Experiment No. 08

Title: Design a webpage using React Component API

Batch:SY-IT(B3) **Roll No:16010423076** **Experiment No:8**

Aim: To design a web page using React JS.

Resources needed: Notepad, any Web Browser and Internet.

Theory:

React is a declarative, efficient, and flexible JavaScript library for building user interfaces.

It lets you compose complex UIs from small and isolated pieces of code called components". We use components to tell React what we want to see on the screen. When our data changes, React will efficiently update and re-render our components. React does not manipulate the browser's DOM directly. Instead, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

Features of React

- − JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but **JSX** it is recommended.
- **Components** − React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.
- **Unidirectional data flow and Flux** − React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.
- **License** − React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

React structure

```
my-app/
  ├── README.md
  ├── node_modules/
  ├── package.json
  ├── public/
  │   ├── favicon.ico
  │   ├── index.html
  │   └── manifest.json
  └── src/
      ├── App.css
      ├── App.jsx
      ├── App.test.js
      ├── index.css
      ├── index.jsx
      ├── logo.svg
      └── serviceWorker.js
```

ReactJS - Environment Setup

1. First you need to install NodeJS
2. Second install ReactJS

Working with ReactJS

Setting up a React Environment
First of all you need to download NodeJs for your operating system version.
https://nodejs.org/en/download/
npm(node packet manager once you download NodeJs Then follow the following commands:

1) run the command "C:\Users\Your Name>npm install -g create-react-app"

2) run the command to create an application name helloworld "C:\Users\Your Name>npx create-react-app Helloworld"

3) run this command to get to the current directory "C:\Users\Your Name>cd helloworld"

4) run this command to start the react application "C:\Users\Your Name\helloworld >npm start" new browser window will pop up with your newly created React App! If not, open your browser and type localhost:3000 in the address bar.

Modify the React Application
Look in the helloworld directory, and you will find a src folder. Inside the src folder there is a file called App.js, open it and make changes to any HTML part.
You will be able to see the change on the newly opened browser
App.js File

```
import React, { Component } from 'react'; import logo from './logo.svg';
import './App.css';
```

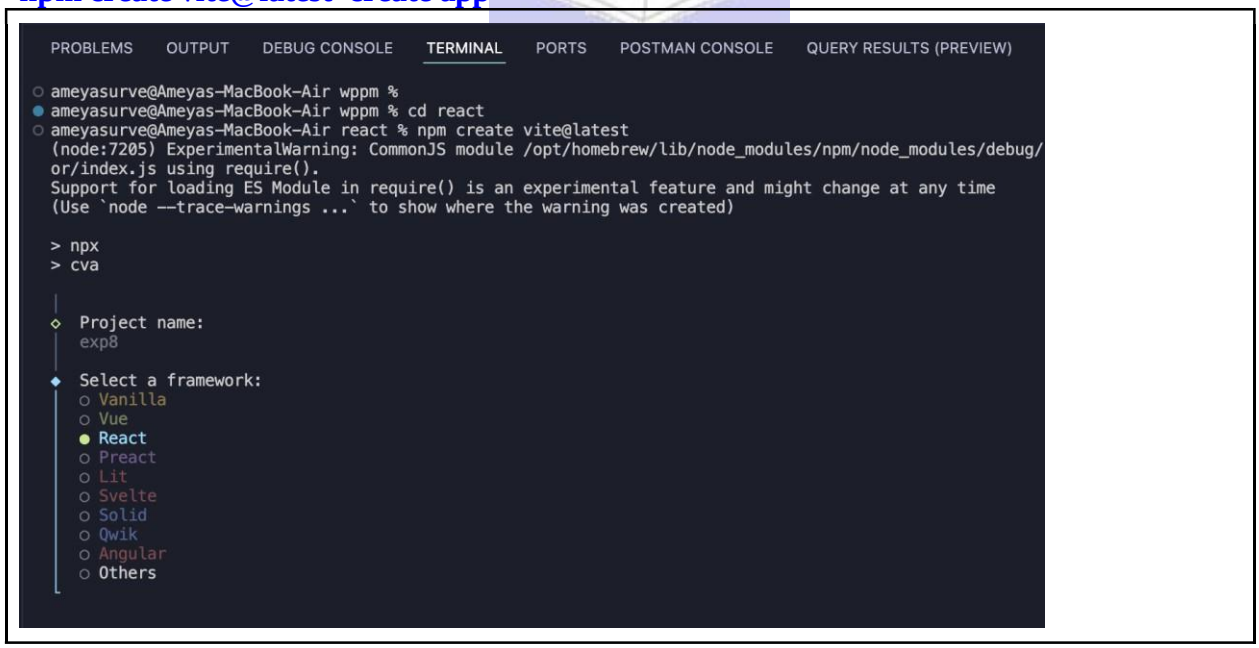
```
class App extends Component { render() {
return (
<div className="App">
<header className="App-header">
<img src={logo} className="App-logo" alt="logo" />
<p>
Edit <code>src/App.js</code> and save to reload.
</p>
<p> Hi </p>
<a
className="App-link" href="https://reactjs.org"
target="_blank" rel="noopener noreferrer"
>


Learn React
</a>
</header>
</div>
);
}
}
```

export default App;

Alternate way

React Docs:-
Create vite + react app

npm create vite@latest create app



```
> npx
> cva

◇  Project name:
│  exp8
│
◇  Select a framework:
│  React
│
◇  Select a variant:
│  JavaScript
│
◇  Scaffolding project in /Users/ameyasurve/Desktop/wp/wppm/react/exp8...
│
└  Done. Now run:

  cd exp8
  npm install
  npm run dev

ameyasurve@Ameyas-MacBook-Air react % cd exp8
ameyasurve@Ameyas-MacBook-Air exp8 % npm run dev

> exp8@0.0.0 dev
> vite

sh: vite: command not found
ameyasurve@Ameyas-MacBook-Air exp8 % npm install
(node:7371) ExperimentalWarning: CommonJS module /opt/homebrew/lib/node_modules/npm/node_modules/de
or/index.js using require().
Support for loading ES Module in require() is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)

added 150 packages, and audited 151 packages in 14s

30 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
ameyasurve@Ameyas-MacBook-Air exp8 %
```

App.jsx for hello world

```
import React from 'react';

function App() {
 return (
   <div>
     <h1>Hello World</h1>
   </div>
 );
}

export default App;
```

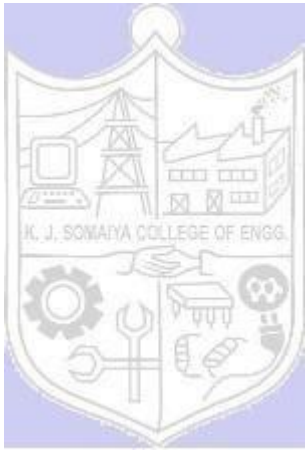
Main react things basics

- **Introduction to React**
- **Setting Up a React Project**
- **JSX Fundamentals**
- **Components**
- **Props**
- **State**
- **Handling Events**
- **Conditional Rendering**
- **Lists and Keys**
- **Forms**

- **Hooks**
- **Styling in React**
- **Common Patterns**

JSX Fundamentals

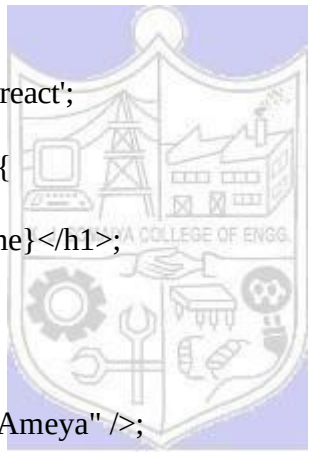
```
// Basic JSX example
const element = <h1>Hello, King!</h1>;

// JSX with expressions
const name = 'Ameya';
const element = <h1>Hello, {name}!</h1>;

// JSX with attributes
const element = <img src={user.avatarUrl} alt="User avatar" />;

// JSX with multiple elements needs a parent wrapper
const element = (
  <div>
    <h1>Hello!</h1>
    <p>React is better than vue</p>
  </div>
);

// Alternative using React Fragment
const element = (
  <>
    <h1>Hello!</h1>
    <p>React is better than vue</p>
  </>
);
```



Components

```
// Simple function component
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

// Arrow function component
const Welcome = (props) => {
  return <h1>Hello, {props.name}</h1>;
};

// Using the component
const element = <Welcome name="Ameya" />;
```

Class Components



```
import React, { Component } from 'react';

class Welcome extends Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}

// Using the component
const element = <Welcome name="Ameya" />;
```

Props

Props (short for properties) are how you pass data from parent to child components.

```
// Parent component
function App() {
return (
    <div>
      <Welcome name="Ameya" />
      <Welcome name="Yashvi" />
      <Welcome name="Akshay" />
    </div>
  );
}

// Child component
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Props are read-only and should not be modified within a component.

State

State is a component's internal data that determines how the component renders and behaves.

State in Class Components

```
import React, { Component } from 'react';
class Counter extends Component {
constructor(props) {
   super(props);
   this.state = {
   count: 0
   };
  }

  increment = () => {
   this.setState({ count: this.state.count + 1 });
  }

  render() {
   return (
     <div>
       <p>Count: {this.state.count}</p>
       <button onClick={this.increment}>Increment</button>
     </div>
   );
  }
}
```

State in Function Components (using Hooks)

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  function increment() {
   setCount(count + 1);
  }

  return (
   <div>
     <p>Count: {count}</p>
     <button onClick={increment}>Increment</button>
   </div>
  );
}
```

Handling Events

React events are named using camelCase and pass functions as event handlers.

```
// Event handling in function component
function ButtonClick() {
  const handleClick = () => {
    alert('Button was clicked!');
  };

  return <button onClick={handleClick}>Click me</button>;
}

// Event handling with parameters
function ButtonClick() {
  const handleClick = (name) => {
    alert(`Hello, ${name}!`);
  };

  return <button onClick={() => handleClick('Ameya')}>Say Hello WPl</button>;
}
```

Conditional Rendering

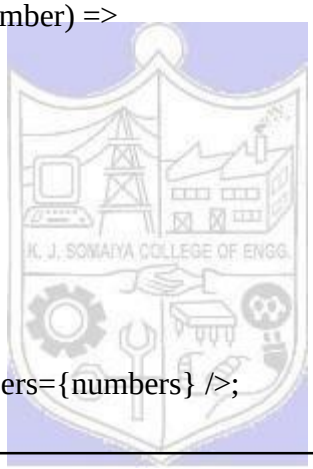
```
In React, you can conditionally render components based on the state of your application.//
Using if statements
function UserGreeting(props) {
  if (props.isLoggedIn) {
    return <h1>Welcome back!</h1>;
  }
  return <h1>Please sign in again.</h1>;
}

// Using ternary operators
function UserGreeting(props) {
return (
    <h1>
      {props.isLoggedIn ? 'Welcome back!' : 'Please sign in again'}
    </h1>
  );
}

// Using && operator
function Mailbox(props) {
  const unreadMessages = props.unreadMessages;
  return (
    <div>
```

```
    <h1>Hello!</h1>
    {unreadMessages.length > 0 &&
      <h2>
        You have {unreadMessages.length} unread messages.
      </h2>
    }
  </div>
 );
}
```

Lists and Keys

React uses keys to identify which items have changed, are added, or are removed in lists.

```
function NumberList(props) {
 const numbers = props.numbers;
 const listItems = numbers.map((number) =>
  <li key={number.toString()}>
    {number}
  </li>
 );

 return <ul>{listItems}</ul>;
}

const numbers = [1, 2, 3, 4, 5];
const element = <NumberList numbers={numbers} />;
```



Forms

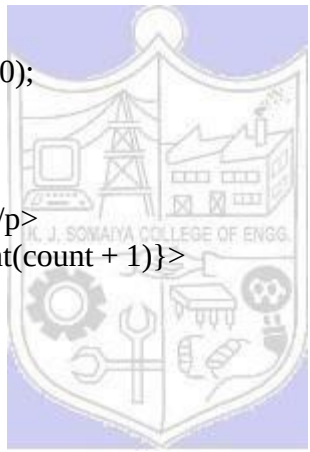
```
import React, { useState } from 'react';

function NameForm() {
 const [name, setName] = useState('');

 const handleChange = (event) => {
  setName(event.target.value);
 };

 const handleSubmit = (event) => {
  alert('A name was submitted: ' + name);
  event.preventDefault();
 };

 return (
```

```
    <form onSubmit={handleSubmit}>
      <label>
        Name:
        <input type="text" value={name} onChange={handleChange} />
      </label>
      <input type="submit" value="Submit" />
    </form>
  );
}
```

Hooks

Hooks are functions that let you "hook into" React state and lifecycle features from function components.

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```



useEffect

```
import React, { useState, useEffect } from 'react';

function Example() {
  const [count, setCount] = useState(0);

  // Similar to componentDidMount and componentDidUpdate
  useEffect(() => {
    document.title = `You clicked ${count} times`;

    // Optional cleanup function (similar to componentWillUnmount)
    return () => {
      document.title = 'React App';
    };
  }, [count]); // Only re-run the effect if count changes
```

```
  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Basic done now routing

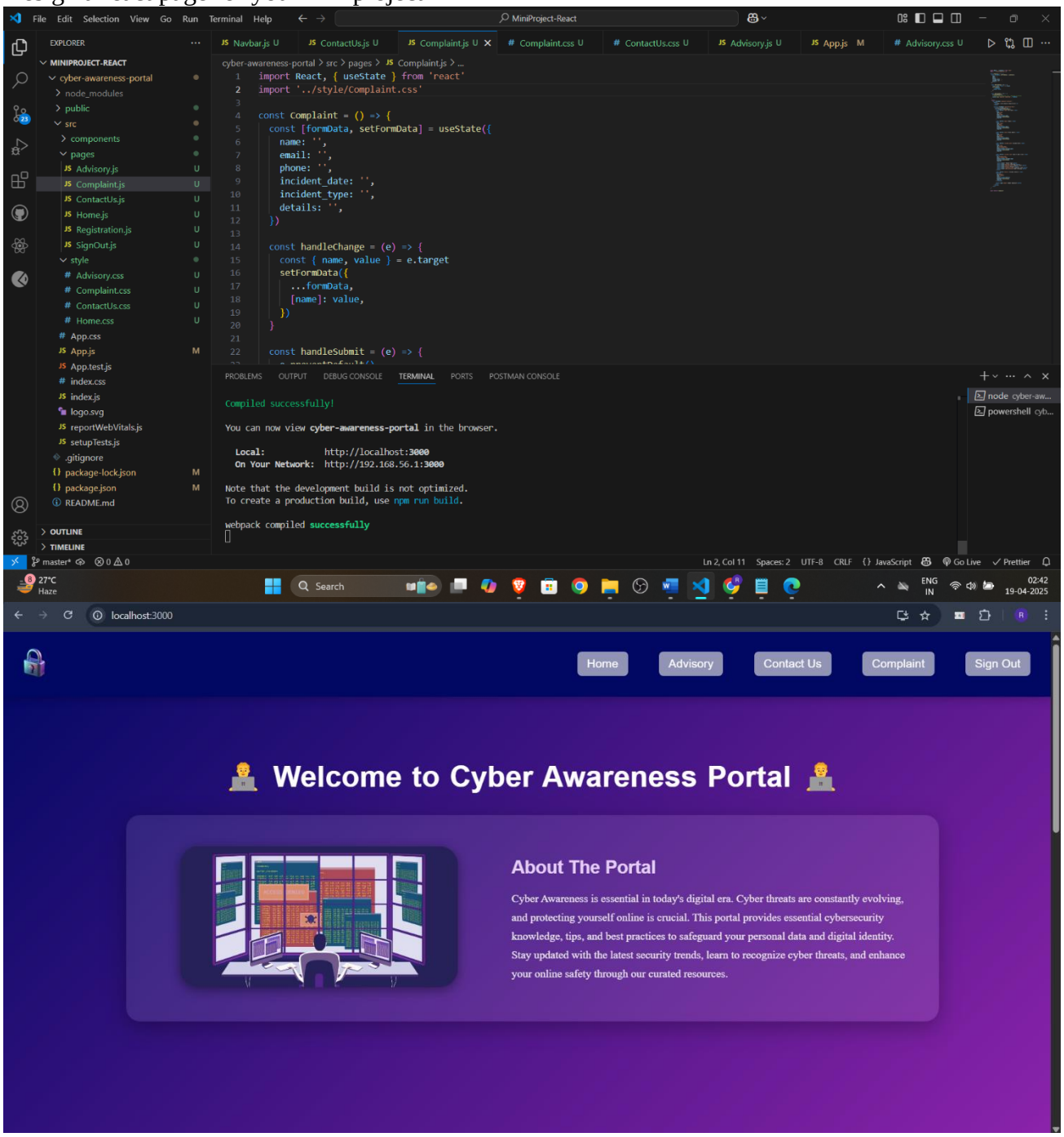
```
# For web
npm install react-router-dom
```

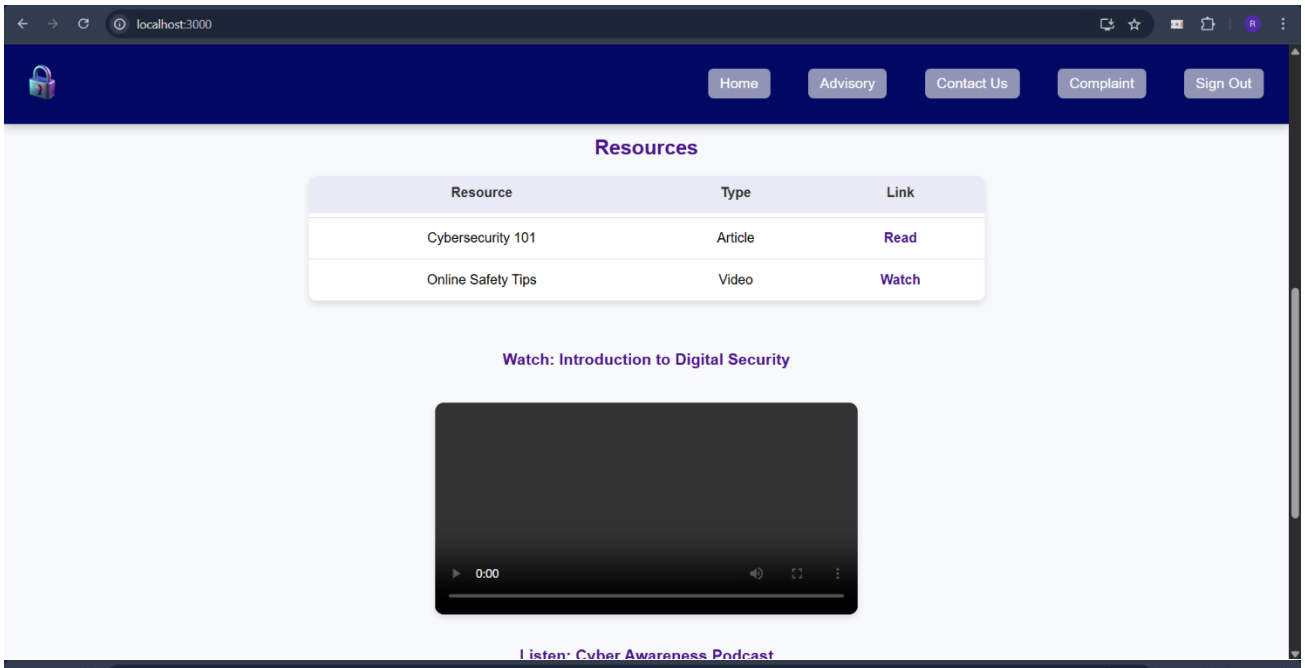
App.jsx

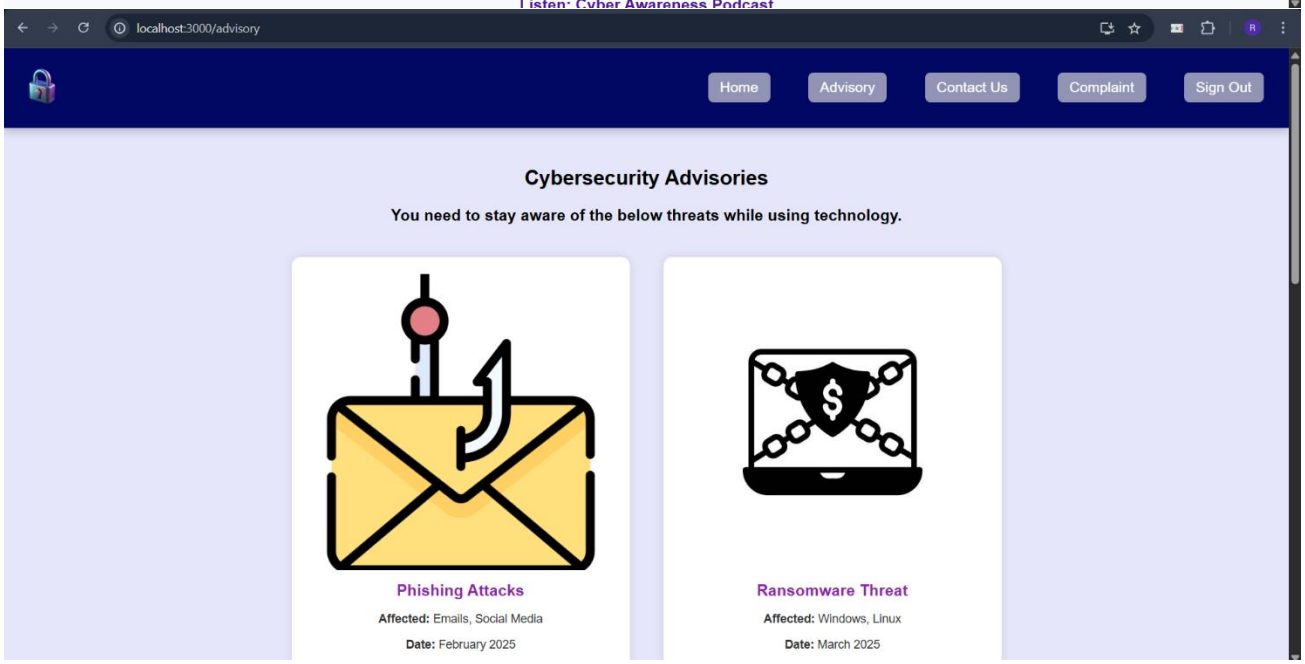
```
import React from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';

// Import your page components
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import NotFound from './pages/NotFound';
import Layout from './components/Layout';

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="about" element={<About />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NotFound />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

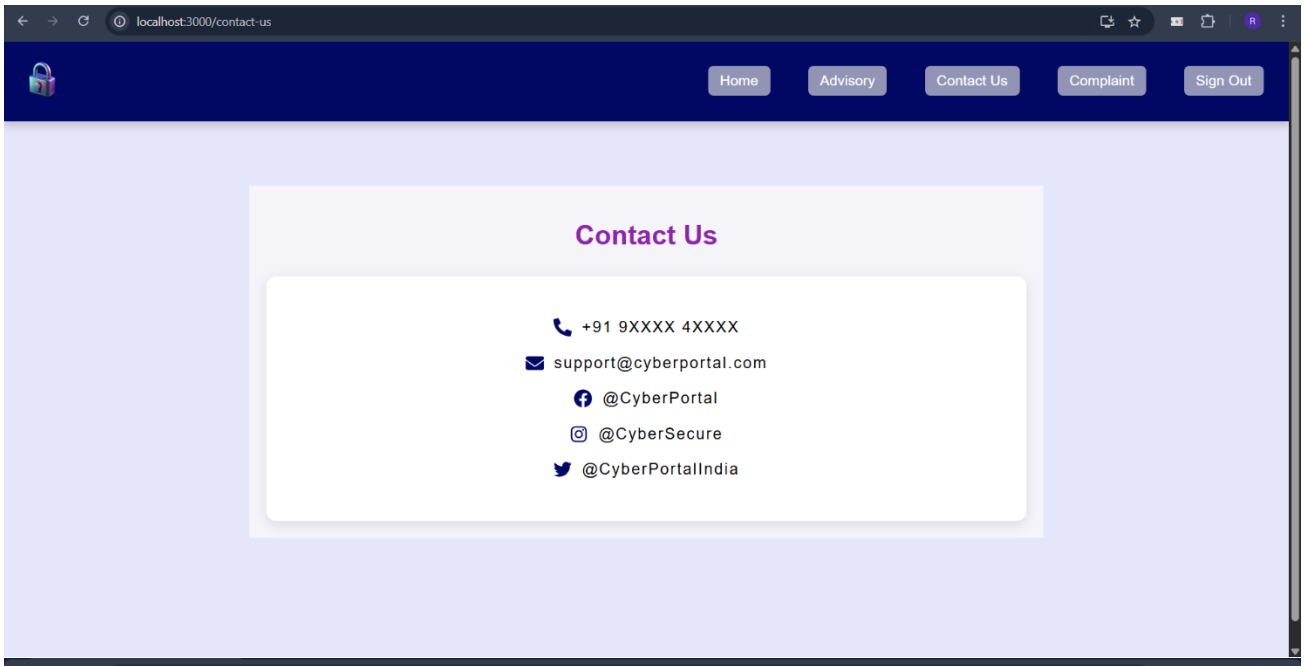
Activity:

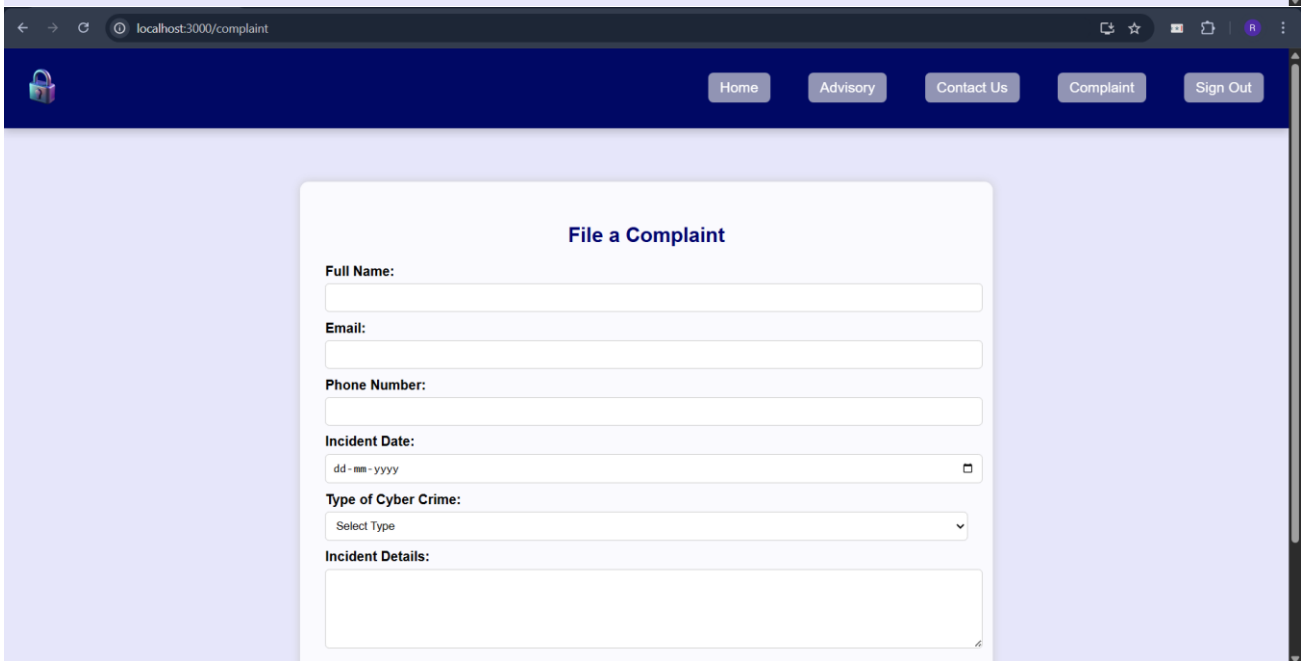
Design a react page for your miniproject




localhost:3000
Home
Advisory
Contact Us
Complaint
Sign Out
Resources
Resource
Type
Link
Cybersecurity 101
Article
Read
Online Safety Tips
Video
Watch
Watch: Introduction to Digital Security
0:00
Listen: Cyber Awareness Podcast


localhost:3000/advisory
Home
Advisory
Contact Us
Complaint
Sign Out
Cybersecurity Advisories
You need to stay aware of the below threats while using technology.
Phishing Attacks
Affected: Emails, Social Media
Date: February 2025
Ransomware Threat
Affected: Windows, Linux
Date: March 2025





Questions

1. What are the different components of ReactJS?
   ReactJS is based on components, which are like small building blocks used to create user interfaces. There are mainly two types of components: Functional Components and Class Components. Functional components are written using JavaScript functions, while class components use ES6 classes. Both can display data and handle user interactions. Inside these components, you often use JSX, which looks like HTML but allows you to write JavaScript code too. Along with components, React also uses props (short for properties) to pass data from one component to another, and state to store and manage local data inside a component. Other important concepts include hooks (like useState, useEffect) which add extra features to functional components, and context which helps in sharing data between components without passing props manually.

2. What is Virtual DOM? How virtual DOM Works? What is the purpose of render of react DOM?

The Virtual DOM is a smart idea used by React to make web apps faster. Instead of directly changing the real HTML on the webpage (which is slow), React first creates a virtual copy of the DOM in memory. When something changes, React compares the old virtual DOM with the new one, finds exactly what changed, and then only updates that small part in the real DOM. This process is called "diffing". By avoiding unnecessary changes to the whole webpage, React apps feel faster and more efficient. The render() function in React is used to display your React component on the web page. It tells React exactly what to show and where to show it. For example, in a basic app, you might use ReactDOM.render(<App />, document.getElementById('root')), which means "Take the App component and place it inside the HTML element with the id 'root'." Every time your component changes (like when the state updates), React will call this render method again to re-display the latest version of the UI.

Course Outcomes : CO4: Implement Web applications and Web Services

(Conclusion to be based on objectives and outcomes achieved)
From this experiment I learned how to design a simple web page using React JS by understanding its core concepts like components, props, state, and the virtual DOM. I successfully set up a React development environment using Node.js and created a basic React application. By modifying the App.js file, I explored how React updates the UI dynamically through its render process. This experiment helped me achieve the objective of implementing a web application interface, aligning with the course outcome CO4.

Grade: AA/AB/BB/BC/CC/CD/DD/FF

Signature of faculty in-charge with date

References: Books/ Journals/ Websites:

1. React – A JavaScript library for building user interfaces (reactjs.org)
2. "React - A JavaScript library for building user interfaces". React. Retrieved 7 April 2018.
3. Krill, Paul (May 15, 2014). "React: Making faster, smoother UIs for data-driven Web apps". InfoWorld.
4. Hemel, Zef (June 3, 2013). "Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews". InfoQ.
5. Dawson, Chris (July 25, 2014). "JavaScript's History and How it Led To ReactJS". The New Stack.
6. Dere, Mohan (2018-02-19). "How to integrate create-react-app with all the libraries