

height: 34px;

border: 1px solid #000000;

outline: 0;

flex: none;

order: 1;

flex-grow: 0;

padding-left: 8px;

}

::placeholder {

font-style: italic;

text-transform: none;

}

}

**How to Safely Store API Keys using Environment Variables**

Avoid hardcoding your API key in your code or any configuration files that could be accessed by others if you want to hide it in a React project. Instead, you can keep your API key in environment variables.

Create a new file called .env in the root of your project. This file should contain your API key like this:

REACT\_APP\_API\_KEY = your-api-key-goes-here

In your code, you can access this environment variable using the process.env object. For example, if your API key is stored in an environment variable called REACT\_APP\_API\_KEY, you can access it like this:

const apiKey = process.env.REACT\_APP\_API\_KEY;

Navigate to **App.js** where we will make use of this environment variable REACT\_APP\_API\_KEY:

import { useEffect, useState } from "react";

import "./App.css";

import Typography from "./components/commons/typography";

import MovieCard from "./components/movies/movieCard";

import NavBar from "./components/navbar/navBar";

import axios from "axios";

const App = () => {

const [movies, setMovies] = useState([]);

const [searchValue, setSearchValue] = useState("");

useEffect(() => {

const fetchMovies = async () => {

// GET request to the OMDB API using the Axios library

// searchValue is a variable that holds the user's search input

// The await keyword is used to wait for the response from the API before assigning it to the data variable.

// Acess the api variable using process.env.REACT\_APP\_API\_KEY

const { data } = await axios.get(

`http://www.omdbapi.com/?s=${searchValue}&apikey=${process.env.REACT\_APP\_API\_KEY}`

);

// Checks the if the data object that was returned from the OMDB API request contains a property called Search.

// If data.Search exists it means the response was successful, setMovies updates the movie state variable with the new search value

if (data.Search) {

setMovies(data.Search);

}

};

fetchMovies(searchValue);

}, [searchValue]);

return (

<div className="App">

<div className="movie-container">

<NavBar searchValue={searchValue} setSearchValue={setSearchValue} />

<div className="movie-set">

<div id="movie-category-container">

<Typography className="movie-category" title="Top Movies" />

</div>

<MovieCard movies={movies} setMovies={setMovies} />

</div>

</div>

</div>

);

};

export default App;

**How to Implement the 'Add to Favorites' and 'Remove from Favorites' Feature**

We'll build a favorites component and give it as a prop to the **MovieCard** component. The favorites component will then appear as an overlay on the **MovieCard**. This feature will probably include the user viewing or adding movies to their favorites list.

Create a new folder in the components folder called **favorites** inside it create **favoriteButton.js** and **favoriteButton.css**

Inside **favoriteButton.js** add the following:

import React from "react";

import "./favoriteButton.css";

export const AddFavorite = () => {

return (

<>

<svg

className="heart-icon"

width="40px"

height="40px"

viewBox="0 0 1024 1024"

version="1.1"

xmlns="http://www.w3.org/2000/svg"

>

<path

d="M927.4 273.5v-95.4h-87.9V82.8h-201v95.3h-87.9v95.4h-78.5v-95.4h-88V82.8H183.2v95.3H95.3v95.4H16.7v190.6h78.6v95.4h75.3v95.3H246v95.3h87.9v95.4h100.5v95.3h153.9v-95.3h100.4v-95.4h88v-95.3H852.1v-95.3h75.3v-95.4h78.5V273.5z"

fill="#E02D2D"

/>

</svg>

</>

);

};

export const RemoveFavorite = () => {

return (

<>

<svg

className="close-icon"

width="40px"

height="40px"

viewBox="0 0 24 24"

fill="none"

xmlns="http://www.w3.org/2000/svg"

>

<path

fillRule="evenodd"

clipRule="evenodd"

d="M5.29289 5.29289C5.68342 4.90237 6.31658 4.90237 6.70711 5.29289L12 10.5858L17.2929 5.29289C17.6834 4.90237 18.3166 4.90237 18.7071 5.29289C19.0976 5.68342 19.0976 6.31658 18.7071 6.70711L13.4142 12L18.7071 17.2929C19.0976 17.6834 19.0976 18.3166 18.7071 18.7071C18.3166 19.0976 17.6834 19.0976 17.2929 18.7071L12 13.4142L6.70711 18.7071C6.31658 19.0976 5.68342 19.0976 5.29289 18.7071C4.90237 18.3166 4.90237 17.6834 5.29289 17.2929L10.5858 12L5.29289 6.70711C4.90237 6.31658 4.90237 5.68342 5.29289 5.29289Z"

fill="#0F1729"

/>

</svg>

</>

);

};

Inside **favoriteButton.css** add these styles:

.heart-icon {

width: 40px;

height: 40px;

}

.close-icon {

width: 40px;

height: 40px;

background-color: white;

}

@media screen and (max-width: 834px) {

.heart-icon {

width: 40px;

height: 40px;

}

.close-icon {

width: 40px;

height: 40px;

}

}

@media only screen and (min-width: 321px) and (max-width: 679px) {

.heart-icon {

width: 20px;

height: 20px;

}

.close-icon {

width: 20px;

height: 20px;

background-color: white;

}

}

Navigate to the **components** directory in your project. Create a new folder called **movies** inside the **components** directory. Inside the **movies** folder, create two new files: **movieCard.js** and **movieCard.css**.

Add the following lines of code to **movieCard.js**:

import "./movieCard.css";

const MovieCard = ({

movies,

handleFavorite,

favouriteMovie: FavouriteMovie,

}) => {

return (

<div className="movie-row-container">

{movies?.map(

(movie) =>

// using regex to check if the Poster property of a movie object is an image file (JPEG, JPG, GIF or PNG format).

movie.Poster.match(/\.(jpeg|jpg|gif|png)$/) != null && (

<div className="movie-frame" key={movie.imdbID}>

<img className="movie-image" src={movie.Poster} alt="movie" />

<p className="movie-header">{movie.Type}</p>

<div

className="overlay-container"

onClick={() => handleFavorite(movie)}

>

<FavouriteMovie />

</div>

</div>

)

)}

</div>

);

};

export default MovieCard;

Add the following lines of styles to **movieCard.css**:

.movie-row-container {

display: flex;

align-items: flex-start;

height: 300px;

width: 1552px;

position: absolute;

top: 49px;

left: 0;

padding: 0;

gap: 13px;

overflow-x: scroll;

overflow-y: hidden;

}

.movie-row-container::-webkit-scrollbar {

display: none;

}

.movie-frame {

display: flex;

align-items: center;

justify-content: flex-start;

height: 300px;

width: 300px;

border-radius: 12px;

gap: 10px;

position: relative;

cursor: pointer;

transition: transform 0.2s;

}

.movie-frame:hover {

cursor: pointer;

transform: scale(0.95);

}

.movie-image {

width: 300px;

height: 300px;

border-radius: 12px;

}

.movie-header {

color: #000000;

font-weight: 400;

font-size: 24px;

position: absolute;

left: 10px;

top: 10px;

}

.movie-frame:hover .overlay-container {

opacity: 1;

}

.overlay-container {

display: flex;

align-items: center;

justify-content: center;

position: absolute;

background: rgba(0, 0, 0, 0.8);

width: 100%;

transition: 0.5s ease;

opacity: 0;

bottom: 0;

font-size: 20px;

padding: 20px;

text-align: center;

border-bottom-left-radius: 12px;

border-bottom-right-radius: 12px;

}

/\* Media Query iPad Pro 11 \*/

@media screen and (max-width: 834px) {

.movie-row-container {

display: flex;

align-items: flex-start;

height: 300px;

width: 1552px;

position: absolute;

top: 49px;

left: 0;

padding: 0;

gap: 13px;

overflow-x: scroll;

overflow-y: hidden;

}

}

/\* Media Query Mobile Devices \*/

@media only screen and (min-width: 321px) and (max-width: 679px) {

.movie-row-container {

display: flex;

align-items: flex-start;

height: 200px;

width: 626px;

position: absolute;

top: 49px;

left: 0;

padding: 0;

gap: 13px;

overflow-x: scroll;

overflow-y: hidden;

}

.movie-frame {

display: flex;

align-items: center;

justify-content: flex-start;

height: 200px;

width: 200px;

border-radius: 12px;

gap: 10px;

position: relative;

cursor: pointer;

}

.movie-image {

width: 200px;

height: 200px;

border-radius: 12px;

}

.overlay-container {

display: flex;

align-items: center;

justify-content: center;

position: absolute;

background: rgba(0, 0, 0, 0.8);

width: 100%;

transition: 0.5s ease;

opacity: 0;

bottom: 0;

font-size: 20px;

padding: 10px;

text-align: center;

border-bottom-left-radius: 12px;

border-bottom-right-radius: 12px;

}

}

Modify your **App.js** file by adding the following code snippet to it:

import { useEffect, useState } from "react";

import "./App.css";

import {

AddFavorite,

RemoveFavorite,

} from "./components/favorites/favoriteButton";

import Typography from "./components/commons/typography";

import MovieCard from "./components/movies/movieCard";

import NavBar from "./components/navbar/navBar";

import axios from "axios";

const App = () => {

const [movies, setMovies] = useState([]);

const [searchValue, setSearchValue] = useState("");

const [favorites, setFavorites] = useState([]);

useEffect(() => {

const fetchMovies = async () => {

// GET request to the OMDB API using the Axios library

// searchValue is a variable that holds the user's search input

// The await keyword is used to wait for the response from the API before assigning it to the data variable.

// Acess the api variable using process.env.REACT\_APP\_API\_KEY

const { data } = await axios.get(

`http://www.omdbapi.com/?s=${searchValue}&apikey=${process.env.REACT\_APP\_API\_KEY}`

);

// Checks the if the data object that was returned from the OMDB API request contains a property called Search.

// If data.Search exists it means the response was successful, setMovies updates the movie state variable with the new search value

if (data.Search) {

setMovies(data.Search);

}

};

fetchMovies(searchValue);

}, [searchValue]);

// The localStorage is queried for an item with the key "react-movie-app", which contain a stringified JSON object of the updated favorites. This value is parsed and then used to update the favorites state using the setFavorites function.

useEffect(() => {

const updatedFavorites = JSON.parse(

localStorage.getItem("react-movie-app")

);

setFavorites(updatedFavorites);

}, []);

const saveToLocalStorage = (items) => {

localStorage.setItem("react-movie-app", JSON.stringify(items));

};

const addFavoriteMovie = (movie) => {

// Use the || operator to ensure that favorites is not undefined before creating the new copy

const newFavorites = [...(favorites || [])];

// add movie to favorites array using push method

newFavorites?.push(movie);

// set the updated favorites array state

setFavorites(newFavorites);

saveToLocalStorage(newFavorites);

};

const removeFavoriteMovie = (movie) => {

const existingFavorites = favorites.filter(

(favorite) => favorite.imdbID !== movie.imdbID

);

setFavorites(existingFavorites);

saveToLocalStorage(existingFavorites);

};

return (

<div className="App">

<div className="movie-container">

<NavBar searchValue={searchValue} setSearchValue={setSearchValue} />

<div className="movie-set">

<div id="movie-category-container">

<Typography className="movie-category" title="Top Movies" />

</div>

<MovieCard

movies={movies}

setMovies={setMovies}

favouriteMovie={AddFavorite}

handleFavorite={addFavoriteMovie}

/>

</div>

<div className="favorite-movie-set">

<div id="movie-category-container">

<Typography className="movie-category" title="My Favorites" />

</div>

<MovieCard

movies={favorites}

favouriteMovie={RemoveFavorite}

handleFavorite={removeFavoriteMovie}

/>

</div>

</div>

</div>

);

};

export default App;