**SBA- REACT\_APP**

**A MOVIE SEARCH APP THAT CONTAINS THE FOLLOWING:**

**TECHNOLOGIES USED:**

* **VS-Code**
* **React**
* **Vite**
* **GitHub**

**CODE STRUCTURE:**

* **API usage/fetching**
* **React components**
* **React props**
* **React states and Hooks**
* **React styling**

**OTHER RESOURCES USED:**

* **React documentation**
* **W3School**
* **Youtube Tutorials**
* **Networking/ developers’ (‘data/assistance/help’)**
* **Google searches**

**Implementation:**

**API:**

1. I obtained an API from the <http://www.omdbapi.com> site. After submitting an API Key request which was sent to my email accont, I went ahead and fetched it to my code editor and started by:

* Creating a variable to store the API: const API\_URL = ‘url goes here’

- Then inside the app() rendering part, created the fetching code by first assigning the variable value: const searchMovies = async (title) => {

const response = await fetch(`${API\_URL}&s=${title}`);

const data = await response.json();

called the function here: setMovies =(data.Search);}

* so **&:s** adds a search parameter to the URL, leaving you to fill in the keyword.
* In (data.Search), Search comes from the API object. Whereas **setMovie** comes from the **useState**(once we’ve created/imported in our App component).

**REACT**

1. **First let’s create a new component called: Moviecard.jsx. inside of it let’s go through the following steps:**

* Let’s use the good **rcfe** emmet to make our lives ‘easier’. Let’s create a variable/function and inside of it destructure the **props/data** in order to render that data to the app component and then display it on the screen.

const MovieCard(that’s the name of the component, btw) = ({movie: {imdbID, Year, Poster, Title, Type}}) => { inside here goes the rest of the code that will make use of the **destructured props**  and make the data available to be rendered.

Return ( we can create a div tag or a react <></> it’s up to us…

<div className? Key={imdbID}>  
<div><p>{Year}</p>  
<img src={Poster !== “N/A” (or whatever the API site specifies) ? Poster : “via.placeholder.com/400(this is just a default online placeholder that helps or provides empty image placeholders to use whenever the api doesn’t provide an image)} alt={Title}/></div>  
<span>{Type}</span>  
<h3>{Title}</h3>

</div>

So, the first part gives data type: the year the movie was released, the picture of the image(s), the source, etc. The second part is for the type(movie or tv show) and title of each movie. We obviously must import this component.

1. **Let’s build our App.jsx component**

* We obviously have to import all the needed ‘pieces’: components, styling, hooks, etc.
* Create a variable that stores the API

Inside the App function, we will start by making use of the hooks:  
const App = () => {

const [XYZ, setXYZ] = useState("");

const [SPQ, setSPQ] = useState([])

* Then we have to use the useEffect hook: useEffect(() =>{searchMovies()},[]) (searchMovie is the name of the API function, and can be whatever name…)
* I’ts time to create our fetch function that will help ‘to fetch the data from the API: …const ‘function name (same name as the one in useEffect)’ async…setSPQ(data.Search) Search comes from the API object data.
* Once that is completed, inside the <return> let’s render each ‘component’

1. **Rendering**

* We can use ‘divs’ or ‘<></>’ to wrap our rendering.
* Create the searching input which contains the event handlers…<input…value={XYZ} onChange ={(e)=> setXYZ(e…)
* Let’s use another event handler to use the searching (icon or whatever that triggers the displaying or that helps to display the input/name of the movie that has been typed by the user…) <img src={searchIcon or whatever} onClick={() => searchMovies(1st hook )} <button onClick={()=> searchMovies(1st hook)
* Now let’s use a ternary operator to determine if a movie exists or not and based on that result, display to the screen. Then we will use the .map() method to iterate over the API data and create/get the data we need to be displayed. {SPQ?.length > 0 ? ***(Notice how the hook(SPQ) has a question mark next to it. For a better understanding on that, GOOGLE IT.)*** return the new array created by .map() (<div className="container"> {SPQ.map((movie) => (<MovieCard (or whatever name the external component has) key={movie.imdbID} movie={movie} />)} :(<div> <h2>No movies found</h2> </div>)} so where does movie come from? Well .map() is iterating over the data and returning **1 movie**; notice how we use then the dot notation(movie.imdbID) to access it and then dynamically pass it as a prop to the <MovieCard component movie={movie}. Inside of the <MovieCard component we have the movie prop: …function MovieCard = ({movie: {imdbID, Year,… we obviously used destructuring props.