MIT World Peace University

Full Stack Development
Third Year B. Tech, Semester 5

Design and Develop an interactive User Interface using React

Lab Assignment 5

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

September 22, 2023

Contents

1 Aim

Design and develop an interactive user interface using React.

2 Objectives

- Articulate what React is and why it is useful.
- Explore the basic architecture of a React application.
- Use React components to build interactive interfaces

3 Problem Statement

Design any UI using React.

4 Theory

4.1 React

Definition 1 ***React*** *is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.*

Figure 1: The React Logo.

4.2 Features

1. **Declarative:** React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

2. **Component-Based:** Build encapsulated components that manage their own state, then compose them to make complex UIs.

3. **Learn Once, Write Anywhere:** We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

4.3 Advantages

1. **Easy to Learn:** React is a JavaScript library, so we recommend that you have a basic understanding of the JavaScript language before diving into React. However, React is not a complex framework, and we believe that it is easy to learn.

2. **Reusable Components:** React allows you to create reusable UI components. This means that you can build encapsulated components that manage their own state, then compose them to make complex UIs.

3. **Virtual DOM:** React uses a virtual DOM (a JavaScript representation of the real DOM). The virtual DOM can improve the performance of a web application by updating the DOM efficiently.

4. **Performance:** React uses a virtual DOM, which is a JavaScript representation of the real DOM. The virtual DOM can improve the performance of a web application by updating the DOM efficiently.

5. **Testability:** React provides a set of utilities to test components and their behavior. You can use those utilities to test both React components and the DOM output.

6. **Developer Tools:** React has a great set of developer tools. These tools can help you create, debug, and test React applications.

4.4 History and Significance

1. React was created by Jordan Walke, a software engineer at Facebook, who released an early prototype of React called "FaxJS". He was influenced by XHP, an HTML component framework for PHP. It was first deployed on Facebook's News Feed in 2011 and later on Instagram in 2012.

2. React Native, which enables native Android, iOS, and UWP development with React, was announced at Facebook's React Conf in February 2015 and open-sourced in March 2015.

3. On April 18, 2017, Facebook announced React Fiber, a new core algorithm of React framework library for building user interfaces. React Fiber was to become the foundation of any future improvements and feature development of the React framework.

4. On September 26, 2017, React 16.0 was released to the public. On February 16, 2019, React 16.8 was released to the public. The release introduced React Hooks.

5. On August 10, 2020, React 17.0 was released to the public. The release is primarily for the React 17 release to be a drop-in replacement for previous versions.

5 Create React App (CRA)

Definition 2 ***Create React App*** *is a tool (built by developers at Facebook) that gives you a massive head start when building React apps. It saves you from time-consuming setup and configuration. You simply run one command and Create React App sets up the tools you need to start your React project.*

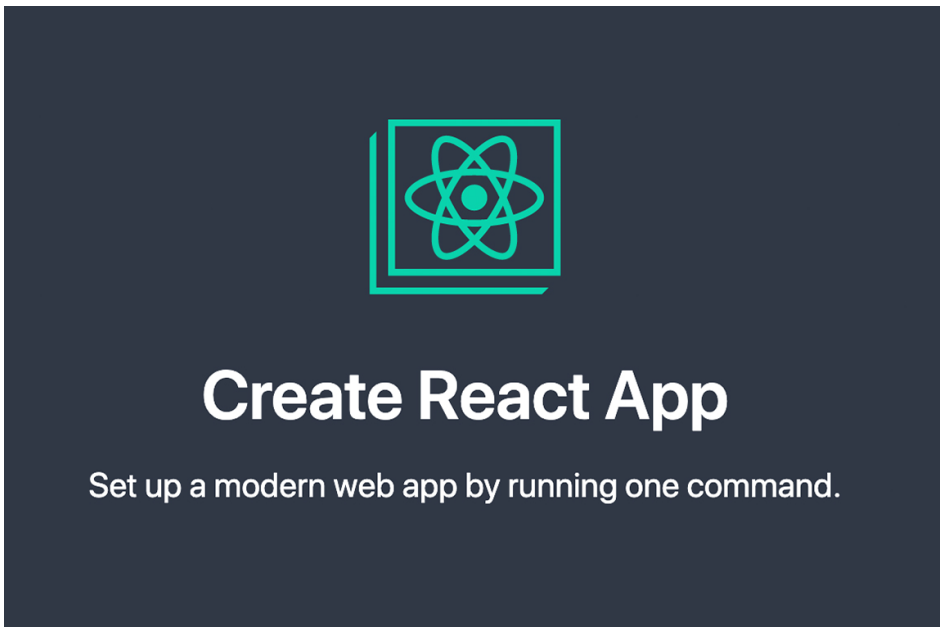

Figure 2: Create React App.

5.1 Modern Alternatives

When it comes to starting a new React project, Create React App (CRA) is a popular choice. However, there are several modern alternatives that offer additional features and flexibility. Here are some of them:

5.1.1 1. Vite

Vite is a build tool that focuses on fast development and optimized production builds. It supports React and provides an instant server start and hot module replacement (HMR) for a smooth development experience. Vite leverages ES modules for quicker bundling and can be used with various front-end frameworks, including React.

5.1.2 2. Snowpack

Snowpack is another build tool that offers near-instant development server startup. It's designed to work with modern web standards and allows you to import npm packages directly in your browser. Snowpack can be configured to work with React and provides a fast development environment.

5.1.3 3. Next.js

Next.js is a popular React framework that provides server-side rendering (SSR) and static site generation (SSG) capabilities out of the box. It simplifies routing, API handling, and SEO optimization.

Next.js is suitable for building not only single-page applications but also full-fledged websites.

5.1.4 4. Parcel

Parcel is a zero-config bundler that supports React and other popular front-end technologies. It's known for its simplicity and ease of use. Parcel automatically analyzes your project's dependencies and bundles them efficiently. It's a great choice for small to medium-sized projects.

5.1.5 5. Remix

Remix is a modern framework for building web applications, including React-based apps. Remix focuses on server-rendered React and offers features like routing, data loading, and client-side navigation. It aims to provide a great developer experience with strong conventions.

5.2 Steps to Create a React App

5.2.1 Prerequisites

Before you begin, ensure you have the following installed:

- **Node.js and npm:** Download and install Node.js from https://nodejs.org/. npm (Node Package Manager) is included with Node.js.
- **Git:** Install Git from https://git-scm.com/.

5.2.2 Steps

1. **Install create-react-app:** Open your terminal or command prompt and run the following command to install the 'create-react-app' globally:

   ```
   npm install -g create-react-app
   ```

2. **Create a new React app:** To create a new React app, run the following command, replacing 'my-react-app' with your desired app name:

   ```
   npx create-react-app my-react-app
   ```

3. **Navigate to the app directory:** Change your working directory to the newly created app directory:

   ```
   cd my-react-app
   ```

4. **Start the development server:** Run the following command to start the development server:

   ```
   npm start
   ```

   This will launch your React app in a web browser, and you can begin development.

5. **Edit the app:** Open the project directory in your code editor and make changes to the source code in the ‘src‘ folder. The app will automatically update in the browser as you save your changes.

6. **Build for production:** When you’re ready to deploy your app, use the following command to create a production build:

   ```
   npm run build
   ```

   This will generate optimized and minified files in the ‘build‘ folder.

7. **Deployment:** You can deploy your React app to various hosting platforms, such as GitHub Pages, Netlify, or a custom server. Follow the hosting platform’s specific instructions for deployment.

8. **Additional Configuration:** If needed, you can further configure your React app by editing the ‘package.json‘ and other configuration files in the project root.

6 Passing Data through Props in React

In React, you can pass data from a parent component to a child component using **props** (short for properties). Props allow you to send information down the component tree, enabling child components to receive and use data from their parent components.

Let’s consider a simple example of passing a person’s name from a parent component to a child component.

6.1 Parent Component

```
1  import React from 'react';
2  import ChildComponent from './ChildComponent';
3
4  function ParentComponent() {
5      const personName = "John";
6
7      return (
8          <div>
9              <h1>Parent Component</h1>
10             <ChildComponent name={personName} />
11         </div>
12     );
13 }
14
15 export default ParentComponent;
```

Listing 1: ParentComponent.js

In this example, we have a `ParentComponent` that defines a `personName` variable and renders a `ChildComponent`. We pass the `personName` as a prop called `name` to the `ChildComponent`.

6.2 Child Component

```
import React from 'react';

function ChildComponent(props) {
    return (
        <div>
            <h2>Child Component</h2>
            <p>Name: {props.name}</p>
        </div>
    );
}

export default ChildComponent;
```

Listing 2: ChildComponent.js

In the `ChildComponent`, we receive the `name` prop and display it within a paragraph.

7 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers or Interpreters: Brave Browser (Chromium v117.0.5938.88.)

8 Input and Output

1. A Webpage was created for searching for images. The user can enter a search term and the number of images to be displayed. The images are fetched from the Unsplash, Pexels and Pixabay APIs, and merged into a single array. The images are then displayed on the webpage.

2. This was done so as to resolve the problem of having to browser multiple webpages to find high quality images for making presentations. This webpage allows the user to search for images from multiple sources at once.

3. The webpage was created using React and Tailwindcss. The images are fetched using the Axios library. The screenshot of the webpage is shown below.

9 Screenshots

9.1 React Frontend

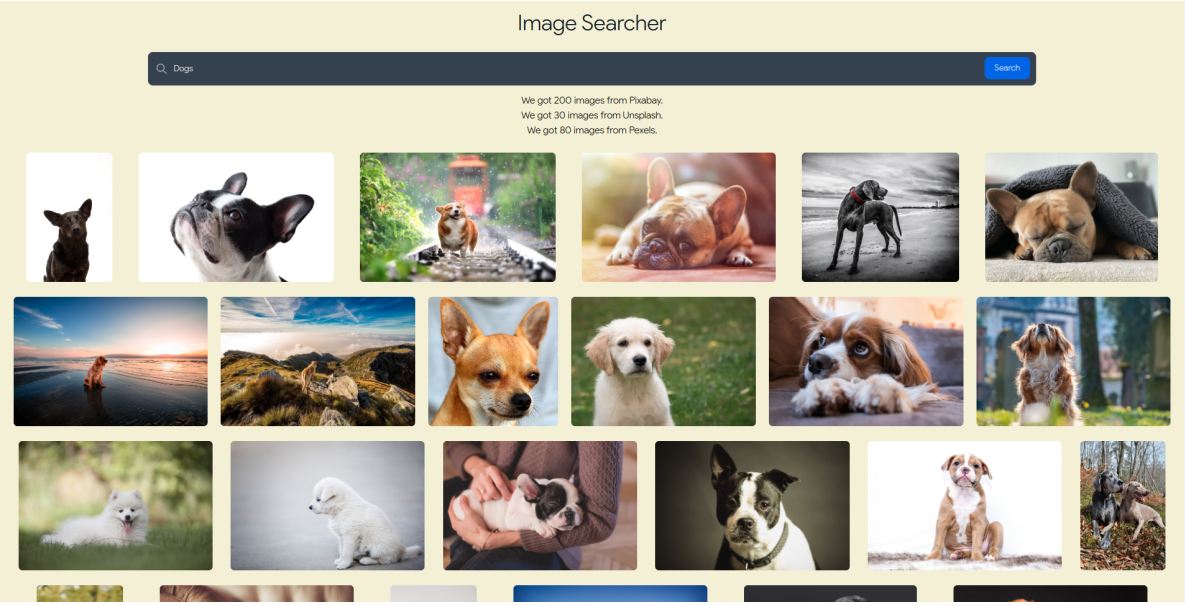

Figure 3: The Image Searcher Webpage, with search term: Dog.

10 Code

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './components/App';

ReactDOM.render(<App />, document.querySelector('#root'));
```

Listing 3: "index.js"

```
import React from "react";
import SearchInput from "./SearchInput";
import ImageList from "./ImageList";
import axios from "axios";
import { createClient } from "pexels";
import "../style.css";

const pixabay_api_key = "18846369-871c95044dd7c0f31f2b303e0";
const unsplash_api_key = "UBAXKPq5WgOxU9z7XjLZrt8B0hayG2dv8gh_vyGQg-0";
const pexels_api_key =
  "RfvKt0kQLXP7affcVkFgHWHcsRSD9CGKz91wOPBeLf7suLFEJJdjSWv3";

class App extends React.Component {
  state = {
    pixabay_images: [],
    unsplash_images: [],
    pexels_images: [],
  };

```

```
  onSearchSubmit = async (entry) => {
    const pixabay_response = await axios.get(
      `https://pixabay.com/api/?key=${pixabay_api_key}&q=${entry}&image_type=photo&pretty=true&per_page=200`
    );

    const numberOfPhotos = 30;

    const unsplash_response = await axios.get(
      `https://api.unsplash.com/search/photos/?query=${entry}&client_id=${unsplash_api_key}&per_page=${numberOfPhotos}`
    );

    const client = createClient(pexels_api_key);
    const query = entry;

    const pexels_response = await client.photos.search({
      query,
      per_page: 80,
    });
    console.log(pexels_response.photos);
    // const unsplash_response = {
    //  data: {
    //    results: [],
    //  },
    // };
    console.log(unsplash_response.data.results);
    // console.log(response.data.hits);
    this.setState({
      pixabay_images: pixabay_response.data.hits,
      unsplash_images: unsplash_response.data.results,
      pexels_images: pexels_response.photos,
    });
  };

  render() {
    return (
      <div className="flex flex-col gap-2">
        <h1 className="text-4xl text-center dark:text-gray-900 text-white m-4">
          Image Searcher
        </h1>
        <SearchInput onSearchSubmit={this.onSearchSubmit} />
        <div className="text-center">
          We got {this.state.pixabay_images.length} images from
          Pixabay.<br></br>
          We got {this.state.unsplash_images.length} images from
          Unsplash.
          <br></br>
          We got {this.state.pexels_images.length} images from Pexels.
        </div>
        <ImageList
          pixabay_images={this.state.pixabay_images}
          unsplash_images={this.state.unsplash_images}
          pexels_images={this.state.pexels_images}
        />
      </div>
    );
  }
}
```

```

export default App;
```

Listing 4: "app.js"

```
import React from "react";

const ImageList = (props) => {

  const pixabay_images = props.pixabay_images.map((image) => {
    return (
      <img
        key={image.id}
        src={image.largeImageURL}
        alt={image.tags}
        className="rounded-md max-h-52"
      />
    );
  });

  const unsplash_images = props.unsplash_images.map((image) => {
    return (
      <img
        key={image.id}
        src={image.urls.full}
        alt={image.alt_description}
        className="rounded-md max-h-52"
      />
    );
    });

    const pexels_images = props.pexels_images.map((image) => {
        return (
            <img
                key={image.id}
                src={image.src.large2x}
                alt={image.alt}
                className="rounded-md max-h-52"
            />
        )
    })

    const images_list = pixabay_images.concat(unsplash_images);
    images_list.concat(pexels_images);
  const shuffledImagesList = images_list.sort(() => Math.random() - 0.5);
  return (
    <div className="flex flex-wrap gap-y-6 gap-x-4 p-4 justify-evenly">
      {shuffledImagesList}
    </div>
  );
};

export default ImageList;
```

Listing 5: "ImageList.js"

```
import React, { Component } from "react";
import "../style.css";

```

```
4  class SearchInput extends Component {
5    state = {
6      entry: "",
7    };
8
9    onInputChange = (event) => {
10     this.setState({ entry: event.target.value });
11   }
12
13     onFormSubmit = (event) => {
14         event.preventDefault();
15         this.props.onSearchSubmit(this.state.entry);
16     }
17   render() {
18     return (
19              <form
20              onSubmit={this.onFormSubmit}>
21         <label
22           htmlFor="default-search"
23           className="mb-2 text-sm font-medium text-gray-900 sr-only dark:text-
     white"
24         >
25           Search
26         </label>
27         <div className="text-center">
28           <div className="relative w-3/4 mx-auto my-1 inline-block">
29             <div className="absolute inset-y-0 left-0 flex items-center pl-3
     pointer-events-none">
30               <svg
31                 aria-hidden="true"
32                 className="w-5 h-5 text-gray-500 dark:text-gray-400"
33                 fill="none"
34                 stroke="currentColor"
35                 viewBox="0 0 24 24"
36                 xmlns="http://www.w3.org/2000/svg"
37               >
38                 <path
39                   strokeLinecap="round"
40                   strokeLinejoin="round"
41                   strokeWidth={2}
42                   d="M21 21l-6-6m2-5a7 7 0 11-14 0 7 7 0 0114 0z"
43                 />
44               </svg>
45             </div>
46             <input
47               type="search"
48               id="default-search"
49               className="block w-full p-4 pl-10 text-sm text-gray-900 border
     border-gray-300 rounded-lg bg-gray-50 focus:ring-blue-500 focus:border-blue-500
      dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-
     white dark:focus:ring-blue-500 dark:focus:border-blue-500"
50               placeholder="Search Mockups, Logos..."
51               required=""
52               onChange={this.onInputChange}
53               value={this.state.entry}
54             />
55             <button
56               type="submit"
57               className="text-white absolute right-2.5 bottom-2.5 bg-blue-700
```

```
    hover:bg-blue-800 focus:ring-4 focus:outline-none focus:ring-blue-300 font-
    medium rounded-lg text-sm px-4 py-2 dark:bg-blue-600 dark:hover:bg-blue-700
    dark:focus:ring-blue-800"
58              >
59                Search
60              </button>
61            </div>
62          </div>
63        </form>
64      );
65    }
66  }
67
68  export default SearchInput;
```

Listing 6: "SearchInput.js"

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8" />
5      <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6      <link rel="stylesheet" href="input.css">
7      <link rel="stylesheet" href="style.css">
8      <meta name="viewport" content="width=device-width, initial-scale=1" />
9      <meta name="theme-color" content="#000000" />
10     <meta
11       name="description"
12       content="Web site created using create-react-app"
13     />
14     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
15     <!--
16       manifest.json provides metadata used when your web app is installed on a
17       user's mobile device or desktop. See https://developers.google.com/web/
     fundamentals/web-app-manifest/
18     -->
19     <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
20     <!--
21       Notice the use of %PUBLIC_URL% in the tags above.
22       It will be replaced with the URL of the `public` folder during the build.
23       Only files inside the `public` folder can be referenced from the HTML.
24
25       Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
26       work correctly both with client-side routing and a non-root public URL.
27       Learn how to configure a non-root public URL by running `npm run build`.
28     -->
29     <title>React App</title>
30   </head>
31   <body>
32     <noscript>You need to enable JavaScript to run this app.</noscript>
33     <div id="root"></div>
34     <!--
35       This HTML file is a template.
36       If you open it directly in the browser, you will see an empty page.
37
38       You can add webfonts, meta tags, or analytics to this file.
39       The build step will place the bundled scripts into the <body> tag.
40
41       To begin the development, run `npm start` or `yarn start`.
```

```
42        To create a production bundle, use `npm run build` or `yarn build`.
43      -->
44    </body>
45  </html>
```

Listing 7: "index.html"

11 Conclusion

Thus, we have learnt how to make a simple React Application. We have also learnt how to use Tailwindcss to style our React Application.

12 FAQ

1. *What are React states and hooks?*

   (a) **State** is a JavaScript object that stores a component's dynamic data and determines the component's behavior. Because state is dynamic, it enables a component to keep track of changing information in between renders and for it to be dynamic and interactive.

   (b) **Hooks** are functions that let you "hook into" React state and lifecycle features from function components. Hooks don't work inside classes — they let you use React without classes.

   Their Use Cases:

   (a) **State:** State is used for mutable data, or data that will change. Because state is mutable, it enables a component to keep track of changing information in between renders and for it to be dynamic and interactive.

   (b) **Hooks:** Hooks are used to add state and lifecycle methods to functional components. They also give you the ability to tie into React features from functional components.