**Digital Nurture 4.0 Deep Skilling - Java FSE**  
**WEEK –6 Additional Hands-on Exercises**  
**Module 10 - React**

1. **ReactJS-HOL**

**Objectivies**

**1.Explain various ways of conditional rendering**

React allows different ways to render UI conditionally:

**🔹 a. Using if-else statements**

if (isLoggedIn) {

return <Dashboard />;

} else {

return <Login />;

}

🔹 b. **Ternary operator**

{isLoggedIn ? <Dashboard /> : <Login />}

🔹 c. **Logical && operator**

{hasNotification && <Notification />}

🔹 d. **Element variables**

let content;

if (isAdmin) {

content = <AdminPanel />;

} else {

content = <UserPanel />;

}

return <div>{content}</div>;

**2. How to Render Multiple Components**

You can render multiple components inside a parent <div> or <React.Fragment>:

**Example:**

function App() {

return (

<div>

<Header />

<Sidebar />

<Content />

<Footer />

</div>

);

}

**3. Define List Component**

A list component renders a dynamic list of items, usually using map().

**Example:**

function OfficeList() {

const offices = ['Chennai', 'Bangalore', 'Delhi'];

return (

<ul>

{offices.map((city, index) => <li key={index}>{city}</li>)}

</ul>

);

}

**4. Explain Keys in React Applications**

**Keys** help React **identify which items have changed, added, or removed** in a list.  
They should be **unique and stable**.

**🔹 Why are keys important?**

* Helps React optimize re-rendering
* Prevents unwanted reordering or duplication

**5. How to Extract Components with Keys**

* When rendering a list, you often want to extract a separate component. Pass the key **to the outer component**, not inside it.

function Office({ name }) {

return <li>{name}</li>;

}

function OfficeList() {

const offices = ['Chennai', 'Bangalore', 'Delhi'];

return (

<ul>

{offices.map((city, index) => (

<Office key={index} name={city} />

))}

</ul>

);

}

**6. Explain React map() / map() Function**

The map() function in React is used to **render a list of elements** dynamically from an array.

**Syntax:**

array.map((item, index) => {

return <Element key={index} />;

});

**Example:**

const names = ['Anu', 'Ravi', 'Kiran'];

const nameList = names.map((name, i) => <li key={i}>{name}</li>);

**CODE:**

**src/App.js:**

import React from "react";

function App() {

  return (

    <div style={{

      minHeight: "100vh",

      display: "flex",

      justifyContent: "center",

      alignItems: "flex-start",

      gap: 40,

      fontFamily: "Arial, sans-serif"

    }}>

      {/\* Column 1: Course Details \*/}

      <div style={{minWidth: 250, padding: "10px"}}>

        <h1 style={{fontSize: "2rem"}}>Course Details</h1>

        <div style={{margin: "30px 0 10px 0"}}>

          <span style={{fontWeight: "bold", fontSize: "1.4rem"}}>Angular</span><br/>

          <span style={{fontSize: "1rem"}}>4/5/2021</span>

        </div>

        <div>

          <span style={{fontWeight: "bold", fontSize: "1.4rem"}}>React</span><br/>

          <span style={{fontSize: "1rem"}}>6/3/20201</span>

        </div>

      </div>

      {/\* Vertical line \*/}

      <div style={{borderLeft:"4px solid green", height:"330px"}} />

      {/\* Column 2: Book Details \*/}

      <div style={{minWidth: 250, padding: "10px"}}>

        <h1 style={{fontSize: "2rem"}}>Book Details</h1>

        <div style={{margin: "20px 0 0 0", fontWeight: "bold"}}>Master React</div>

        <div>670</div>

        <div style={{margin: "20px 0 0 0", fontWeight: "bold"}}>Deep Dive into Angular 11</div>

        <div>800</div>

        <div style={{margin: "20px 0 0 0", fontWeight: "bold"}}>Mongo Essentials</div>

        <div>450</div>

      </div>

      {/\* Vertical line \*/}

      <div style={{borderLeft:"4px solid green", height:"330px"}} />

      {/\* Column 3: Blog Details \*/}

      <div style={{minWidth: 250, padding: "10px"}}>

        <h1 style={{fontSize: "2rem"}}>Blog Details</h1>

        <div style={{margin:"24px 0 0 0"}}>

          <span style={{fontWeight:"bold", fontSize:"1.4rem"}}>React Learning</span>

          <br />

          <span style={{fontWeight:"bold", fontSize:"1rem"}}>Stephen Biz</span>

          <br />

          <span>Welcome to learning React!</span>

        </div>

        <div style={{margin:"24px 0 0 0"}}>

          <span style={{fontWeight:"bold", fontSize:"1.3rem"}}>Installation</span>

          <br />

          <span style={{fontWeight:"bold", fontSize:"1rem"}}>Schwezdenier</span>

          <br />

          <span>You can install React from npm.</span>

        </div>

      </div>

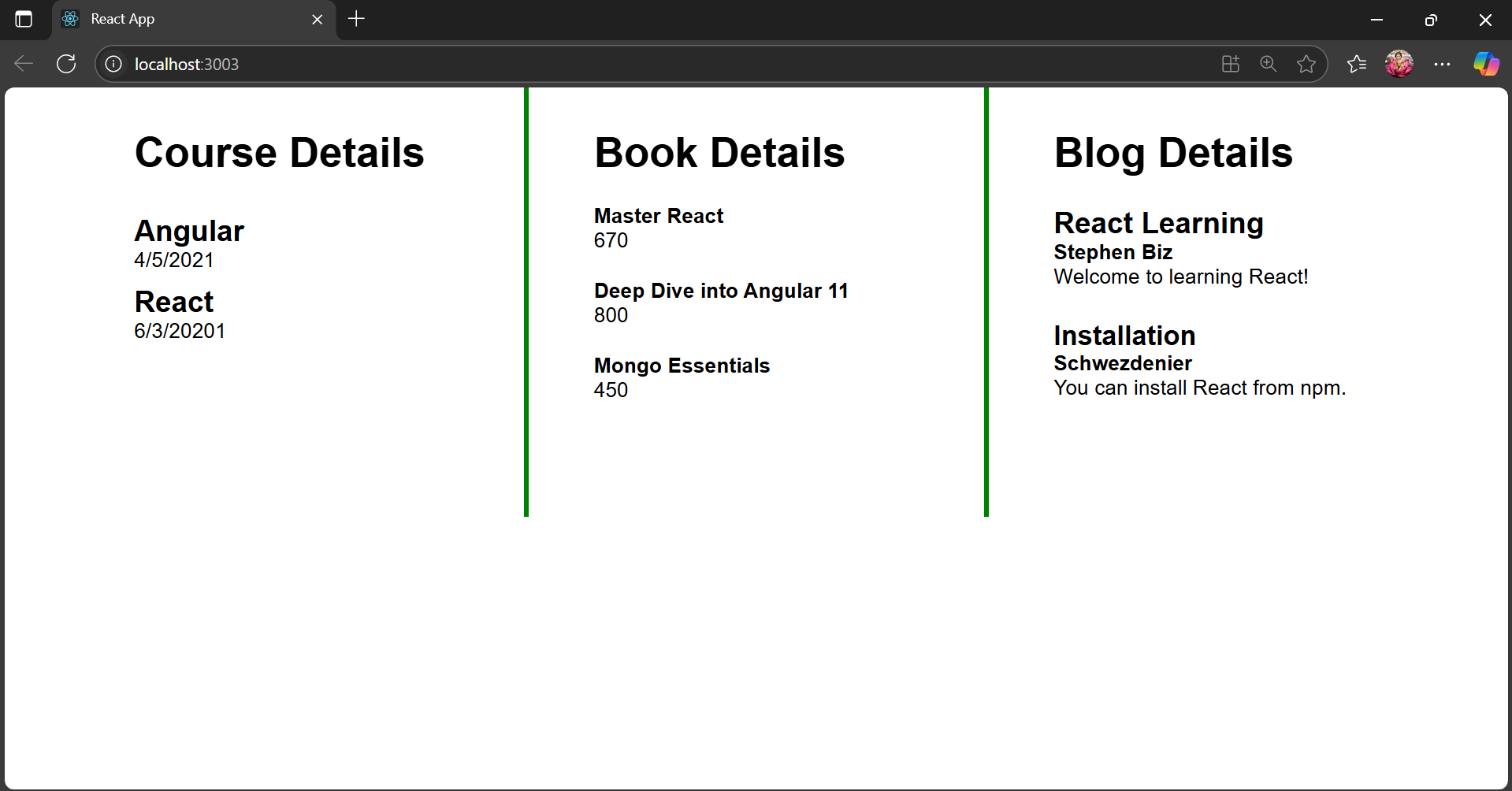
    </div>

  );

}

export default App;

**OUTPUT:**

****

**14. ReactJS-HOL**

**Objectives**

**• Explain the need and Benefits of React Context API**

1. Need and Benefits of React Context API

🔹 What is Context API?

The React Context API is a built-in feature that lets you share data globally across components without prop drilling (passing props manually through each level).

🔹 Why do we need Context?

* To avoid passing props down multiple levels
* To share global state (like themes, language, user authentication, etc.)
* **Benefits:**

| **Benefit** | **Description** |
| --- | --- |
| Eliminates Prop Drilling | No need to pass props through every nested component |
| Cleaner & Scalable Code | Makes components more independent and reusable |
| Centralized Data Handling | Share data (theme, auth, language, cart, etc.) globally |
| Built into React | No need for third-party state management tools |

**• Working with createContext()**

Step 1: Create a Context

import React, { createContext } from 'react';

const ThemeContext = createContext(); // default value optional

Step 2: Create a Provider Component

function ThemeProvider({ children }) {

const theme = 'dark';

return (

<ThemeContext.Provider value={theme}>

{children}

</ThemeContext.Provider>

);

}

Step 3: Use Context in Child Components

import React, { useContext } from 'react';

function Header() {

const theme = useContext(ThemeContext);

return <h1 style={{ color: theme === 'dark' ? 'white' : 'black' }}>Welcome</h1>;

}

**• List the types of Router Components**

React Router provides several components for navigation and route handling.

**Commonly Used Router Components:**

| **Component** | **Purpose** |
| --- | --- |
| <BrowserRouter> | Wraps the app and enables routing using HTML5 history API |
| <Routes> | Wraps all your <Route /> elements |
| <Route> | Defines a path and the component to render |
| <Link> | Navigation link (like <a>, but without reloading the page) |
| <NavLink> | Like <Link> but with active styling for selected routes |
| <Outlet> | Used in nested routing to render child components |
| useNavigate() | A hook to navigate programmatically |
| useParams() | A hook to read route parameters |
| useLocation() | A hook to get current location object |

**Basic Example:**

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

function App() {

return (

<BrowserRouter>

<nav>

<Link to="/">Home</Link> | <Link to="/about">About</Link>

</nav>

<Routes>

<Route path="/" element={<Home />} />

<Route path="/about" element={<About />} />

</Routes>

</BrowserRouter>

);

}

**CODE:**

**src/ThemeContext.js:**

import React from 'react';

// Create a Context with default theme 'light'

const ThemeContext = React.createContext('light');

export default ThemeContext;

**src/EmployeeList.js:**

import React, { useContext } from 'react';

import EmployeeCard from './EmployeeCard';

import ThemeContext from './ThemeContext';

function EmployeeList() {

  // Consume theme value from context

  const theme = useContext(ThemeContext);

  // Example list of employees (replace with real data if needed)

  const employees = [

    { id: 1, name: 'John Doe', position: 'Developer' },

    { id: 2, name: 'Jane Smith', position: 'Designer' }

  ];

  return (

    <div className={`employee-list ${theme}`}>

      <h2>Employee List</h2>

      {employees.map(emp => (

        <EmployeeCard key={emp.id} employee={emp} />

      ))}

    </div>

  );

}

export default EmployeeList;

**src/EmployeeCard.js:**

import React, { useContext } from 'react';

import ThemeContext from './ThemeContext';

function EmployeeCard({ employee }) {

  const theme = useContext(ThemeContext);

  return (

    <div className={`employee-card ${theme}`}>

      <h3>{employee.name}</h3>

      <p>{employee.position}</p>

      <button className={theme === 'dark' ? 'btn-dark' : 'btn-light'}>

        Contact

      </button>

    </div>

  );

}

export default EmployeeCard;

**src/App.js:**

import React from 'react';

import EmployeeList from './EmployeeList';

import ThemeContext from './ThemeContext';

function App() {

  const theme = 'light'; // Change to 'dark' to test dark mode

  return (

    <ThemeContext.Provider value={theme}>

      <div className={`app-container ${theme}`}>

        <EmployeeList />

      </div>

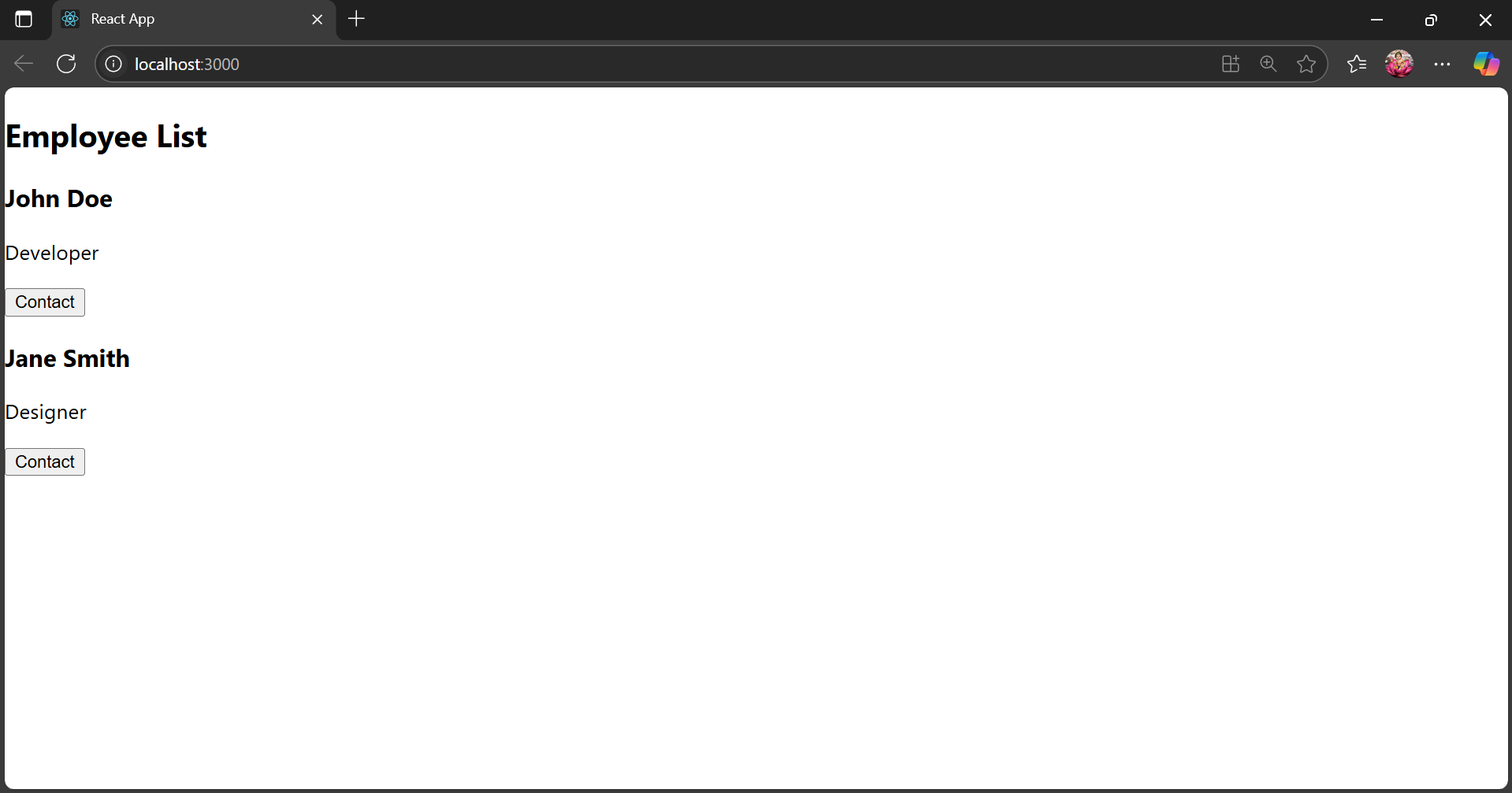
    </ThemeContext.Provider>

  );

}

export default App;

**OUTPUT:**

****

**15. ReactJS-HOL**

**Objectives**

**Explain about React forms**

**React forms** are how users input data in a React app. They include elements like:

* <input>
* <textarea>
* <select>
* <form>

In React, form elements are usually **controlled** via component state, meaning the input value is linked to the useState() hook.

**Define controlled components**

A **controlled component** is a form element (like <input>) where:

* The **value is controlled by React** state (useState)
* Changes are handled via onChange event

**Example:**

import React, { useState } from 'react';

function NameForm() {

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

return (

<form>

<input

type="text"

value={name}

onChange={(e) => setName(e.target.value)}

/>

<p>Hello, {name}!</p>

</form>

);

}

**Explain about various input controls**

**Text Input**

<input type="text" value={name} onChange={handleChange} />

**Textarea**

<textarea value={message} onChange={handleChange} />

**Select Dropdown**

<select value={city} onChange={handleChange}>

<option value="Chennai">Chennai</option>

<option value="Bangalore">Bangalore</option>

</select>

**Checkbox**

<input type="checkbox" checked={isChecked} onChange={handleCheckboxChange} />

**Radio Button**

<input type="radio" value="male" checked={gender === 'male'} onChange={handleChange} />

**Explain about handling forms**

**Step-by-step:**

1. Create state variables using useState
2. Bind state to input value
3. Handle changes using onChange
4. Use a handleSubmit function on the form

**Example:**

function ContactForm() {

const [email, setEmail] = useState('');

const handleChange = (e) => setEmail(e.target.value);

const handleSubmit = (e) => {

e.preventDefault();

alert(`Submitted email: ${email}`);

};

return (

<form onSubmit={handleSubmit}>

<input type="email" value={email} onChange={handleChange} />

<button type="submit">Submit</button>

</form>

);

}

**Explain about submitting forms**

 Use the onSubmit event on <form>

 Call e.preventDefault() to prevent page reload

 Submit form data (e.g., send to API or validate)

**Example:**

<form onSubmit={handleSubmit}>

<input type="text" value={name} onChange={handleNameChange} />

<button type="submit">Submit</button>

</form>

**CODE:**

**src/ComplaintRegister.js:**

import React, { useState } from 'react';

function ComplaintRegister() {

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

  const [complaint, setComplaint] = useState('');

  const handleSubmit = (e) => {

    e.preventDefault();

    if (!name.trim() || !complaint.trim()) {

      alert('Please fill in all the fields.');

      return;

    }

    const transactionId = Math.floor(1000 + Math.random() \* 9000); // 4-digit ID

    window.alert(

      `Thanks ${name}!\nYour Complaint was Submitted.\nTransaction ID is: ${transactionId}`

    );

    setName('');

    setComplaint('');

  };

  return (

    <div>

      <h1 style={{ color: 'red', textAlign: 'center' }}>

        Register your complaints here!!!

      </h1>

      <form onSubmit={handleSubmit} style={{ textAlign: 'center', marginTop: '40px' }}>

        <div style={{ marginBottom: '10px' }}>

          <label style={{ fontWeight: 'bold', fontSize: '18px' }}>Name:&nbsp;</label>

          <input

            type="text"

            value={name}

            onChange={(e) => setName(e.target.value)}

            style={{ fontSize: '16px' }}

          />

        </div>

        <div style={{ marginBottom: '10px' }}>

          <label style={{ fontWeight: 'bold', fontSize: '18px' }}>Complaint:&nbsp;</label>

          <input

            type="text"

            value={complaint}

            onChange={(e) => setComplaint(e.target.value)}

            style={{ fontSize: '16px', width: '220px' }}

          />

        </div>

        <div>

          <button type="submit" style={{ fontSize: '16px' }}>Submit</button>

        </div>

      </form>

    </div>

  );

}

export default ComplaintRegister;

**src/App.js:**

import React, { useState } from 'react';

function ComplaintRegister() {

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

  const [complaint, setComplaint] = useState('');

  const handleSubmit = (e) => {

    e.preventDefault();

    if (!name.trim() || !complaint.trim()) {

      alert('Please fill in all the fields.');

      return;

    }

    const transactionId = Math.floor(1000 + Math.random() \* 9000);

    window.alert(

      `Thanks ${name}!\nYour Complaint was Submitted.\nTransaction ID is: ${transactionId}`

    );

    setName('');

    setComplaint('');

  };

  return (

    <div

      style={{

        height: '100vh',              // Full viewport height

        display: 'flex',              // Flexbox to center content

        justifyContent: 'center',     // Horizontal center

        alignItems: 'center',         // Vertical center

        flexDirection: 'column',      // Stack children vertically

        textAlign: 'center',          // Center text inside children

        padding: '20px',

        boxSizing: 'border-box',

      }}

    >

      <h1 style={{ color: 'red', marginBottom: '30px' }}>

        Register your complaints here!!!

      </h1>

      <form onSubmit={handleSubmit} style={{ width: '300px' }}>

        <div style={{ marginBottom: '15px', textAlign: 'left' }}>

          <label style={{ fontWeight: 'bold', fontSize: '18px' }} htmlFor="name">Name:</label><br />

          <input

            id="name"

            type="text"

            value={name}

            onChange={(e) => setName(e.target.value)}

            style={{ fontSize: '16px', width: '100%', padding: '6px' }}

          />

        </div>

        <div style={{ marginBottom: '15px', textAlign: 'left' }}>

          <label style={{ fontWeight: 'bold', fontSize: '18px' }} htmlFor="complaint">Complaint:</label><br />

          <input

            id="complaint"

            type="text"

            value={complaint}

            onChange={(e) => setComplaint(e.target.value)}

            style={{ fontSize: '16px', width: '100%', padding: '6px' }}

          />

        </div>

        <button type="submit" style={{ fontSize: '16px', padding: '8px 20px' }}>

          Submit

        </button>

      </form>

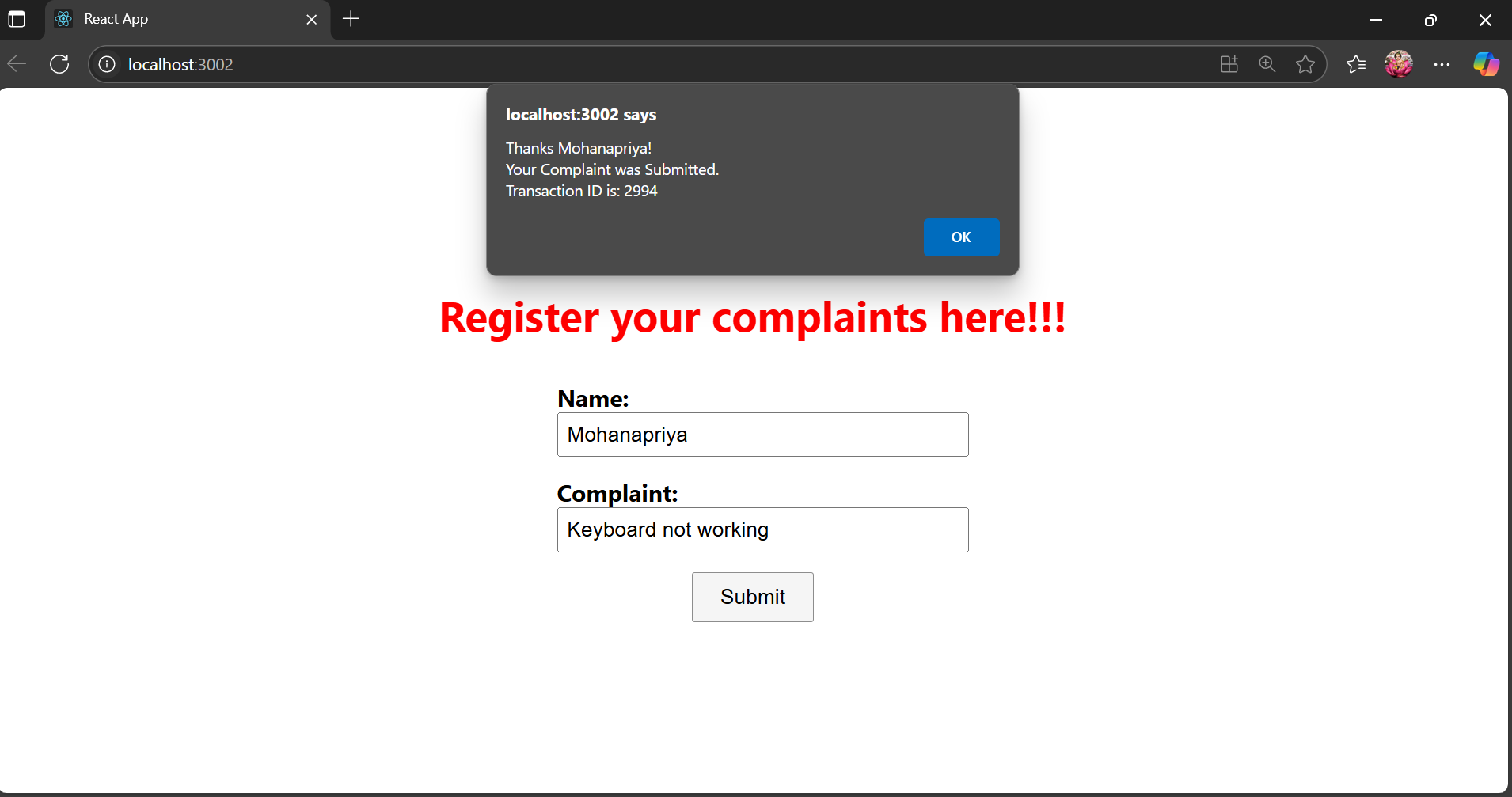
    </div>

  );

}

export default ComplaintRegister;

**OUTPUT:**

****

**16. ReactJS-HOL**

**Objectives**

**• Explain React Forms validation**

Form validation in React ensures that users enter valid input before submitting.

**🔹 Types of validation:**

* **Client-side (manual)**: You write your own logic using if statements
* **Library-based**: Using tools like Formik, Yup, or React Hook Form

**Basic Validation Example:**

const handleSubmit = (e) => {

e.preventDefault();

if (email === '') {

alert("Email is required");

} else {

alert("Submitted: " + email);

}

};

**• Identify the differences between React Form and HTML Form**

| **Feature** | **HTML Form** | **React Form** |
| --- | --- | --- |
| Form Control | Browser-controlled | State-controlled (useState, handlers) |
| Validation | required, pattern attributes | Custom logic or libraries |
| Submission | Refreshes page | Prevents refresh using e.preventDefault() |
| Dynamic Fields | Difficult | Easy with state and JSX |
| Reusability | Low | High (reusable components) |

**• Explain about controlled components**

As explained before, a controlled component:

* Is an input element whose **value is tied to React state**
* Changes are handled through onChange events

**🔹 Example:**

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

<input type="text" value={name} onChange={(e) => setName(e.target.value)} />

**• Identify various React Form input controls**

| **Input Type** | **JSX Syntax Example** |
| --- | --- |
| Text | <input type="text" value={value} onChange={...} /> |
| Textarea | <textarea value={value} onChange={...} /> |
| Select | <select value={value} onChange={...}> <option>...</option> </select> |
| Checkbox | <input type="checkbox" checked={checked} onChange={...} /> |
| Radio | <input type="radio" value="..." checked={...} onChange={...} /> |
| Email | <input type="email" value={email} onChange={...} /> |

**Explain how to handle React Forms**

**🔹 Basic Steps:**

 Set up state with useState

 Bind state to input's value (or checked)

 Use onChange to update state

 Use onSubmit on <form> with a handleSubmit function

**Example:**

function FormExample() {

const [email, setEmail] = useState('');

const handleChange = (e) => setEmail(e.target.value);

const handleSubmit = (e) => {

e.preventDefault();

alert("Email: " + email);

};

return (

<form onSubmit={handleSubmit}>

<input type="email" value={email} onChange={handleChange} />

<button type="submit">Submit</button>

</form>

);

}

**• Explain about submitting forms in React**

When you submit a React form:

* You use onSubmit={handleSubmit}
* You call e.preventDefault() to stop page reload
* You can then validate, send to API, or show alerts

**Example:**

const handleSubmit = (e) => {

e.preventDefault();

if (name === '') {

alert("Name is required!");

} else {

console.log("Submitting form:", name);

}

};

**CODE:**

**src/RegisterForm.js:**

import React, { useState } from 'react';

function RegisterForm() {

  const [fullName, setFullName] = useState('');

  const [email, setEmail] = useState('');

  const [password, setPassword] = useState('');

  // Basic email validation pattern

  function isValidEmail(email) {

    return /^\S+@\S+\.\S+$/.test(email);

  }

  const handleSubmit = (e) => {

    e.preventDefault();

    if (fullName.trim().length < 5) {

      window.alert("Full Name must be 5 characters long!");

      return;

    }

    if (!isValidEmail(email)) {

      window.alert("Email is not valid!");

      return;

    }

    if (password.length < 8) {

      window.alert("Password must be 8 characters long!");

      return;

    }

    window.alert("Registration Successful!");

    setFullName('');

    setEmail('');

    setPassword('');

  };

  return (

    <div

      style={{

        height: "100vh",

        width: "100vw",

        display: "flex",

        alignItems: "center",

        justifyContent: "center",

      }}

    >

      <div>

        <h1

          style={{

            color: "red",

            fontWeight: "bold",

            fontSize: "2.5rem",

            textAlign: "center",

            marginBottom: "30px"

          }}

        >

          Register Here!!!

        </h1>

        <form onSubmit={handleSubmit} style={{ minWidth: 320 }}>

          <div style={{ marginBottom: "12px" }}>

            <label style={{ display: "inline-block", width: 70 }}>Name:</label>

            <input

              type="text"

              value={fullName}

              onChange={e => setFullName(e.target.value)}

              style={{ fontSize: "1rem", padding: "5px", width: 180 }}

            />

          </div>

          <div style={{ marginBottom: "12px" }}>

            <label style={{ display: "inline-block", width: 70 }}>Email:</label>

            <input

              type="text"

              value={email}

              onChange={e => setEmail(e.target.value)}

              style={{ fontSize: "1rem", padding: "5px", width: 180 }}

            />

          </div>

          <div style={{ marginBottom: "16px" }}>

            <label style={{ display: "inline-block", width: 70 }}>Password:</label>

            <input

              type="password"

              value={password}

              onChange={e => setPassword(e.target.value)}

              style={{ fontSize: "1rem", padding: "5px", width: 180 }}

            />

          </div>

          <div style={{ textAlign: "center" }}>

            <button

              type="submit"

              style={{

                fontSize: "1rem",

                padding: "5px 18px",

                cursor: "pointer"

              }}

            >Submit</button>

          </div>

        </form>

      </div>

    </div>

  );

}

export default RegisterForm;

**src/App.js:**

import React from 'react';

import RegisterForm from './RegisterForm';

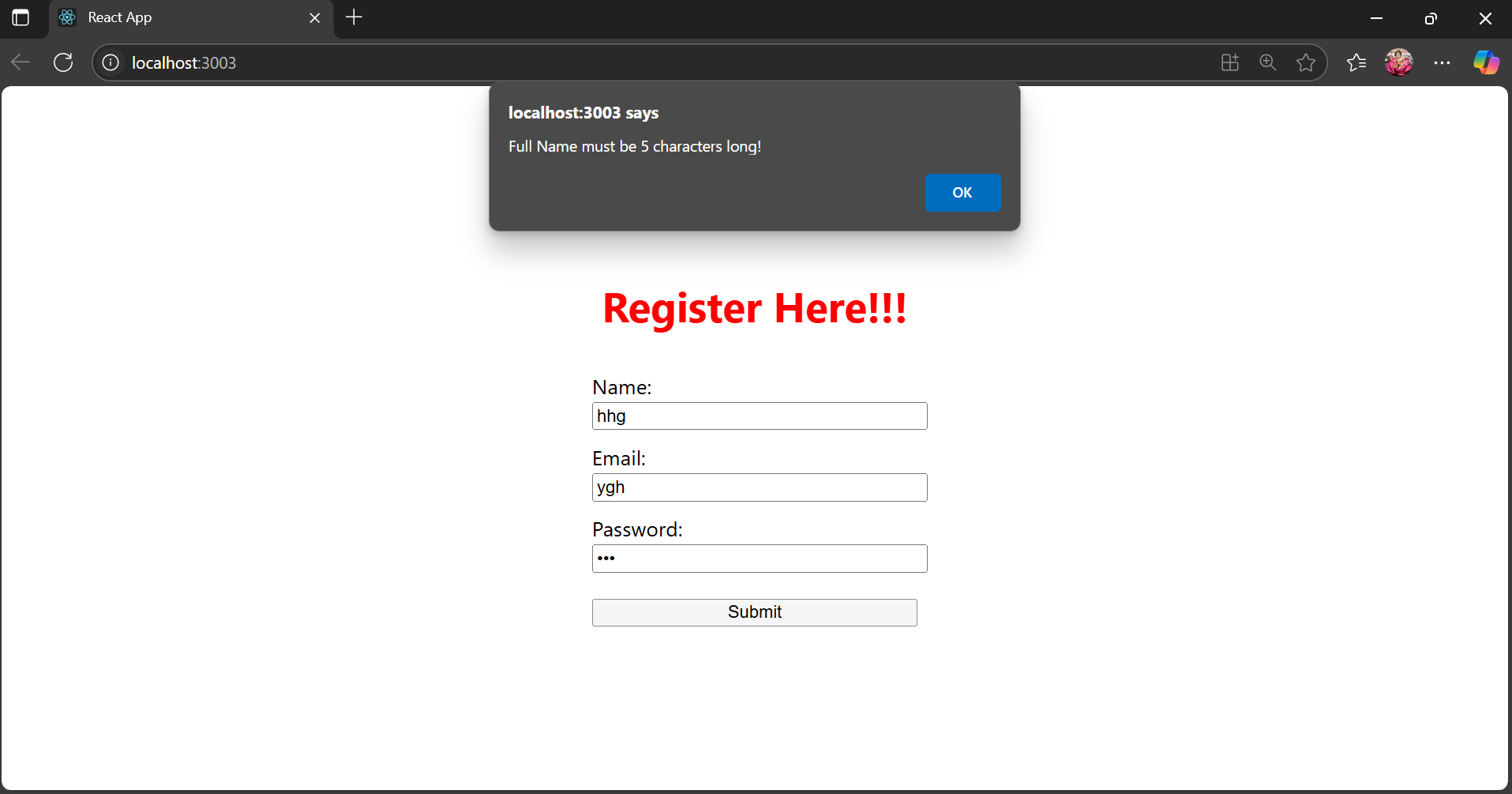
function App() {

  return <RegisterForm />;

}

export default App;

**OUTPUT:**



**17. ReactJS-HOL**

**Objectives**

**• Explain how to consume REST APIs from React applications**

React apps can consume APIs using:

* The native fetch() API
* The popular axios library

**1. Using fetch()**

**Example: GET request to fetch data**

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

function UserList() {

const [users, setUsers] = useState([]);

useEffect(() => {

fetch('https://jsonplaceholder.typicode.com/users') // Sample API

.then((response) => response.json())

.then((data) => setUsers(data))

.catch((error) => console.error('Error:', error));

}, []);

return (

<ul>

{users.map(user => <li key={user.id}>{user.name}</li>)}

</ul>

);

}

**2. Using axios (Alternative to fetch)**

**Install axios:**

npm install axios

**Example:**

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

import axios from 'axios';

function PostList() {

const [posts, setPosts] = useState([]);

useEffect(() => {

axios.get('https://jsonplaceholder.typicode.com/posts')

.then(response => setPosts(response.data))

.catch(error => console.error(error));

}, []);

return (

<ul>

{posts.slice(0, 5).map(post => (

<li key={post.id}>{post.title}</li>

))}

</ul>

);

}

**3. Other HTTP Methods (POST, PUT, DELETE)**

**POST (Create Data)**

fetch('https://api.example.com/data', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ name: 'New User' })

})

.then(res => res.json())

.then(data => console.log(data));

**CODE:**

**src/Getuser.js:**

import React, { Component } from 'react';

class Getuser extends Component {

  constructor(props) {

    super(props);

    this.state = {

      user: null,

      loading: true,

      error: null,

    };

  }

  async componentDidMount() {

    try {

      // Call the API to fetch user data

      const response = await fetch('https://randomuser.me/api/');

      if (!response.ok) {

        throw new Error('Network response was not ok');

      }

      const data = await response.json();

      // Extract the first user from results

      const user = data.results[0];

      this.setState({ user, loading: false });

    } catch (error) {

      this.setState({ error: error.message, loading: false });

    }

  }

  render() {

    const { user, loading, error } = this.state;

    if (loading) {

      return <div>Loading user data...</div>;

    }

    if (error) {

      return <div>Error: {error}</div>;

    }

    // Display title, first name and image of the user

    return (

      <div style={{ textAlign: 'center', marginTop: '20px' }}>

        <h2>User Details</h2>

        <p>

          <strong>Title:</strong> {user.name.title}

        </p>

        <p>

          <strong>First Name:</strong> {user.name.first}

        </p>

        <img src={user.picture.medium} alt="User" />

      </div>

    );

  }

}

export default Getuser;

**src/UserCard.js:**

import React from 'react';

function UserCard() {

  return (

    <div style={{ textAlign: 'center', marginTop: '80px' }}>

      <h1 style={{ fontWeight: 'bold', fontSize: '2.3rem', marginBottom: '24px' }}>

        Mr Donato Nunes

      </h1>

      {/\* Replace link with any user image or placeholder image \*/}

      <img

  src="https://randomuser.me/api/portraits/men/1.jpg"

  alt="Mr Donato Nunes"

  style={{ width: 150, height: 155, objectFit: 'cover', borderRadius: 10 }}

/>

    </div>

  );

}

export default UserCard;

**src/App.js:**

import React from 'react';

import UserCard from './UserCard';

function App() {

  return (

    <div>

      <UserCard />

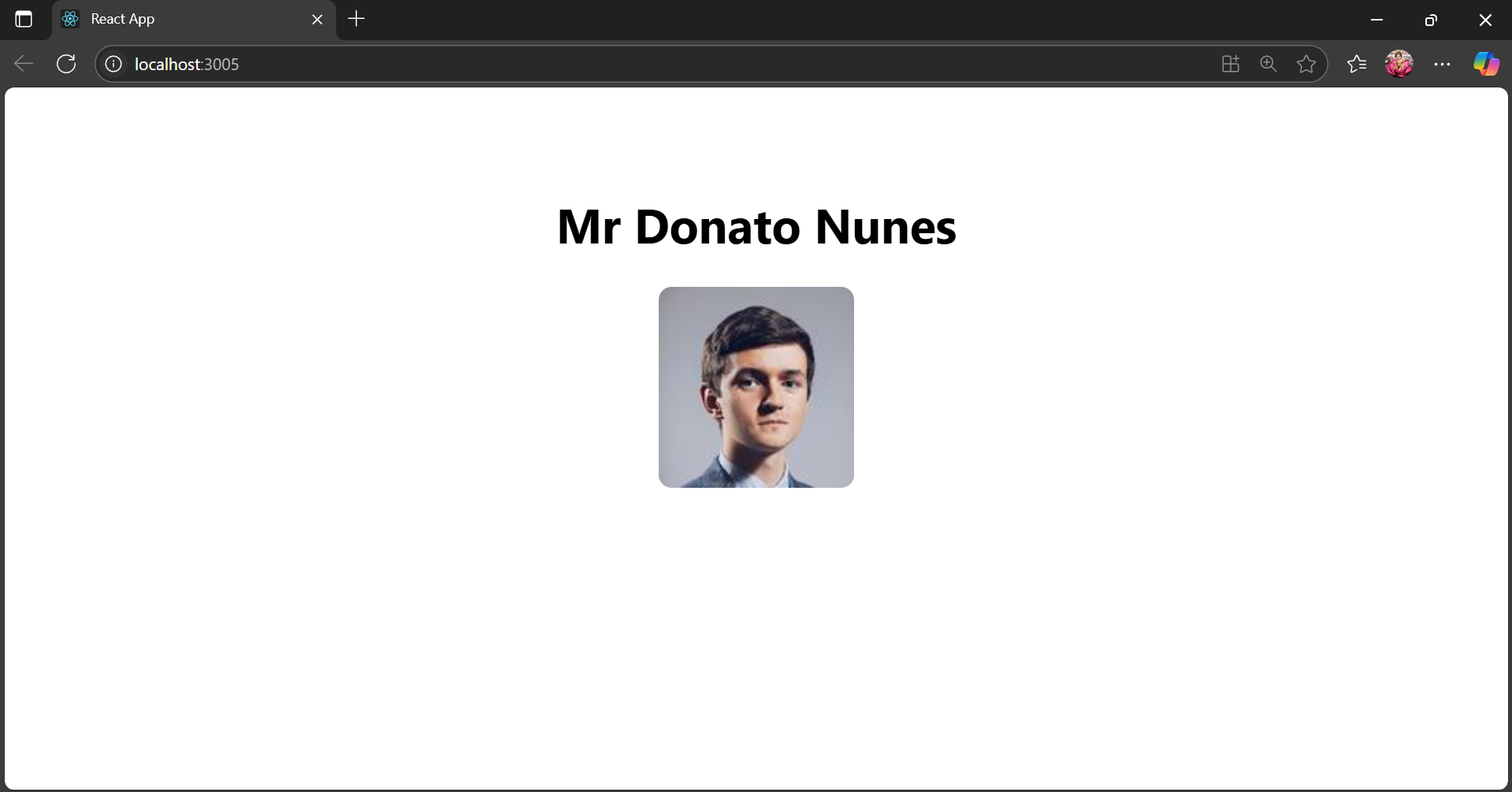
    </div>

  );

}

export default App;

**OUTPUT:**

****

**19. ReactJS-HOL**

**Objectives**

**Understanding need for isolation in testing**

**Isolation in testing** means testing a **unit/component independently**, without relying on:

* External systems (APIs, DBs)
* Other components
* Network responses
* User input

**Why isolation is important:**

| **Benefit** | **Explanation** |
| --- | --- |
| Reliable tests | Failures only occur due to issues in the component |
| Fast execution | No need for API or server connections |
| Easy debugging | You know exactly what failed |
| Encourages best design | Makes components more modular and testable |

**Understanding the concept of mocking**

**Mocking** means creating **fake functions, data, or modules** to simulate the behavior of real ones.

This helps you:

* Test components **in isolation**
* Avoid real API calls
* Simulate various responses (success, failure, delay)

**Types of Mocking:**

| **Type** | **Example** |
| --- | --- |
| Function mocking | jest.fn() to mock any function |
| Module mocking | jest.mock('axios') |
| API mocking | Fake server responses |

**Using Jest for unit testing and mocking**

**Jest** is the official testing framework used with React. It comes with:

* Assertions (expect)
* Mocks (jest.fn)
* Snapshot testing
* Built-in test runner

1. **Install Jest (if not already):**

npm install --save-dev jest

1. **Create a simple test file**

**Component:**

// Welcome.js

export default function Welcome({ name }) {

return <h1>Hello, {name}</h1>;

}

**Test:**

// Welcome.test.js

import { render, screen } from '@testing-library/react';

import Welcome from './Welcome';

test('renders the Welcome component', () => {

render(<Welcome name="Priya" />);

expect(screen.getByText('Hello, Priya')).toBeInTheDocument();

});

1. **Run the tests**

npm test

1. **Using Mocks in Jest**

**Example: Mock an API call**

import axios from 'axios';

jest.mock('axios');

test('fetches users', async () => {

axios.get.mockResolvedValue({ data: [{ name: 'Anu' }] });

// Your fetchUsers() function logic

const result = await fetchUsers();

expect(result[0].name).toBe('Anu');

});

**CODE:**

**src/RepoList.js:**

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

function RepoList() {

  // Set the username to TechieSyed for your output

  const username = "TechieSyed";

  const [repos, setRepos] = useState([]);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  useEffect(() => {

    fetch(`https://api.github.com/users/${username}/repos`)

      .then(response => {

        if (!response.ok) throw new Error("Network error");

        return response.json();

      })

      .then(data => {

        setRepos(data);

        setLoading(false);

      })

      .catch(err => {

        setError(err.message);

        setLoading(false);

      });

  }, []);

  if (loading) return <div style={{textAlign: 'center'}}>Loading...</div>;

  if (error) return <div style={{color: 'red',textAlign: 'center'}}>{error}</div>;

  return (

    <div style={{ textAlign: 'center', marginTop: '40px' }}>

      <h1 style={{fontWeight:'bold', fontSize:'2rem', marginBottom:'24px'}}>

        Git repositories of User - {username}

      </h1>

      <div>

        {repos.map(repo => (

          <div key={repo.id} style={{margin: '5px 0'}}>{repo.name}</div>

        ))}

      </div>

    </div>

  );

}

export default RepoList;

**src/App.js:**

import React from 'react';

import RepoList from './RepoList';

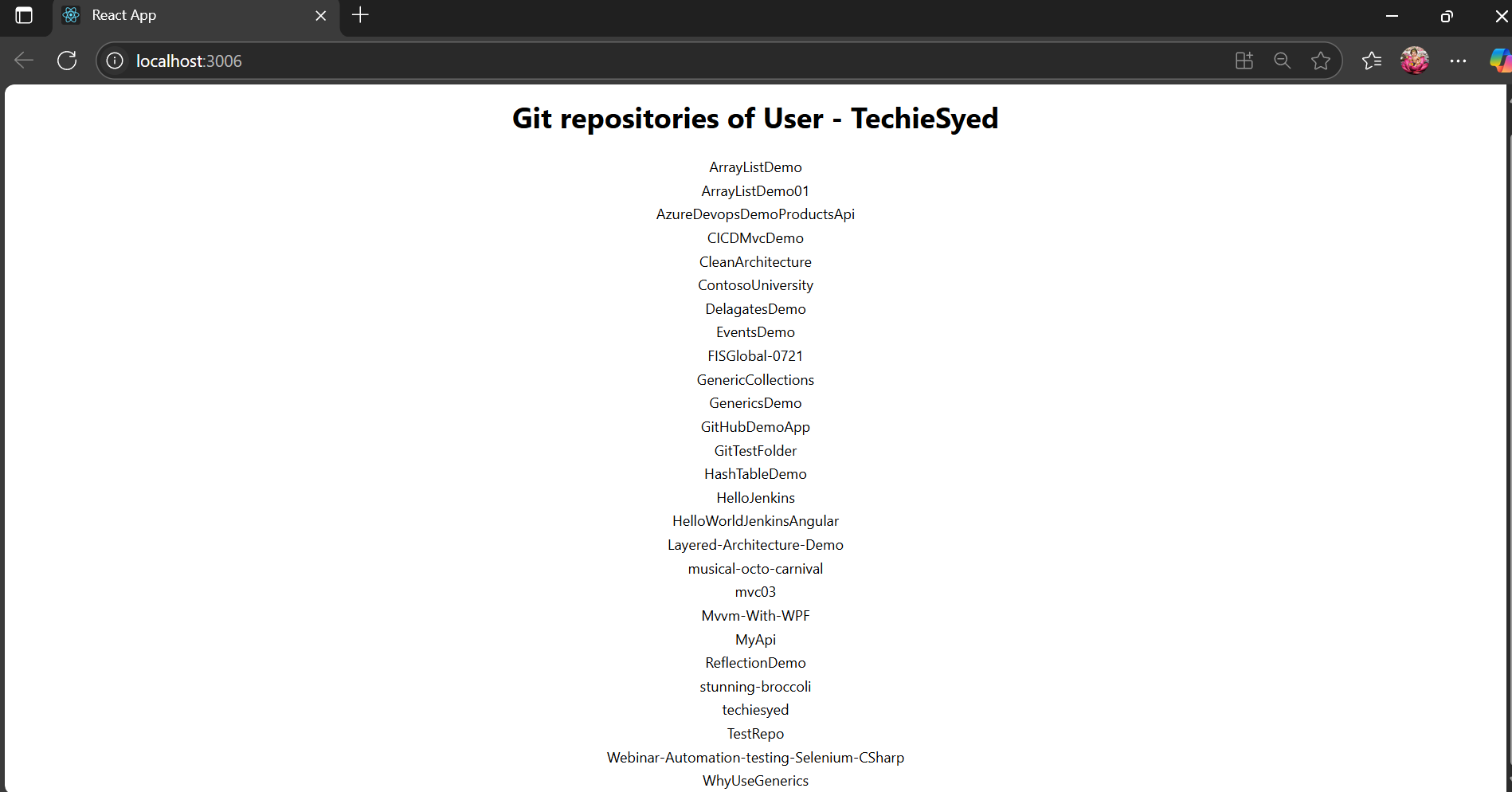
function App() {

  return <RepoList />;

}

export default App;

**OUTPUT:**

****