**9.ReactJS\_HOL**

**1. Features of ES6:**

* let and const for block-scoped variables.
* Arrow functions () => {} for concise function expressions.
* Classes and class inheritance (class, extends).
* Template literals (Hello ${name}).
* Destructuring assignment.
* Spread/rest operators (...).
* Modules (import / export).
* Promises and async/await.
* Enhanced object literals.
* Set and Map data structures.

**2. JavaScript let:**

* Declares block-scoped variables.
* Cannot be redeclared in the same scope.
* Allows reassignment.

**3. Difference between var and let:**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Redeclaration | Allowed | Not allowed in same scope |
| Hoisting | Yes (undefined) | Yes (TDZ – Temporal Dead Zone) |

**4. JavaScript const:**

* Block-scoped.
* Must be initialized at declaration.
* Cannot be reassigned, but internal object/array values can change.

**5. ES6 Class Fundamentals:**

class Player {

constructor(name, score) {

this.name = name;

this.score = score;

}

}

**6. ES6 Class Inheritance:**

class Batsman extends Player {

constructor(name, score, strikeRate) {

super(name, score);

this.strikeRate = strikeRate;

}

}

**7. ES6 Arrow Functions:**

const add = (a, b) => a + b;

* No this binding.
* Concise syntax.

**8. Identify set() and map():**

const mySet = new Set([1, 2, 3]); // No duplicates

const myMap = new Map([['a', 1], ['b', 2]]);

**App.js**

import React from 'react';

import ListofPlayers from './components/ListofPlayers';

import Scorebelow70 from './components/Scorebelow70';

import { OddPlayers } from './components/OddPlayers';

import { EvenPlayers } from './components/EvenPlayers';

import ListofIndianPlayers from './components/ListofIndianPlayers';

const players = [

{ name: 'Jack', score: 50 },

{ name: 'Michael', score: 70 },

{ name: 'John', score: 40 },

{ name: 'Ann', score: 61 },

{ name: 'Elisabeth', score: 61 },

{ name: 'Sachin', score: 95 },

{ name: 'Dhoni', score: 100 },

{ name: 'Virat', score: 84 },

{ name: 'Jadeja', score: 64 },

{ name: 'Raina', score: 75 },

{ name: 'Rohit', score: 80 }

];

const T20Players = ['First Player', 'Second Player', 'Third Player'];

const RanjiTrophyPlayers = ['Fourth Player', 'Fifth Player', 'Sixth Player'];

export const IndianPlayers = [...T20Players, ...RanjiTrophyPlayers];

const IndianTeam = ['Sachin1', 'Dhoni2', 'Virat3', 'Rohit4', 'Yuvaraj5', 'Raina6'];

const flag = true;

function App() {

if (flag === true) {

return (

<div>

<h1>List of Players</h1>

<ListofPlayers players={players} />

<hr />

<h1>List of Players having Scores Less than 70</h1>

<Scorebelow70 players={players} />

</div>

);

} else {

return (

<div>

<div>

<h1>Indian Team</h1>

<h1>Odd Players</h1>

{OddPlayers(IndianTeam)}

<hr />

<h1>Even Players</h1>

{EvenPlayers(IndianTeam)}

</div>

<hr />

<div>

<h1>List of Indian Players Merged:</h1>

<ListofIndianPlayers IndianPlayers={IndianPlayers} />

</div>

</div>

);

}

}

export default App;

**ListofPlayers.jsx**

import React from 'react';

export default function ListofPlayers({ players }) {

return (

<div>

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

<div key={index}>

<li>Mr. {item.name} <span>{item.score}</span></li>

</div>

))}

</div>

);

}

**Scorebelow70.jsx**

import React from 'react';

export default function Scorebelow70({ players }) {

const players70 = [];

players.map((item) => {

if (item.score <= 70) {

players70.push(item);

}

});

return (

<div>

{players70.map((item, index) => (

<div key={index}>

<li>Mr. {item.name} <span>{item.score}</span></li>

</div>

))}

</div>

);

}

**Scorebelow70.jsx**

import React from 'react';

export default function Scorebelow70({ players }) {

const players70 = [];

players.map((item) => {

if (item.score <= 70) {

players70.push(item);

}

});

return (

<div>

{players70.map((item, index) => (

<div key={index}>

<li>Mr. {item.name} <span>{item.score}</span></li>

</div>

))}

</div>

);

}

**OddPlayers.jsx**

import React from 'react';

export function OddPlayers([first, , third, , fifth]) {

return (

<div>

<li>First : {first}</li>

<li>Third : {third}</li>

<li>Fifth : {fifth}</li>

</div>

);

}

**EvenPlayers.jsx**

import React from 'react';

export function EvenPlayers([, second, , fourth, , sixth]) {

return (

<div>

<li>Second : {second}</li>

<li>Fourth : {fourth}</li>

<li>Sixth : {sixth}</li>

</div>

);

}

**ListofIndianPlayers.jsx**

import React from 'react';

export default function ListofIndianPlayers({ IndianPlayers }) {

return (

<div>

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

<div key={index}>

<li>Mr. {item}</li>

</div>

))}

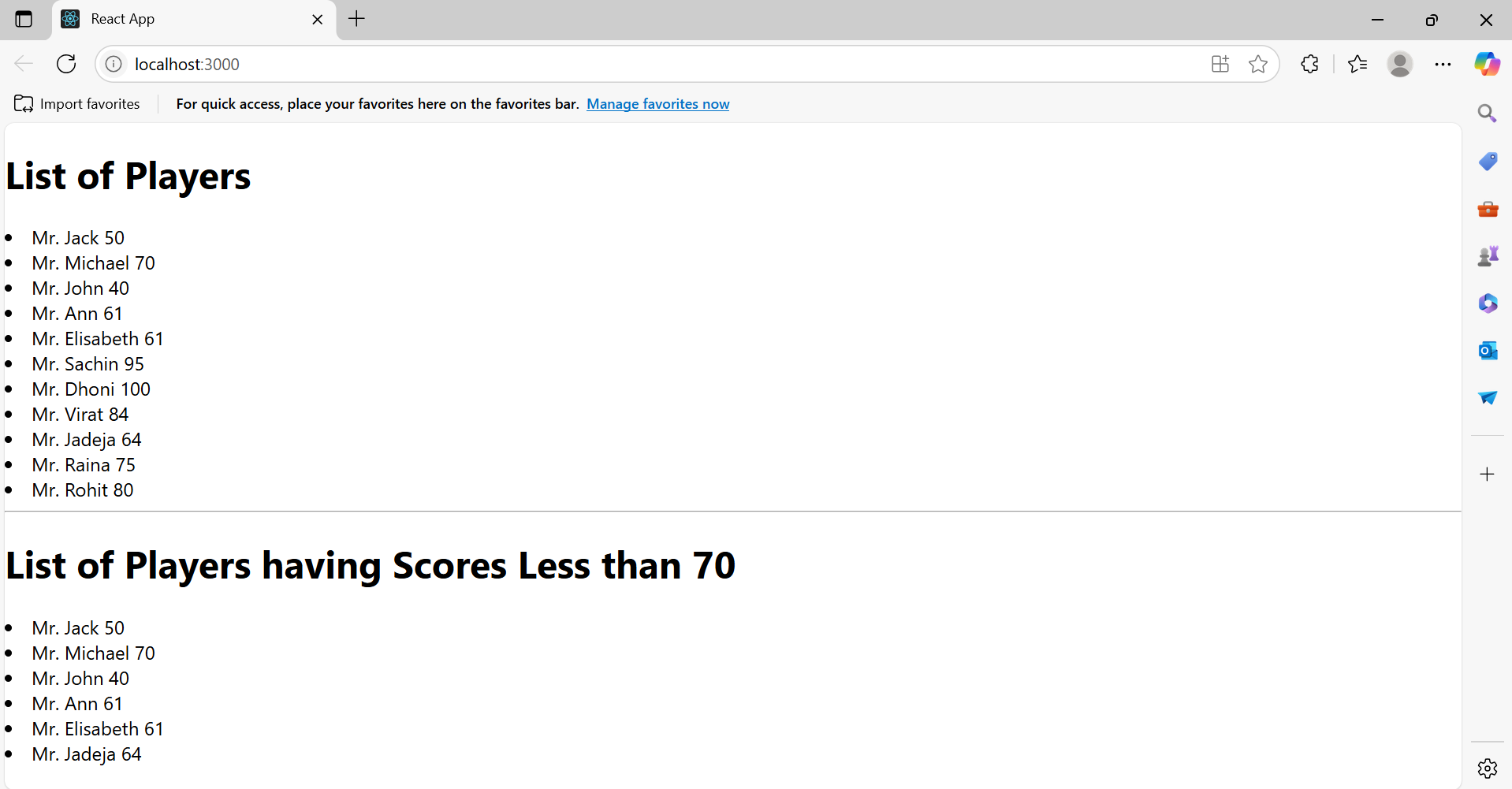
</div>

);

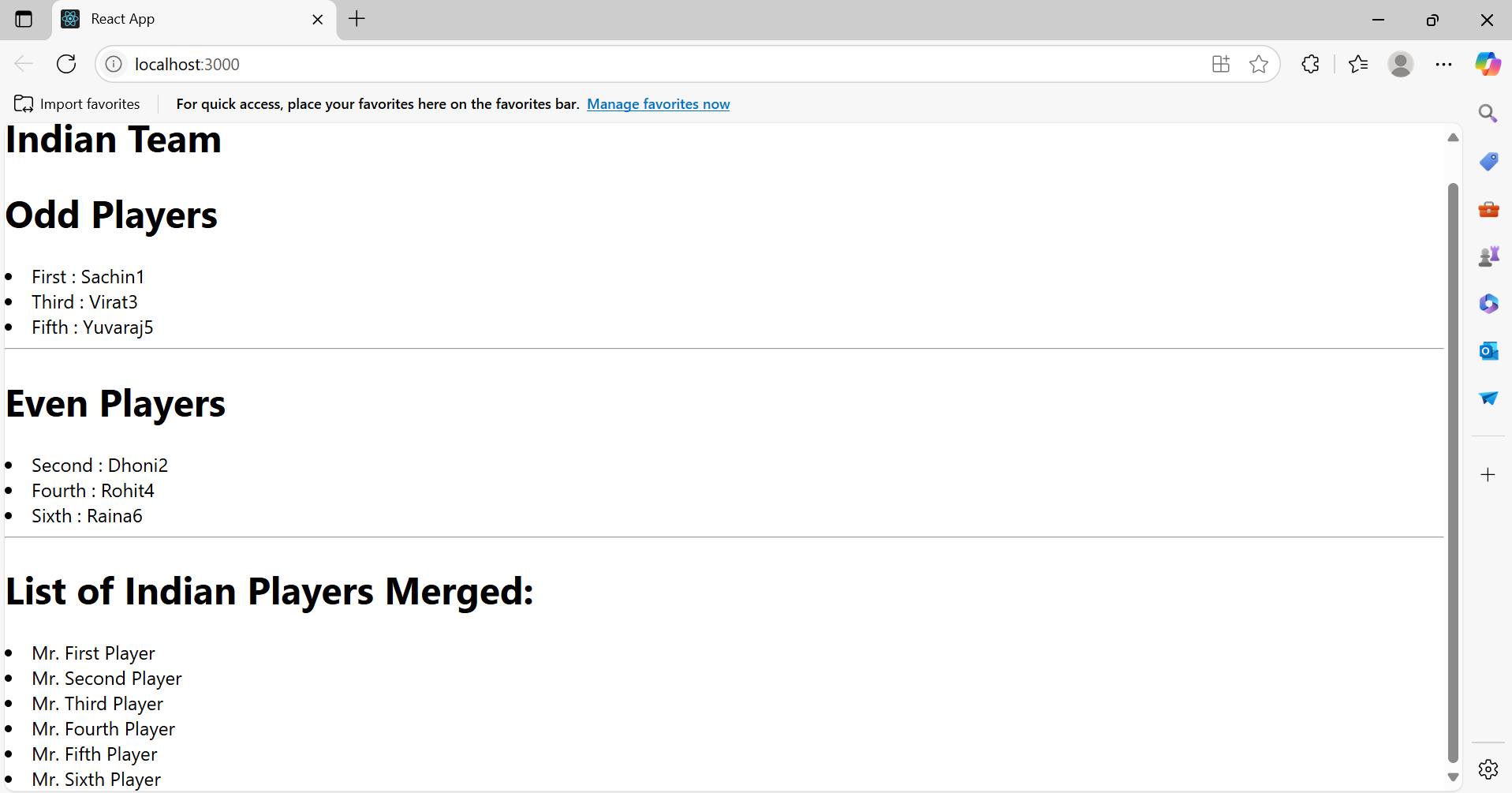
}

**Output:**

**Flag==true**

****

**Flag==false**

****

**10.ReactJS\_HOL**

1. **Define JSX:**JSX (JavaScript XML) is a syntax extension for JavaScript that looks similar to HTML and is used in React to describe UI structure. JSX allows you to write HTML elements directly in JavaScript.
2. **Explain ECMA Script:**ECMAScript (ES) is the standard scripting language specification on which JavaScript is based. ES6 (ECMAScript 2015) introduced major features like let, const, arrow functions, classes, and modules.
3. **Explain React.createElement():**This is a core method provided by React to create React elements. It takes three arguments: the type of the element, props (attributes), and children.

React.createElement('h1', {className: 'title'}, 'Hello World');

1. **Explain how to create React nodes with JSX:**React nodes can be created with JSX using HTML-like syntax inside a component:

const element = <h1>Hello, World!</h1>;

1. **Define how to render JSX to DOM:**You use ReactDOM.render() to render JSX into a DOM node:

ReactDOM.render(<App />, document.getElementById('root'));

1. **Explain how to use JavaScript expressions in JSX:**You can embed JS expressions using {}:

const name = 'Office Space';

const element = <h1>Welcome to {name}</h1>;

1. **Explain how to use inline CSS in JSX:**Inline styles in JSX are written as objects:

const style = { color: 'red', fontSize: '20px' };

<p style={style}>Hello</p>

**App.js**

import React from 'react';

function App() {

const offices = [

{

name: 'DBS',

rent: 50000,

address: 'Chennai',

image: 'https://images.unsplash.com/photo-1598228723793-52759bba239c?auto=format&fit=crop&w=800&q=80'

},

{

name: 'WeWork',

rent: 75000,

address: 'Bangalore',

image: 'https://images.unsplash.com/photo-1586201375761-83865001e31b?auto=format&fit=crop&w=800&q=80'

}

];

return (

<div style={{ padding: '30px', fontFamily: 'Arial' }}>

<h1 style={{ fontWeight: 'bold' }}>Office Space , at Affordable Range</h1>

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

<div key={index} style={{ marginBottom: '40px' }}>

<img

src={office.image}

alt={`Office at ${office.address}`}

width="300"

style={{ margin: '20px 0', borderRadius: '10px' }}

/>

<h2>Name: {office.name}</h2>

<p style={{ color: office.rent < 60000 ? 'red' : 'green', fontWeight: 'bold' }}>

Rent: Rs. {office.rent}

</p>

<p><strong>Address:</strong> {office.address}</p>

</div>

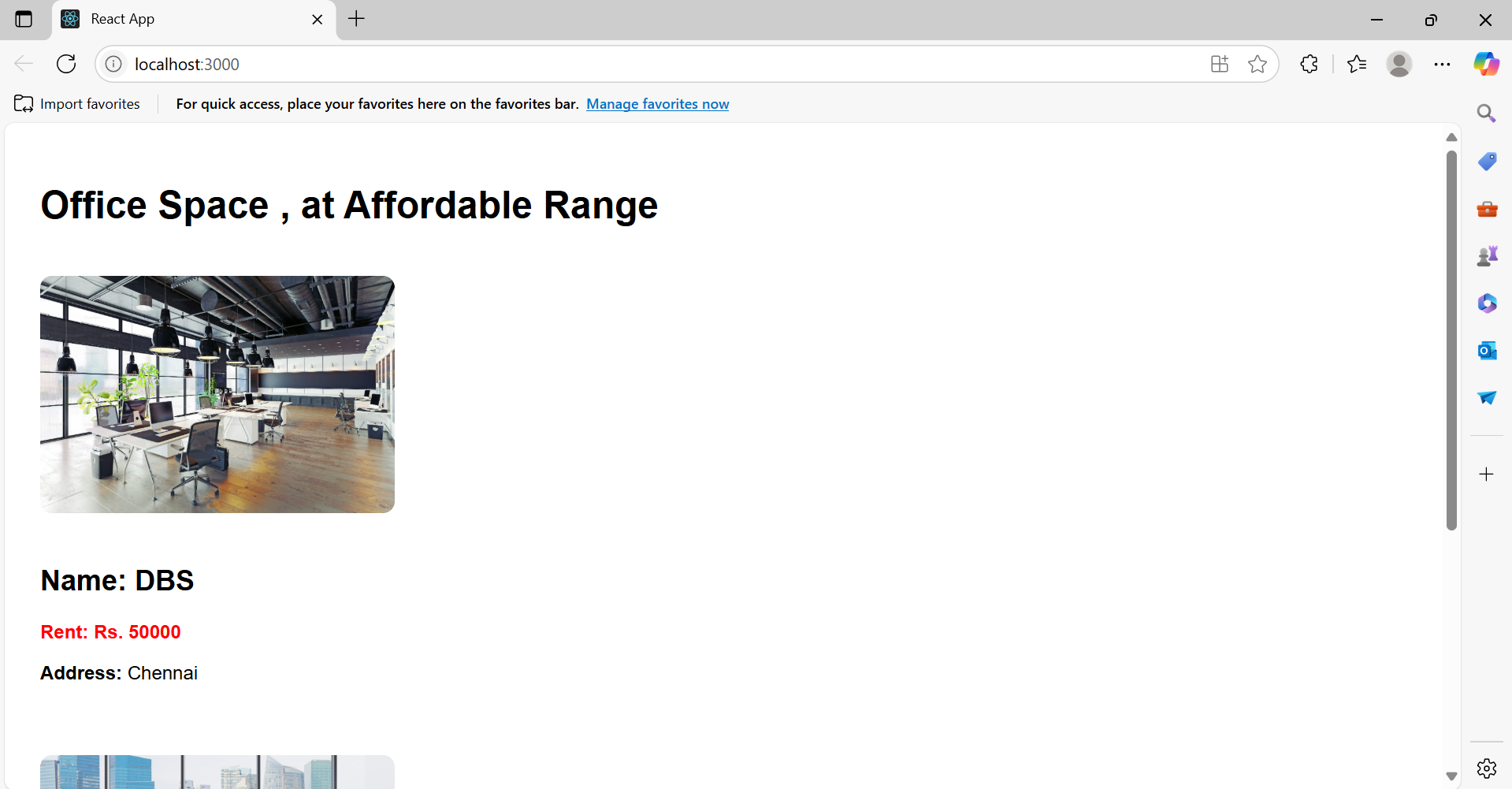
))}

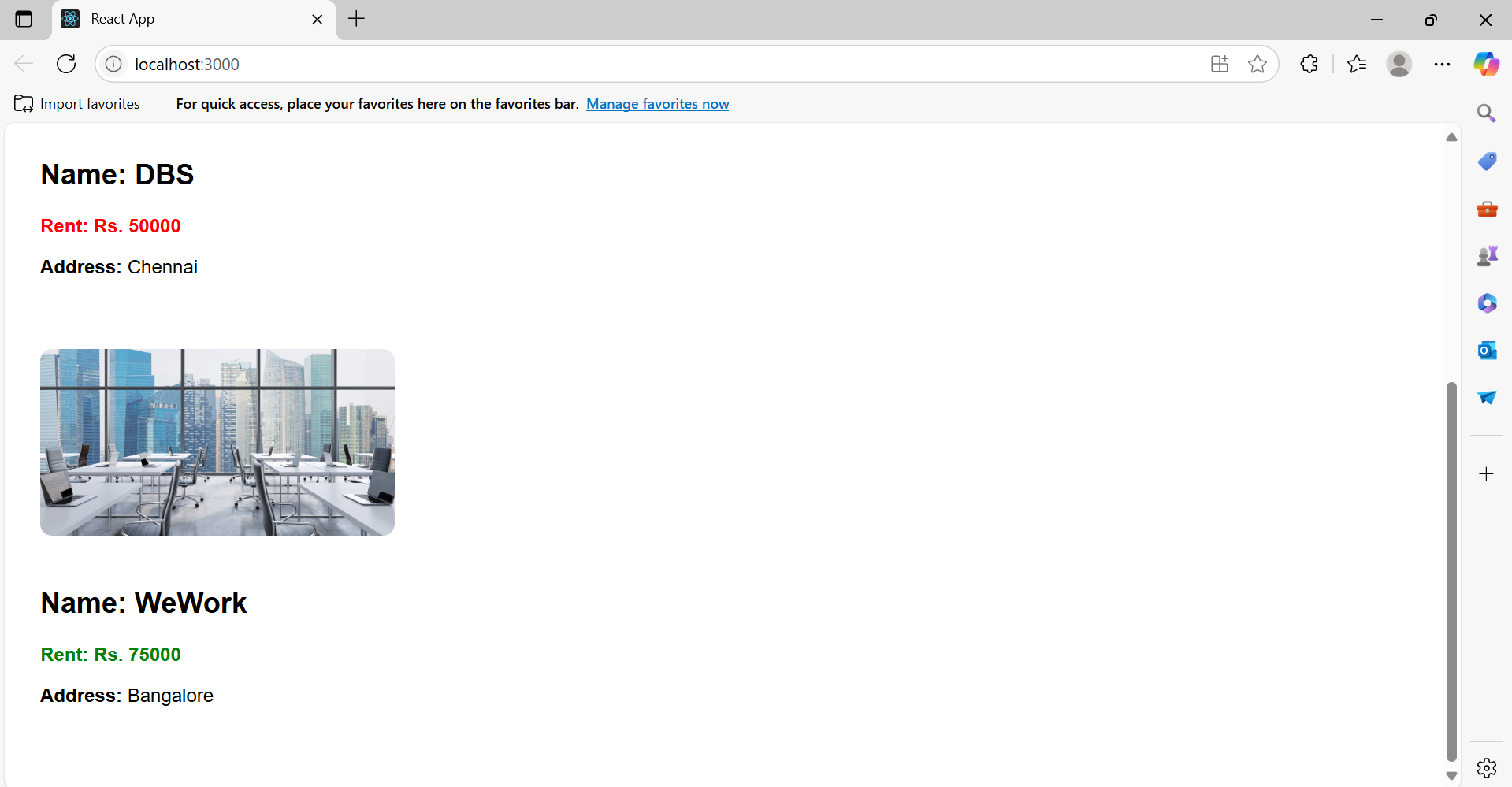
</div>

);

}

export default App;

**Output:  
**

****

**11.ReactJS\_HOL**

**1. React Events**

React events are similar to DOM events but follow a different naming convention and work across all browsers. Examples include onClick, onChange, onSubmit, etc.

**2. Event Handlers**

Event handlers are functions that are triggered in response to an event. In React, they are passed as props to elements like:

<button onClick={handleClick}>Click</button>

**3. Synthetic Event**

React wraps native events in a cross-browser wrapper called SyntheticEvent. It normalizes the event object to work consistently across different browsers.

**4. React Event Naming Convention**

* React uses camelCase instead of lowercase:
  + onClick, onChange
  + onclick, onchange

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**App.js**

import React from 'react';

import Counter from './Counter';

import WelcomeButton from './WelcomeButton';

import SyntheticEventExample from './SyntheticEventExample';

import CurrencyConvertor from './CurrencyConvertor';

function App() {

return (

<div style={{ padding: '20px' }}>

<h1>React Event Examples</h1>

<Counter />

<hr />

<WelcomeButton />

<hr />

<SyntheticEventExample />

<hr />

<CurrencyConvertor />

</div>

);

}

export default App;

**Counter.js**

import React, { Component } from 'react';

class Counter extends Component {

constructor(props) {

super(props);

this.state = { count: 0 };

this.increment = this.increment.bind(this);

this.sayHello = this.sayHello.bind(this);

}

increment() {

this.setState(

(prevState) => ({ count: prevState.count + 1 }),

() => {

this.sayHello();

}

);

}

decrement = () => {

this.setState({ count: this.state.count - 1 });

};

sayHello() {

alert(`Hello Member ${this.state.count}`);

}

render() {

return (

<div>

<h2>Counter: {this.state.count}</h2>

<button onClick={this.increment}>Increment</button>{' '}

<button onClick={this.decrement}>Decrement</button>

</div>

);

}

}

export default Counter;

**WelcomeButton.js**

import React from 'react';

function sayMessage(message) {

alert(message);

}

function WelcomeButton() {

return (

<div>

<button onClick={() => sayMessage("Welcome!")}>Say Welcome</button>

</div>

);

}

export default WelcomeButton;

**SyntheticEventExample.js**

import React from 'react';

function SyntheticEventExample() {

function handleClick(e) {

alert("I was clicked");

console.log(e);

}

return (

<button onClick={handleClick}>Click Me</button>

);

}

export default SyntheticEventExample;

**CurrencyConvertor.js**

import React, { useState } from 'react';

function CurrencyConvertor() {

const [rupees, setRupees] = useState('');

const [currency, setCurrency] = useState('EUR');

const [converted, setConverted] = useState('');

const conversionRates = {

EUR: 90,

USD: 83,

GBP: 105,

};

const currencySymbols = {

EUR: '€',

USD: '$',

GBP: '£',

};

const handleSubmit = () => {

if (!rupees || isNaN(rupees)) {

alert("Please enter a valid amount");

return;

}

const rate = conversionRates[currency];

const symbol = currencySymbols[currency];

const result = (parseFloat(rupees) / rate).toFixed(2);

setConverted(`${symbol}${result}`);

alert(`Converting to ${currency}. Amount is: ${symbol}${result}`);

};

return (

<div>

<h3>Currency Converter</h3>

<input

type="number"

placeholder="Amount in ₹"

value={rupees}

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

/>{' '}

<select value={currency} onChange={(e) => setCurrency(e.target.value)}>

<option value="EUR">Euro (€)</option>

<option value="USD">US Dollar ($)</option>

<option value="GBP">British Pound (£)</option>

</select>{' '}

<button onClick={handleSubmit}>Convert</button>

{converted && <p>Converted amount: {converted}</p>}

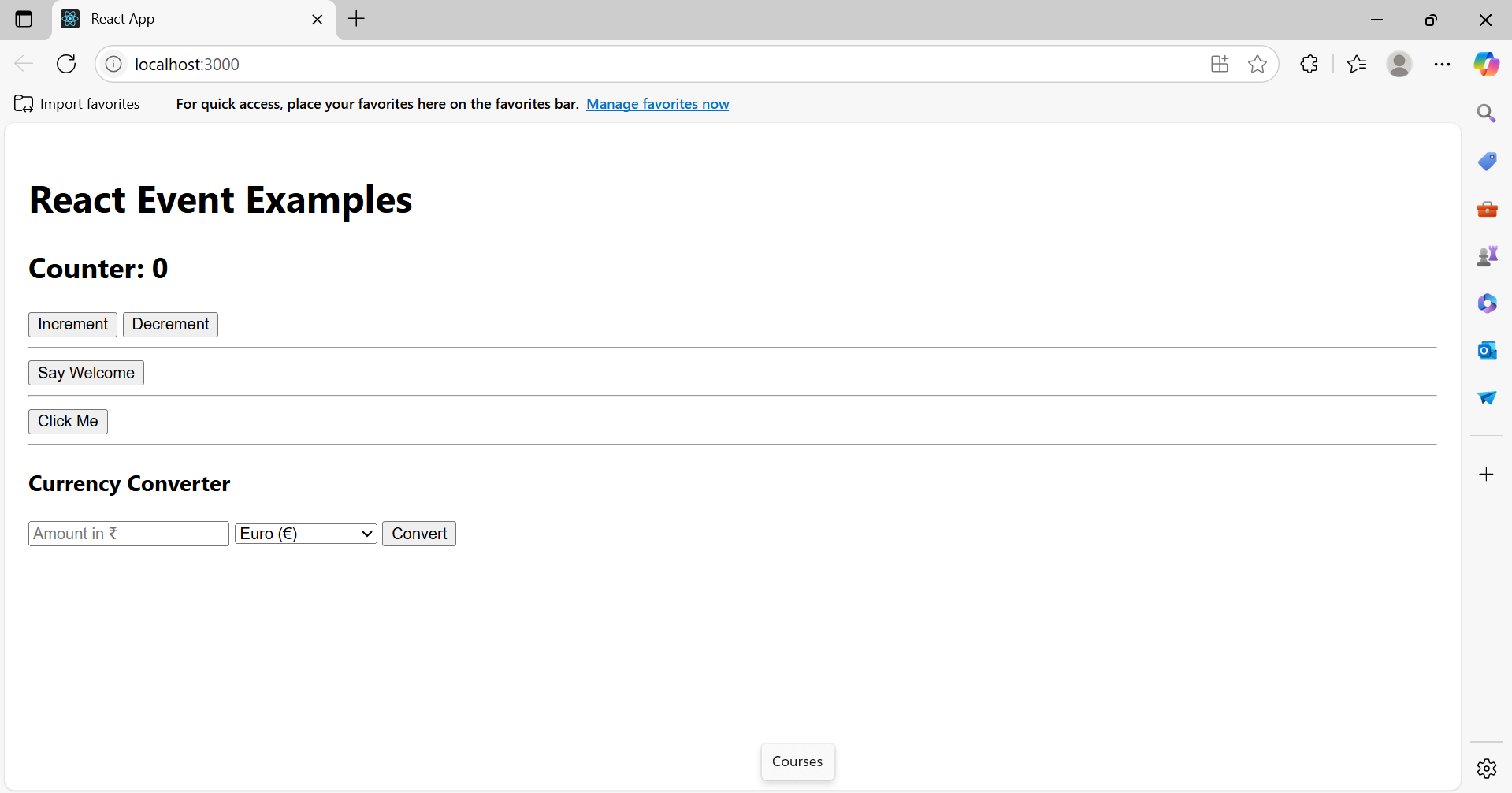
</div>

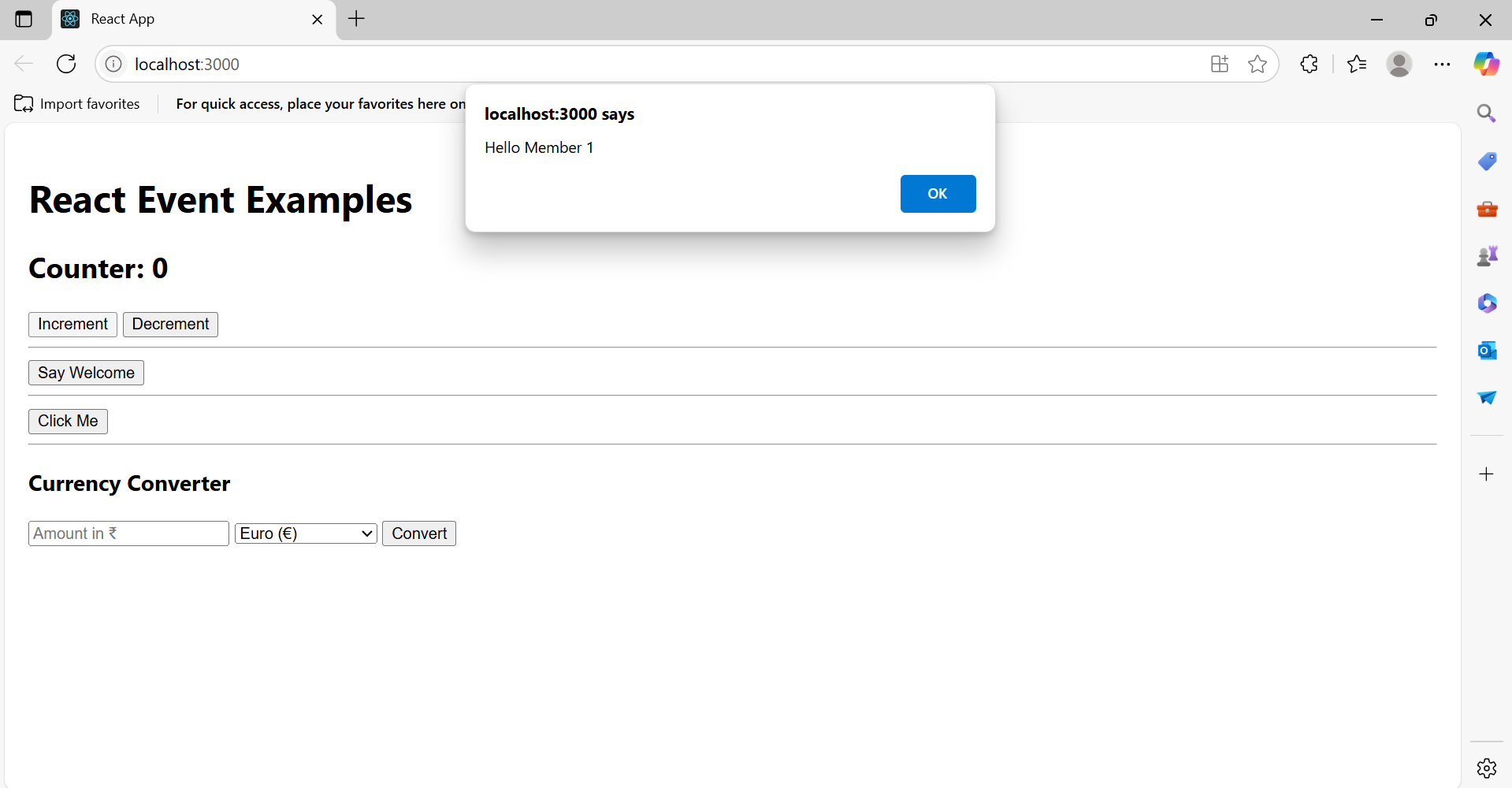
);

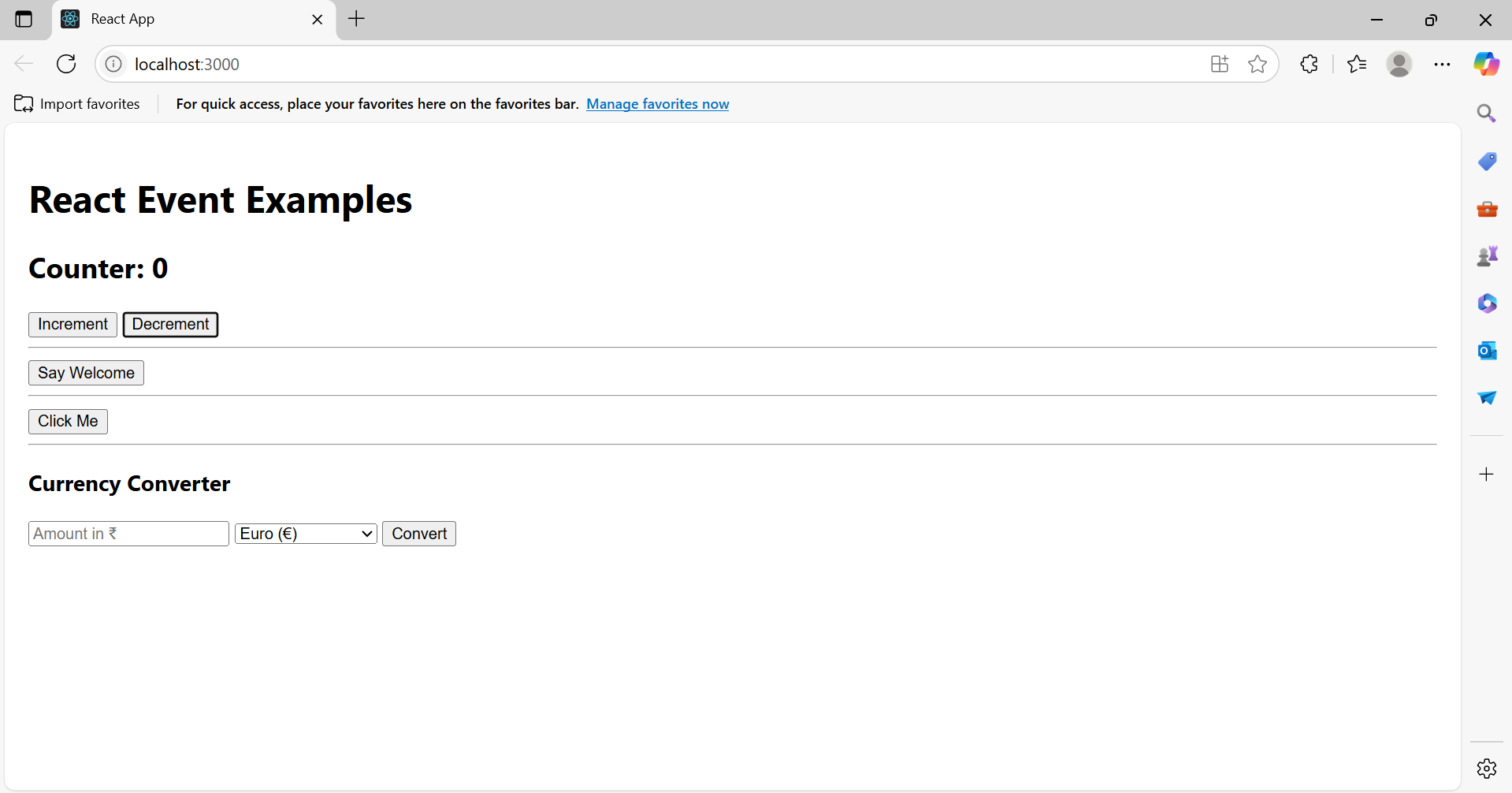
}

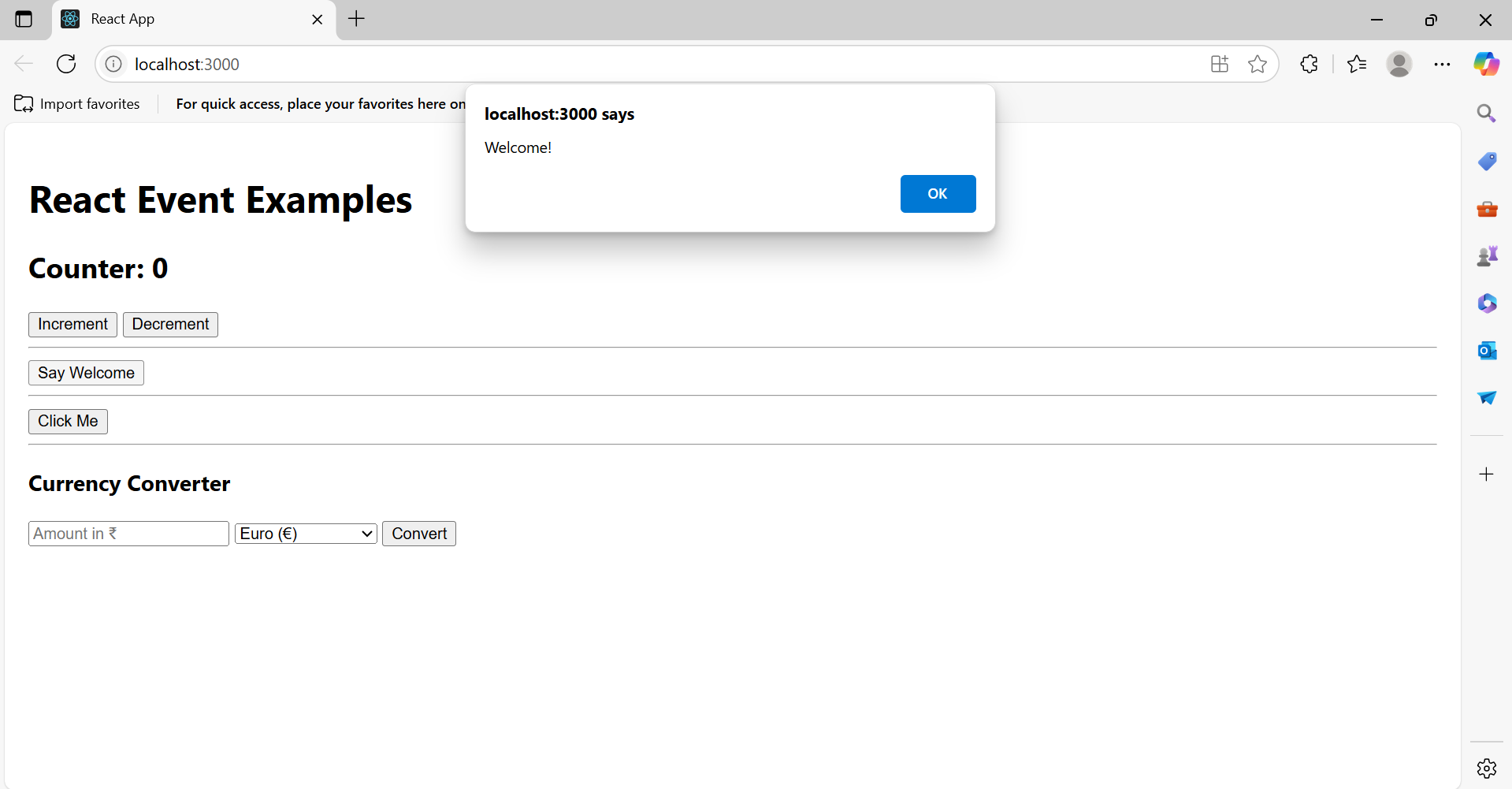
export default CurrencyConvertor;

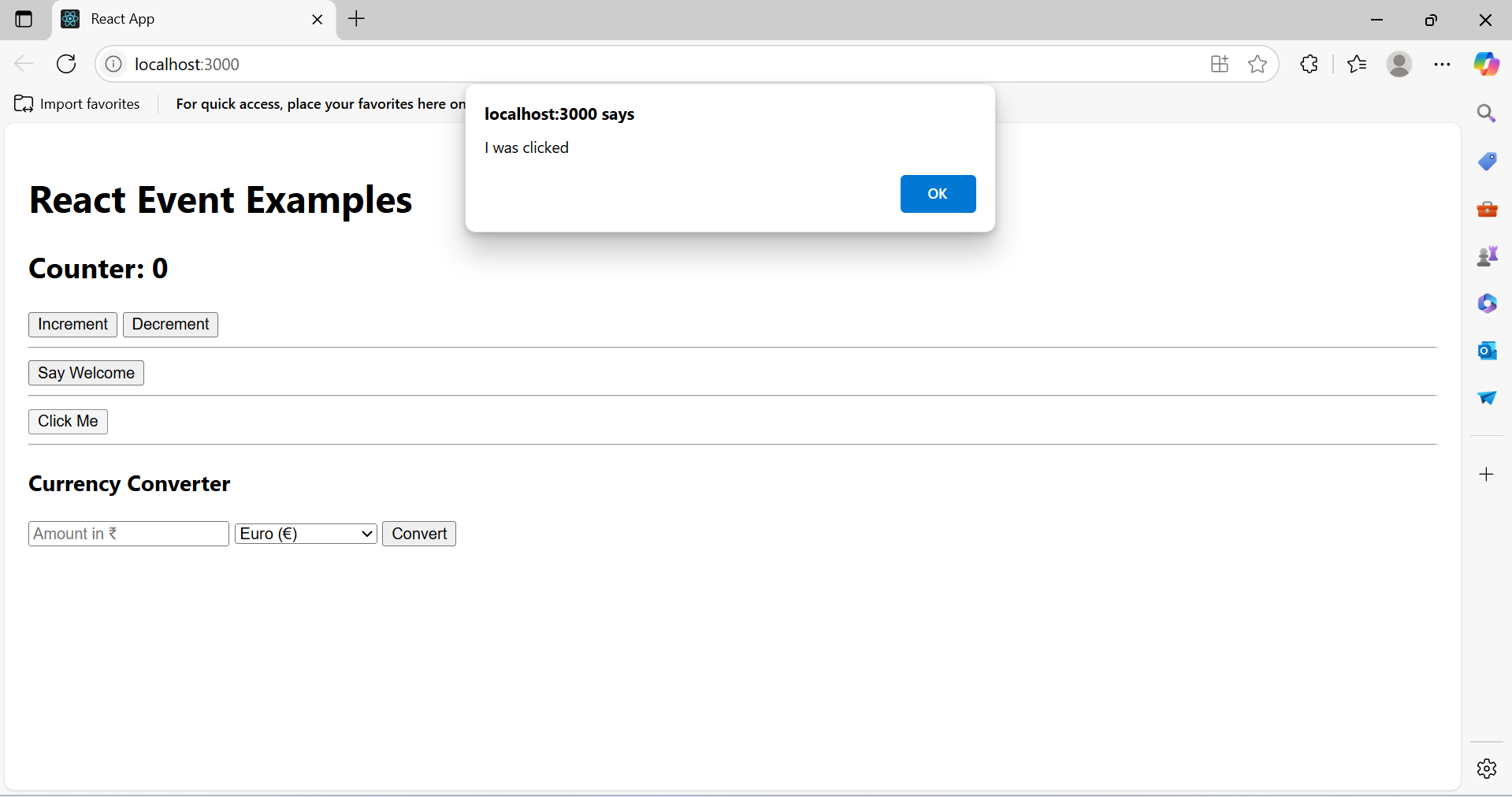
**Output:**

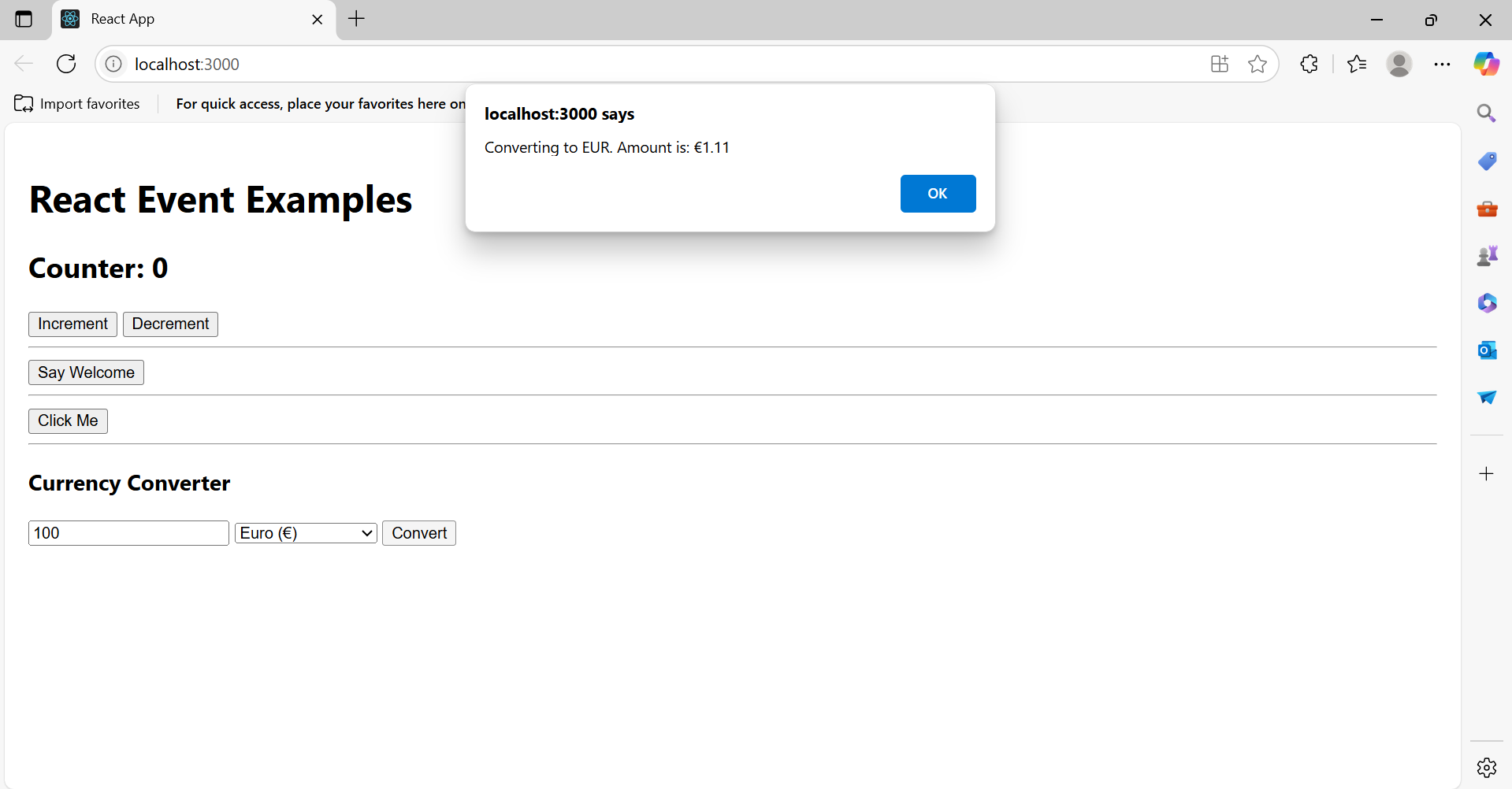
****

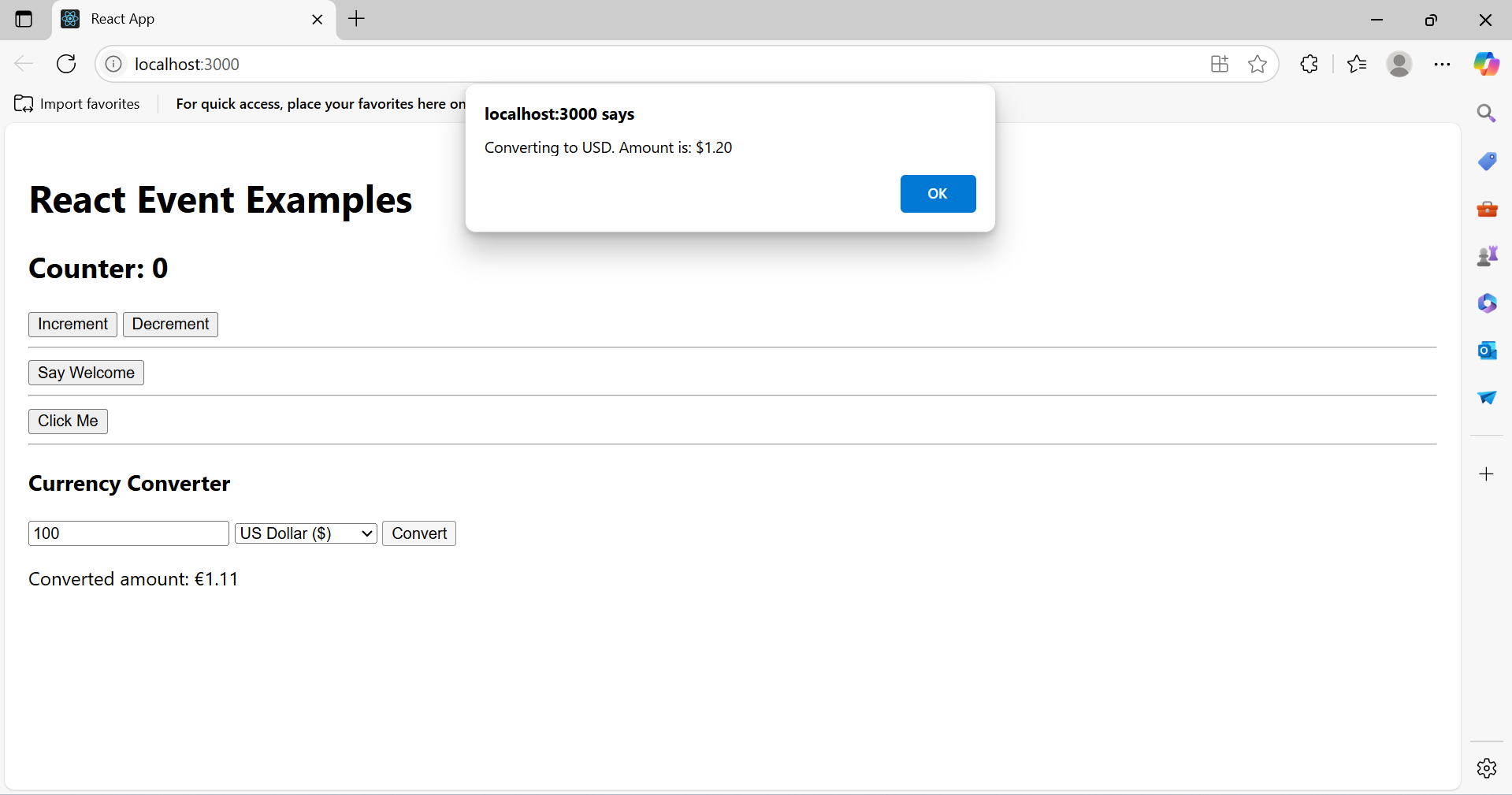
****

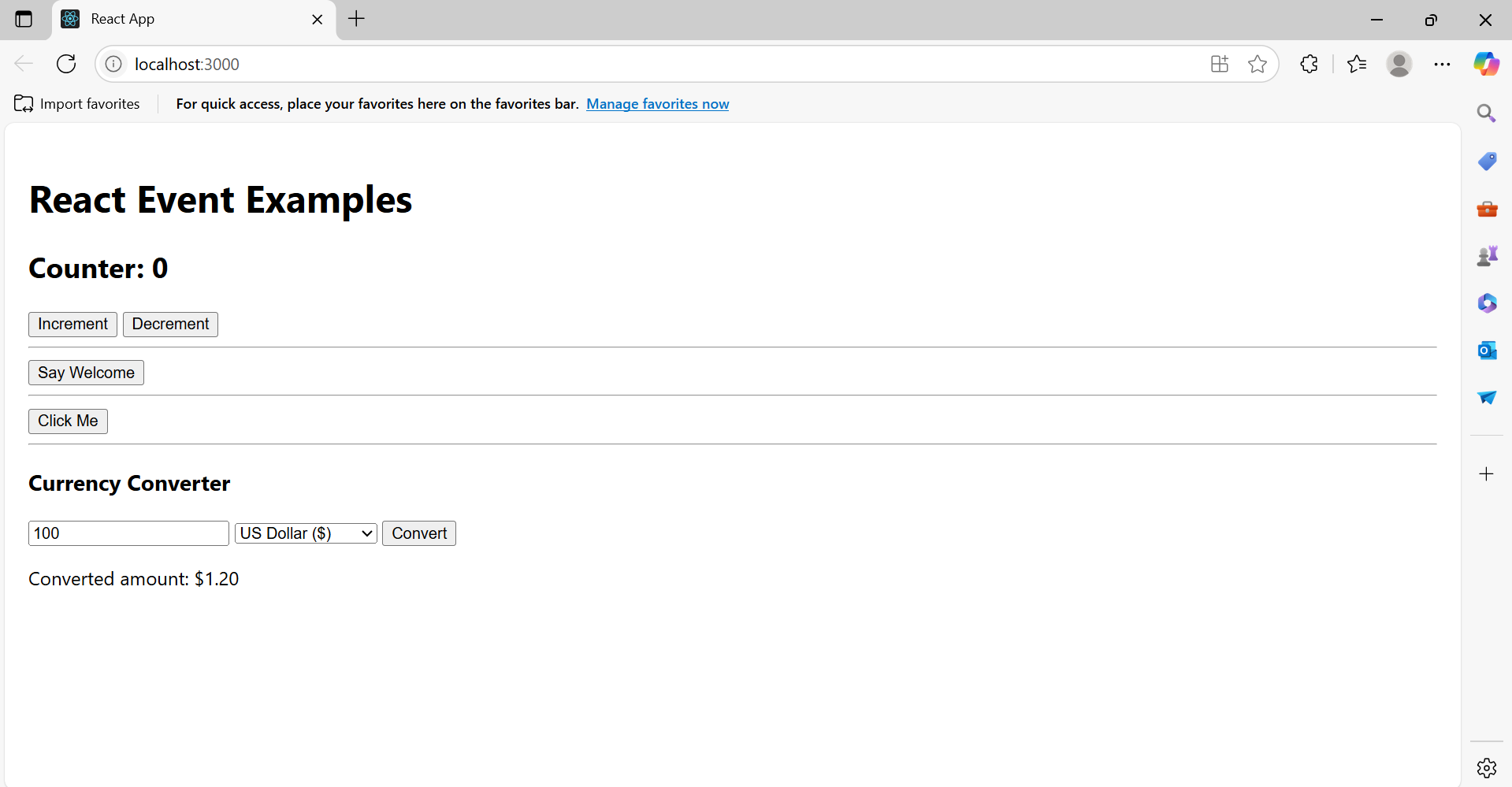
****

****

****

****

****

****

**12.ReactJS\_HOL**

**1. Explain about conditional rendering in React**

Conditional rendering in React means displaying different UI elements based on certain conditions (usually state or props). It's like using if or switch statements inside JSX. You can render elements, components, or return null (to render nothing) depending on the logic.

**2. Define element variables**

Element variables in React store JSX elements in variables which you can render conditionally.

Example:

let message;

if (isLoggedIn) {

message = <h1>Welcome back!</h1>;

} else {

message = <h1>Please sign up.</h1>;

}

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

**3. Explain how to prevent components from rendering**

To prevent a component from rendering, you can:

* Use a condition to return null inside the component.

function Warning(props) {

if (!props.show) {

return null;

}

return <div>Warning!</div>;

}

* Or skip rendering using conditional logic in the parent component.

**App.js**

import React, { useState } from 'react';

function GuestPage({ onLogin }) {

return (

<div style={styles.container}>

<h1>Please Sign Up</h1>

<button onClick={onLogin} style={styles.button}>Login</button>

</div>

);

}

function UserPage({ onLogout }) {

return (

<div style={styles.container}>

<h1>Welcome Back</h1>

<button onClick={onLogout} style={styles.button}>Logout</button>

</div>

);

}

function App() {

const [isLoggedIn, setIsLoggedIn] = useState(false);

const handleLogin = () => setIsLoggedIn(true);

const handleLogout = () => setIsLoggedIn(false);

let content;

if (isLoggedIn) {

content = <UserPage onLogout={handleLogout} />;

} else {

content = <GuestPage onLogin={handleLogin} />;

}

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

}

const styles = {

container: {

textAlign: 'center',

marginTop: '100px'

},

button: {

padding: '10px 20px',

fontSize: '16px',

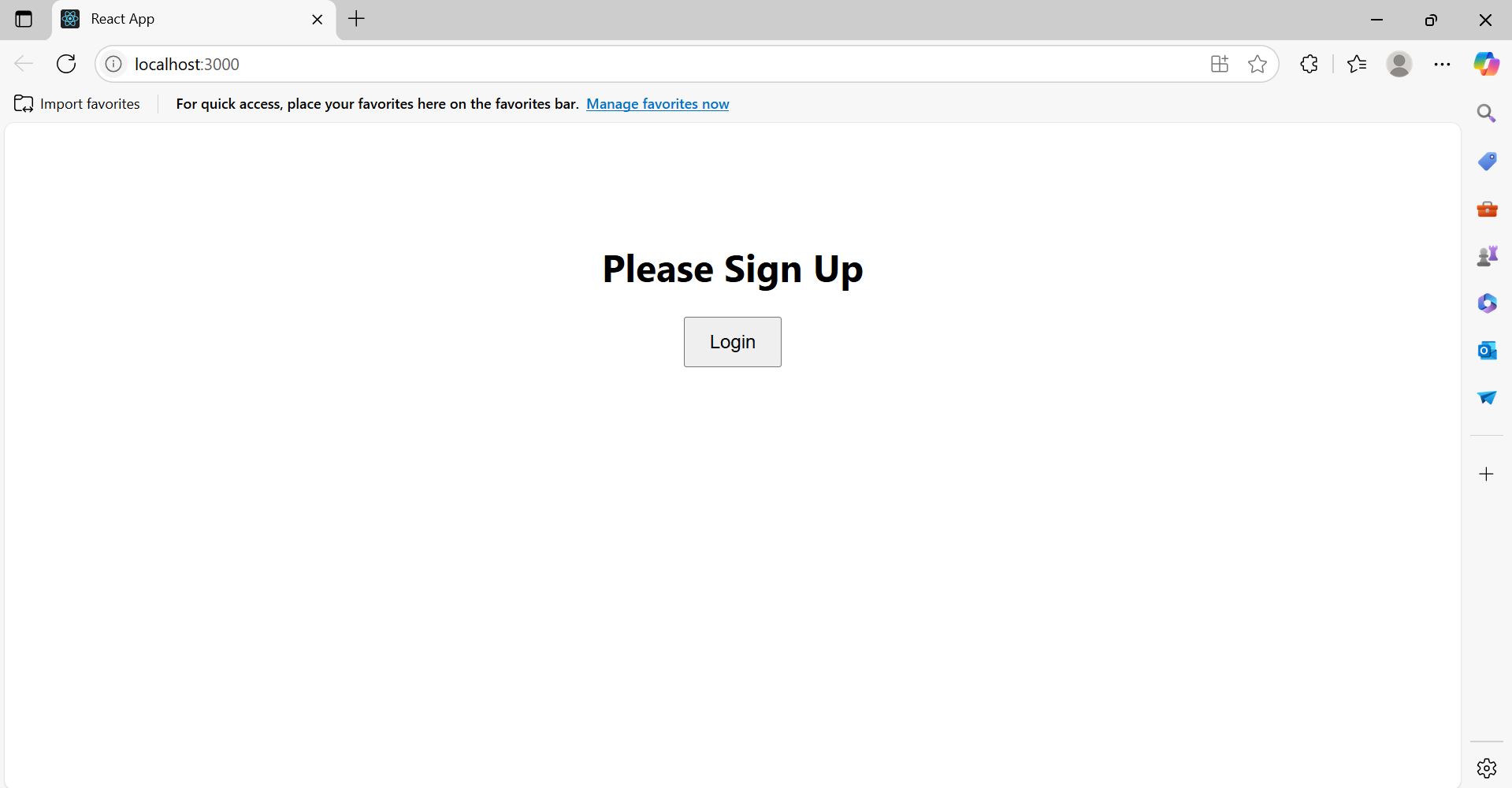
cursor: 'pointer'

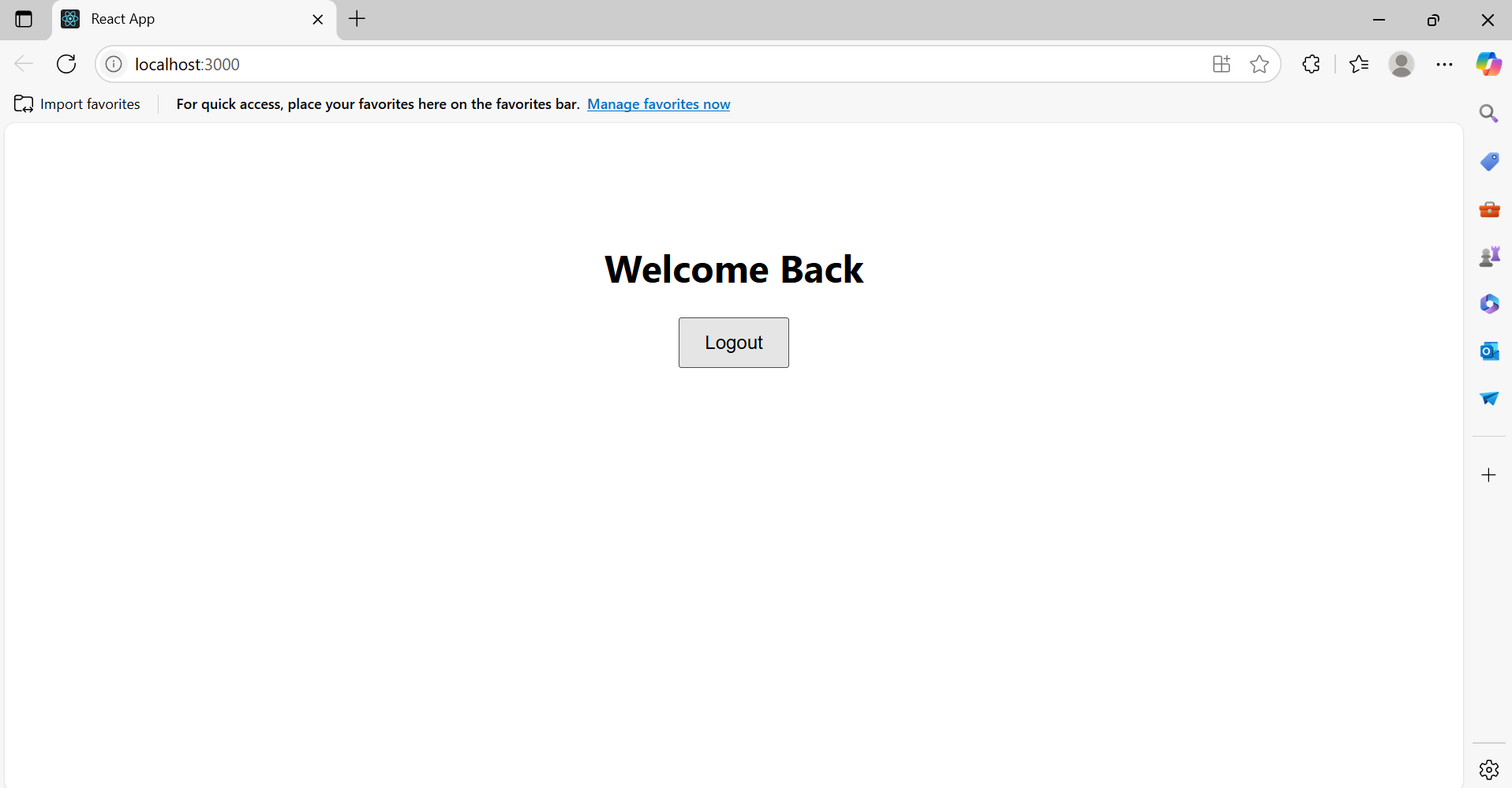
}

};

export default App;

**Output:**

****

****

**13.ReactJS\_HOL**

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

In React, you can conditionally render content using:

* If/else statements
* Ternary operator: {condition ? <A /> : <B />}
* Logical AND (&&) operator: {condition && <Component />}
* Immediately Invoked Function Expression (IIFE) inside JSX
* Element variables: Store JSX in variables and render conditionally

**2. Explain how to render multiple components**

You can render multiple components by simply placing them together inside a parent component (e.g., <div> or <>...</>).

function App() {

return (

<>

<Header />

<Body />

<Footer />

</>

);

}

**3. Define list component**

A list component is a component that takes an array of items and returns a list of rendered elements using .map().

function BlogList({ blogs }) {

return (

<ul>

{blogs.map(blog => <li key={blog.id}>{blog.title}</li>)}

</ul>

);

}

**4. Explain about keys in React applications**

Keys are unique identifiers used in lists to help React identify which items have changed, added, or removed. They improve performance and consistency in rendering.

**5. Explain how to extract components with keys**

When mapping over an array to create components, you should extract the mapped item into a component and assign a unique key:

function BlogItem({ blog }) {

return <li>{blog.title}</li>;

}

{blogs.map(blog => <BlogItem key={blog.id} blog={blog} />)}

**6. Explain React Map, map() function**

The map() function is used in React to iterate over an array and render elements/components from each item.

const items = ['a', 'b', 'c'];

const list = items.map(item => <li key={item}>{item}</li>);

**App.js**

import React from 'react';

import CourseDetails from './CourseDetails';

import BookDetails from './BookDetails';

import BlogDetails from './BlogDetails';

function App() {

return (

<div style={styles.container}>

<div style={styles.section}><CourseDetails /></div>

<div style={styles.verticalLine}></div>

<div style={styles.section}><BookDetails /></div>

<div style={styles.verticalLine}></div>

<div style={styles.section}><BlogDetails /></div>

</div>

);

}

const styles = {

container: {

display: 'flex',

padding: '20px',

justifyContent: 'space-around',

alignItems: 'flex-start',

},

section: {

width: '30%',

padding: '10px',

},

verticalLine: {

width: '2px',

backgroundColor: '#ccc',

height: '100%',

},

};

export default App;

**CourseDetails.js**

import React from 'react';

function CourseDetails() {

const courses = [

{ name: 'Angular', date: '4/5/2021' },

{ name: 'React', date: '6/3/20201' },

];

return (

<div>

<h2>Course Details</h2>

<ul>

{courses.map((course, index) => (

<li key={index}>

<strong>{course.name}</strong><br />

{course.date && <span>{course.date}</span>}

</li>

))}

</ul>

</div>

);

}

export default CourseDetails;

**BookDetails.js**

import React from 'react';

function BookDetails() {

const books = [

{ title: 'Master React', price: '670' },

{ title: 'Deep Dive into Angular 11' ,price: ‘800’},

{ title: 'Mongo Essentials', price: '450' },

];

return (

<div>

<h2>Book Details</h2>

<ul>

{books.map((book, index) => (

<li key={index}>

<strong>{book.title}</strong><br />

{book.price && <span>{book.price}</span>}

</li>

))}

</ul>

</div>

);

}

export default BookDetails;

**BlogDetails.js**

import React from 'react';

function BlogDetails() {

const blogs = [

{ title: 'React Learning', author: 'Stephen Biz', content: 'Welcome to learning React!' },

{ title: 'Installation', author: 'Schewzdenier', content: 'You can install React from nom' },

];

return (

<div>

<h2>Blog Details</h2>

<ul>

{blogs.map((blog, index) => (

<li key={index}>

<strong>{blog.title}</strong><br />

<em>{blog.author}</em><br />

<span>{blog.content}</span>

</li>

))}

</ul>

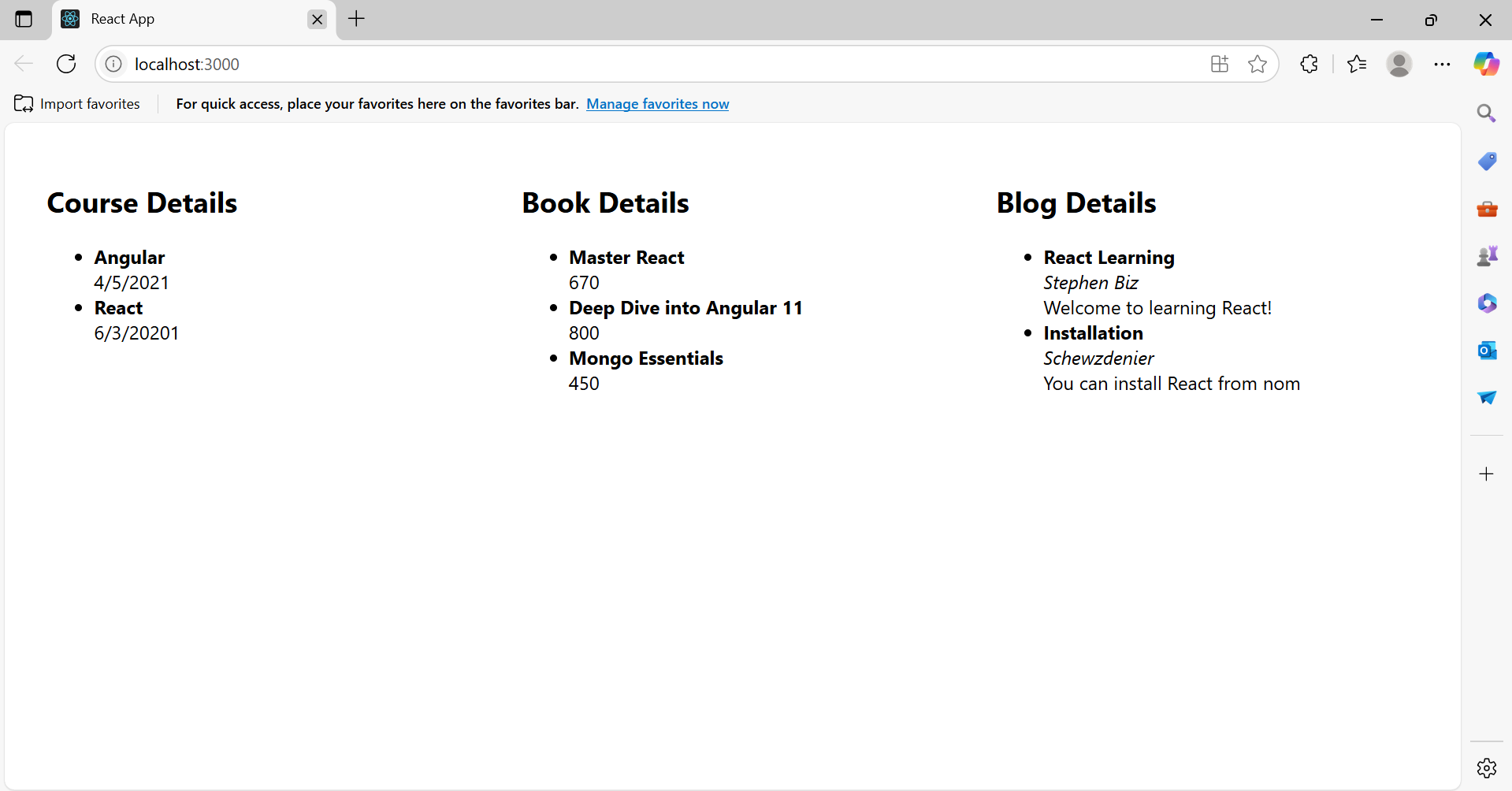
</div>

);

}

export default BlogDetails;

**Output:**

****

**14.ReactJS\_HOL**

**1. Explain the need and benefits of React Context API**

The React Context API is used for passing data deeply through the component tree without having to pass props manually at every level. It's especially useful when dealing with global data like themes, authentication, or user info.

Benefits:

* Avoids prop drilling
* Provides a cleaner structure
* Promotes code reusability
* Useful for global states (like themes, user, locale)

**2. Working with createContext()**

import React from 'react';

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

export default ThemeContext;

* createContext() creates a context object.
* We use a Provider to wrap components and Consumer or useContext() to access it.

**3. List the types of Router Components**

From react-router-dom, common router components include:

* <BrowserRouter> – base router for web apps using HTML5 history API
* <Routes> – wrapper for all <Route> elements
* <Route> – defines path and component
* <Link> – for navigation without page reload
* <NavLink> – like <Link> but with active styling
* <useNavigate>, <useParams> – React Router hooks

**ThemeContext.js**

import { createContext } from "react";

const ThemeContext = createContext("light");

export default ThemeContext;

**App.js**

import React, { useState } from "react";

import EmployeeList from "./EmployeeList";

import ThemeContext from "./ThemeContext";

import './App.css';

function App() {

const [theme, setTheme] = useState("light");

const toggleTheme = () => {

setTheme((prev) => (prev === "light" ? "dark" : "light"));

};

return (

<ThemeContext.Provider value={theme}>

<div className={`App ${theme}`}>

<h1 className={`header ${theme === "dark" ? "white-text" : ""}`}>

Employee Management System

</h1>

<button className="toggle-btn" onClick={toggleTheme}>

Toggle Theme

</button>

<EmployeeList />

</div>

</ThemeContext.Provider>

);

}

export default App;

**EmployeeList.js**

import React from "react";

import EmployeeCard from "./EmployeeCard";

const employees = [

{ id: 1, name: "John Doe", role: "Frontend Developer" },

{ id: 2, name: "Jane Smith", role: "Backend Developer" },

{ id: 3, name: "Alice Johnson", role: "Full Stack Developer" },

];

function EmployeeList() {

return (

<div className="employee-list">

{employees.map((emp) => (

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

))}

</div>

);

}

export default EmployeeList;

**EmployeeCard.js**

import React, { useContext } from "react";

import ThemeContext from "./ThemeContext";

function EmployeeCard({ employee }) {

const theme = useContext(ThemeContext);

return (

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

<h3>{employee.name}</h3>

<p>{employee.role}</p>

<button className={`card-btn ${theme}`}>View</button>

</div>

);

}

export default EmployeeCard;

**App.css**

.App {

text-align: center;

padding: 2rem;

}

.toggle-btn {

margin: 1rem;

padding: 0.5rem 1rem;

font-size: 16px;

}

.employee-list {

display: flex;

justify-content: center;

gap: 2rem;

flex-wrap: wrap;

margin-top: 2rem;

}

.card {

padding: 1rem;

border-radius: 8px;

width: 200px;

box-shadow: 0 0 8px rgba(0, 0, 0, 0.1);

}

.card-btn {

margin-top: 1rem;

padding: 0.4rem 1rem;

border: none;

cursor: pointer;

font-weight: bold;

}

.light {

background-color: #ffffff;

color: #333333;

}

.card-btn.light {

background-color: #f0f0f0;

color: #000;

}

.dark {

background-color: #1e1e1e;

color: #ffffff;

}

.card-btn.dark {

background-color: #333;

color: #fff;

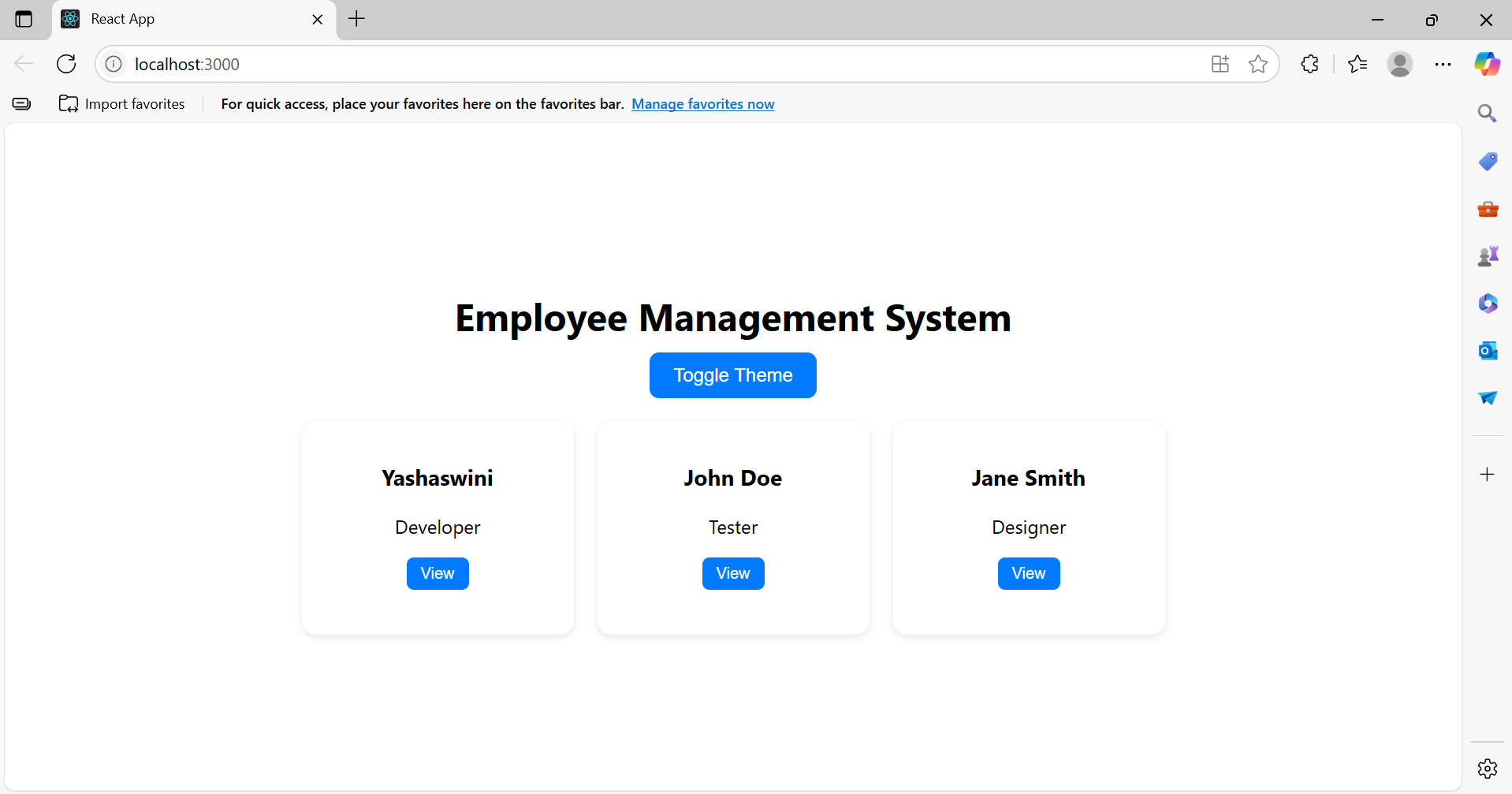
}

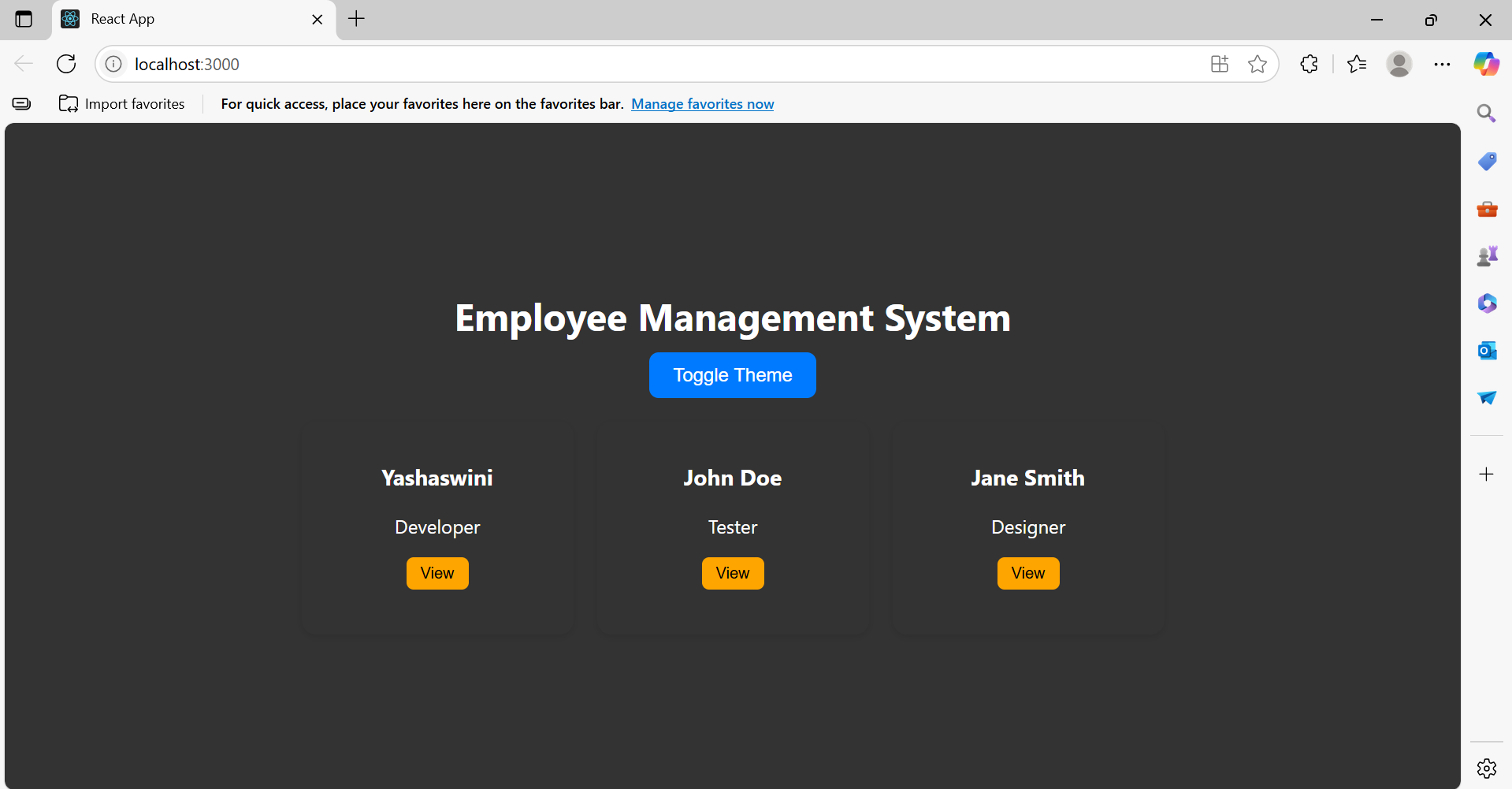
.white-text {

color: white;

}

**Output:**

****

****

**15.ReactJS\_HOL**

**• Explain about React forms**

React forms are used to capture user input through various controls like <input>, <textarea>, and <select>. In React, forms are handled by maintaining the form data in the component's state, making it easier to control and validate user input.

**• Define controlled components**

Controlled components are form elements whose values are controlled by React state. The value of the input field is bound to the component’s state and updated via onChange handlers.

**• Explain about various input controls**

React supports:

* <input type="text"> – Textbox
* <textarea> – Multi-line input
* <button> – Submit actions
* <select> – Dropdown selection

**• Explain about handling forms**

Handling forms in React involves:

* Setting up state variables for each input field.
* Using onChange handlers to update the state.
* Preventing default form behavior using e.preventDefault() in the submit handler.

**• Explain about submitting forms**

On submitting, the handleSubmit function is triggered, which:

* Prevents page reload.
* Processes the form data.
* Displays it using alert() or any method (API call, storage, etc.).

**ComplaintRegister.js**

import React, { useState } from 'react';

const ComplaintRegister = () => {

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

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

const handleSubmit = (e) => {

e.preventDefault();

const referenceId = 'TRN' + Math.floor(100000 + Math.random() \* 900000);

alert(`Thanks ${name}\n\nYour Complaint was Submitted\n\nTransaction ID: ${referenceId}`);

setName('');

setComplaint('');

};

return (

<div style={styles.container}>

<h2>Register your complaints here!!!</h2>

<form onSubmit={handleSubmit} style={styles.form}>

<label>Name:</label>

<input

type="text"

value={name}

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

required

style={styles.input}

/>

<label>Complaint</label>

<textarea

value={complaint}

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

required

style={styles.textarea}

/>

<button type="submit" style={styles.button}>Submit</button>

</form>

</div>

);

};

const styles = {

container: {

margin: '50px auto',

width: '400px',

padding: '20px',

border: '2px solid #ccc',

borderRadius: '10px',

backgroundColor: '#f7f7f7',

fontFamily: 'Arial'

},

form: {

display: 'flex',

flexDirection: 'column',

},

input: {

marginBottom: '15px',

padding: '10px',

fontSize: '16px',

},

textarea: {

marginBottom: '15px',

padding: '10px',

fontSize: '16px',

height: '80px',

},

button: {

padding: '10px',

fontSize: '16px',

backgroundColor: '#007BFF',

color: 'white',

border: 'none',

cursor: 'pointer'

}

};

export default ComplaintRegister;

**App.js**

import React from 'react';

import ComplaintRegister from './ComplaintRegister';

function App() {

return (

<div>

<ComplaintRegister />

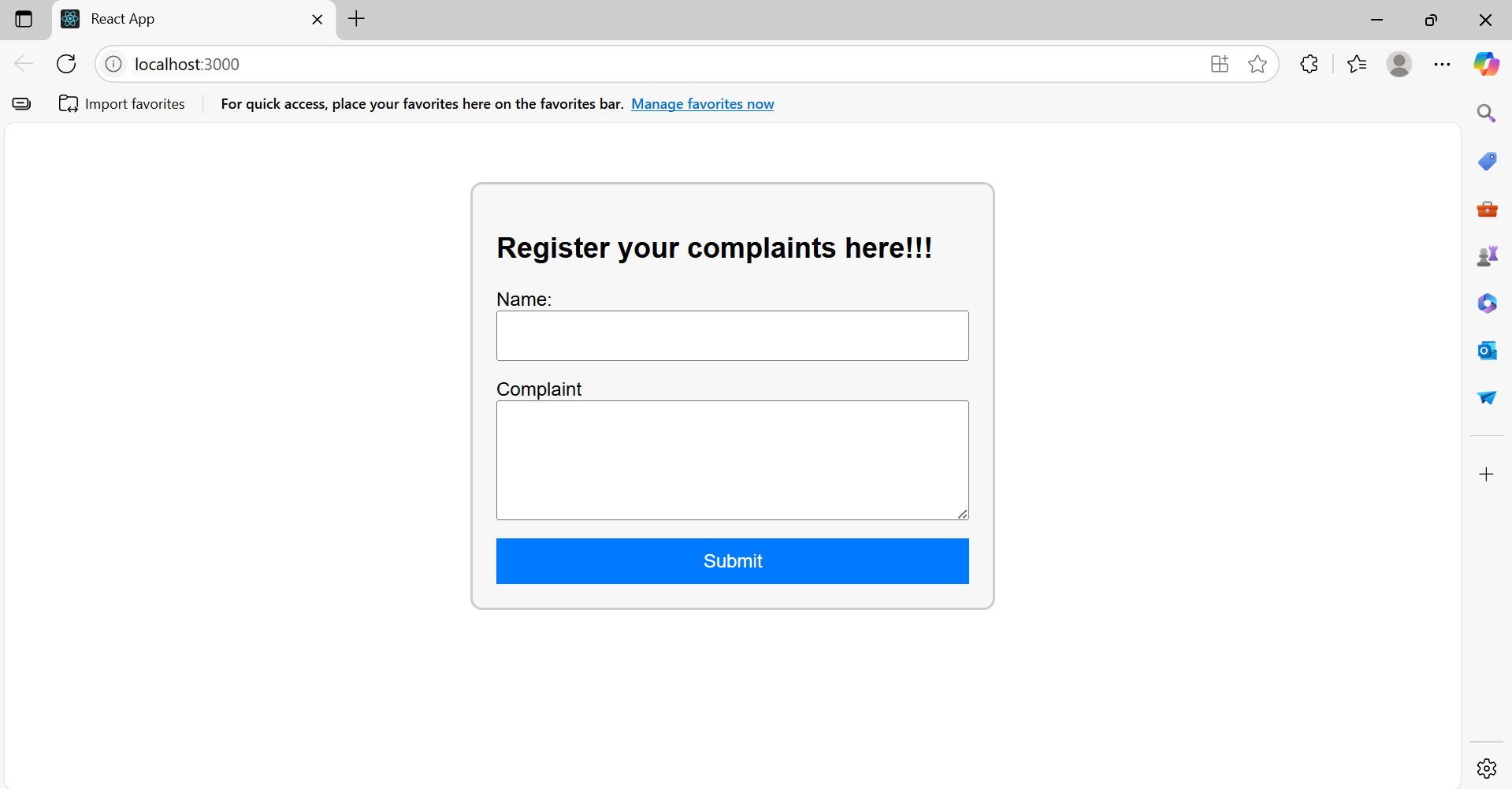
</div>

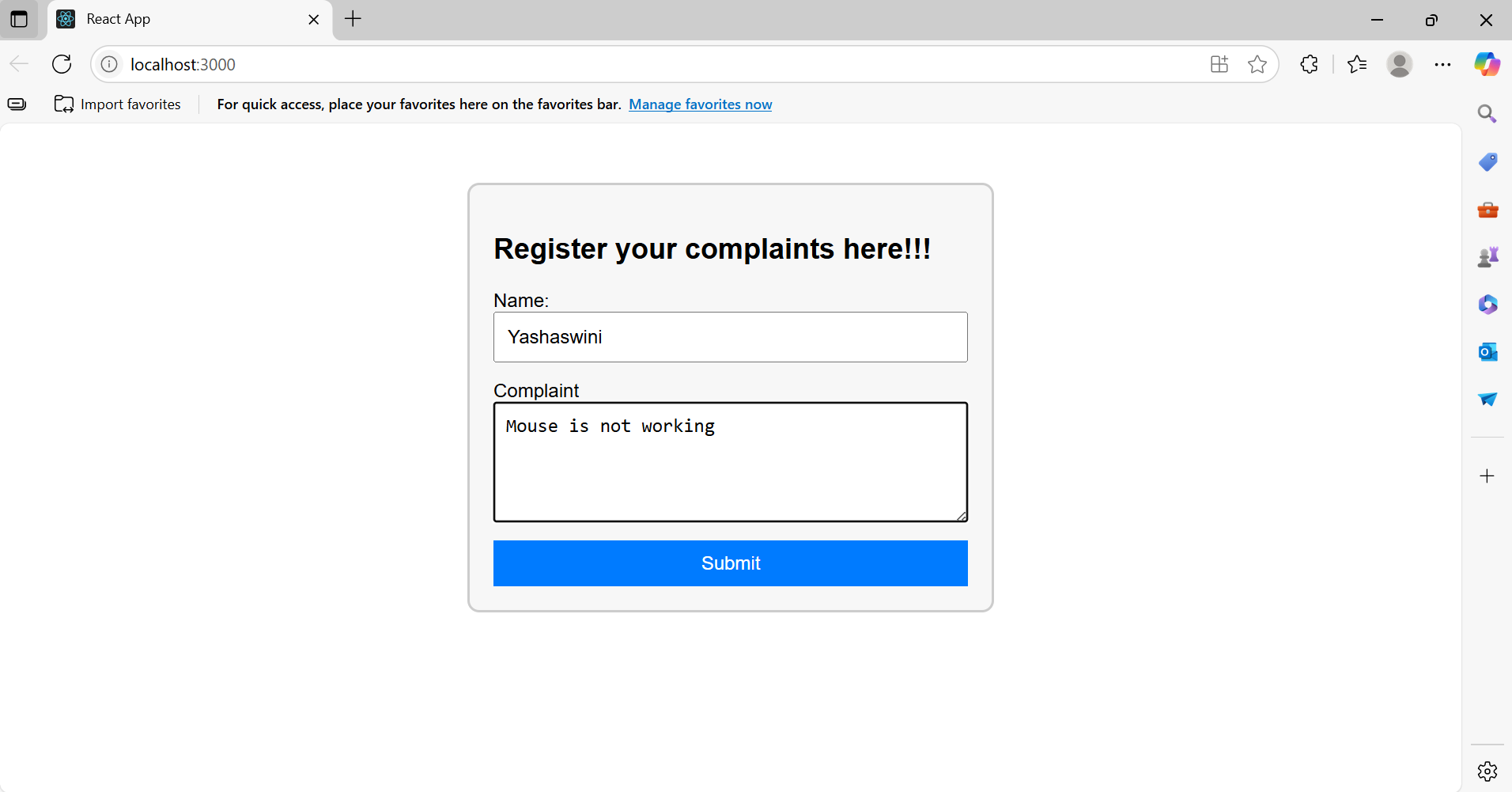
);

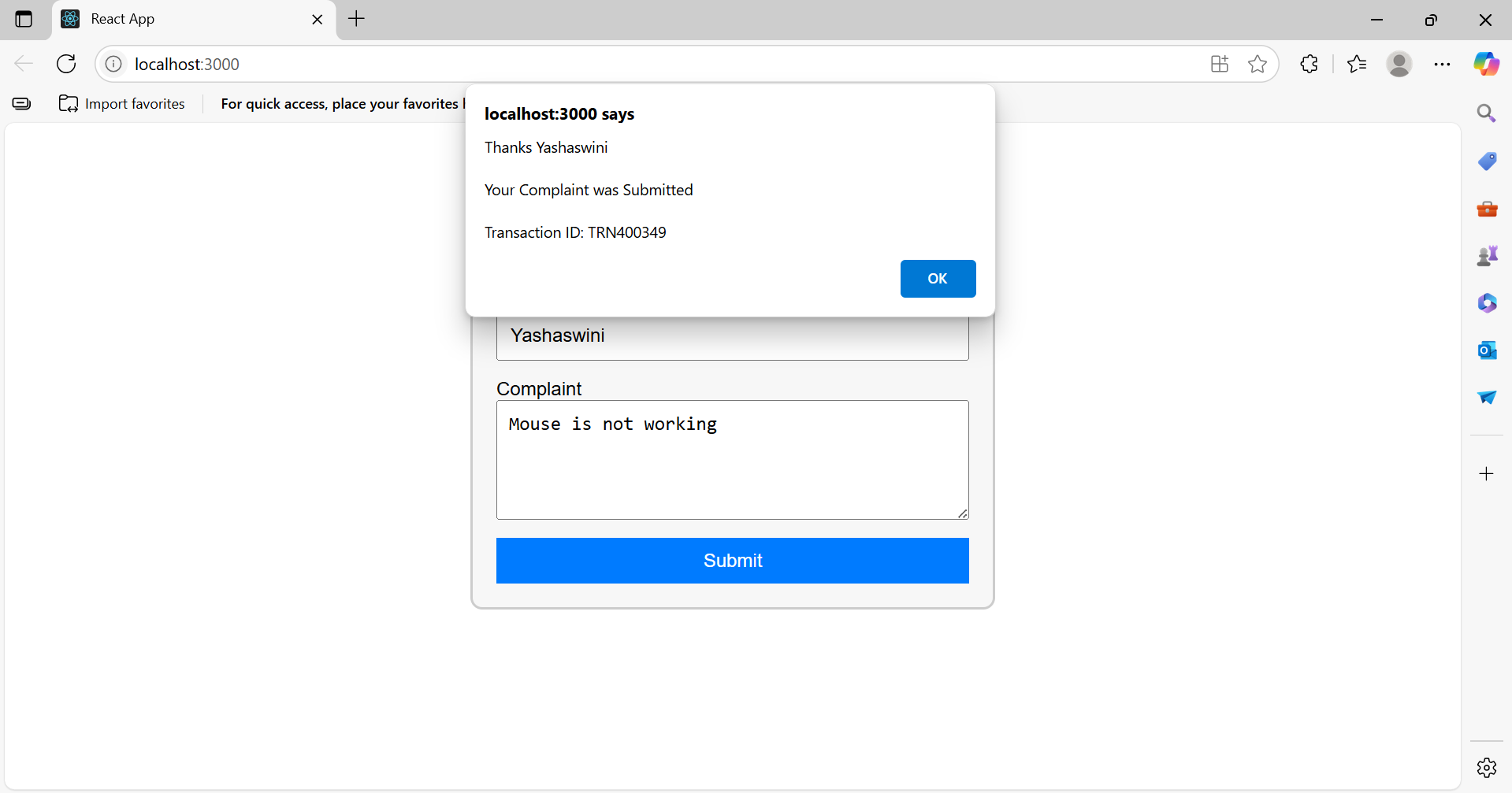
}

export default App;

**Output:**

****

****

****

**16.ReactJS\_HOL**

**• Explain React Forms validation**

React forms validation ensures that the user provides correct and complete data before submission. It can be done:

* Inline (onChange)
* On form submission (onSubmit)
* Using state to track errors and values

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

| **HTML Form** | **React Form** |
| --- | --- |
| Uses DOM for state | Uses component state (useState) |
| Less control over data | Fully controlled via React |
| Uses onsubmit, onchange directly | Uses handler functions |
| No JSX | Built using JSX components |

**• Explain about controlled components**

In controlled components:

* The form input's value is tied to the component’s state.
* The state is updated using onChange.

**• Identify various React Form input controls**

React supports:

* <input type="text">, <input type="email">, <input type="password">
* <textarea>
* <button type="submit">

**• Explain how to handle React Forms**

React forms are handled by:

1. Using useState to store each field.
2. Writing onChange handlers to update state.
3. Writing onSubmit handler to validate and process data.

**• Explain about submitting forms in React**

On submit:

* Prevent default form submission using e.preventDefault()
* Validate form fields
* Show feedback (e.g., alert() or error message)

**App.js**

import React from "react";

import Register from "./Register";

import "./App.css";

function App() {

return (

<div className="App">

<Register />

</div>

);

}

export default App;

**Register.js**

import React, { useState } from "react";

function Register() {

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

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

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

const handleSubmit = (e) => {

e.preventDefault()

if (name.length < 5) {

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

return;

}

if (!email.includes("@") || !email.includes(".")) {

alert("Email is not valid");

return;

}

if (password.length < 8) {

alert("Password must be 8 characters long");

return;

}

alert(`Thanks ${name}!\nYou have been successfully registered.`);

};

return (

<div className="form-container">

<h2>Register Here!!!</h2>

<form onSubmit={handleSubmit}>

<label>Name:</label>

<input

type="text"

value={name}

placeholder="Enter your name"

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

/>

<label>Email:</label>

<input

type="email"

value={email}

placeholder="Enter your email"

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

/>

<label>Password:</label>

<input

type="password"

value={password}

placeholder="Enter password"

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

/>

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

</form>

</div>

);

}

export default Register;

**App.css**

body {

margin: 0;

font-family: Arial, sans-serif;

background-color: #f4f4f4;

}

.form-container {

background-color: white;

width: 400px;

margin: 100px auto;

padding: 30px;

border-radius: 12px;

box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);

}

h2 {

text-align: center;

color: #333;

}

form {

display: flex;

flex-direction: column;

}

label {

margin-top: 15px;

margin-bottom: 5px;

font-weight: bold;

}

input {

padding: 10px;

border: 1px solid #ccc;

border-radius: 8px;

font-size: 16px;

}

button {

margin-top: 25px;

padding: 12px;

background-color: #007bff;

color: white;

border: none;

font-size: 16px;

border-radius: 8px;

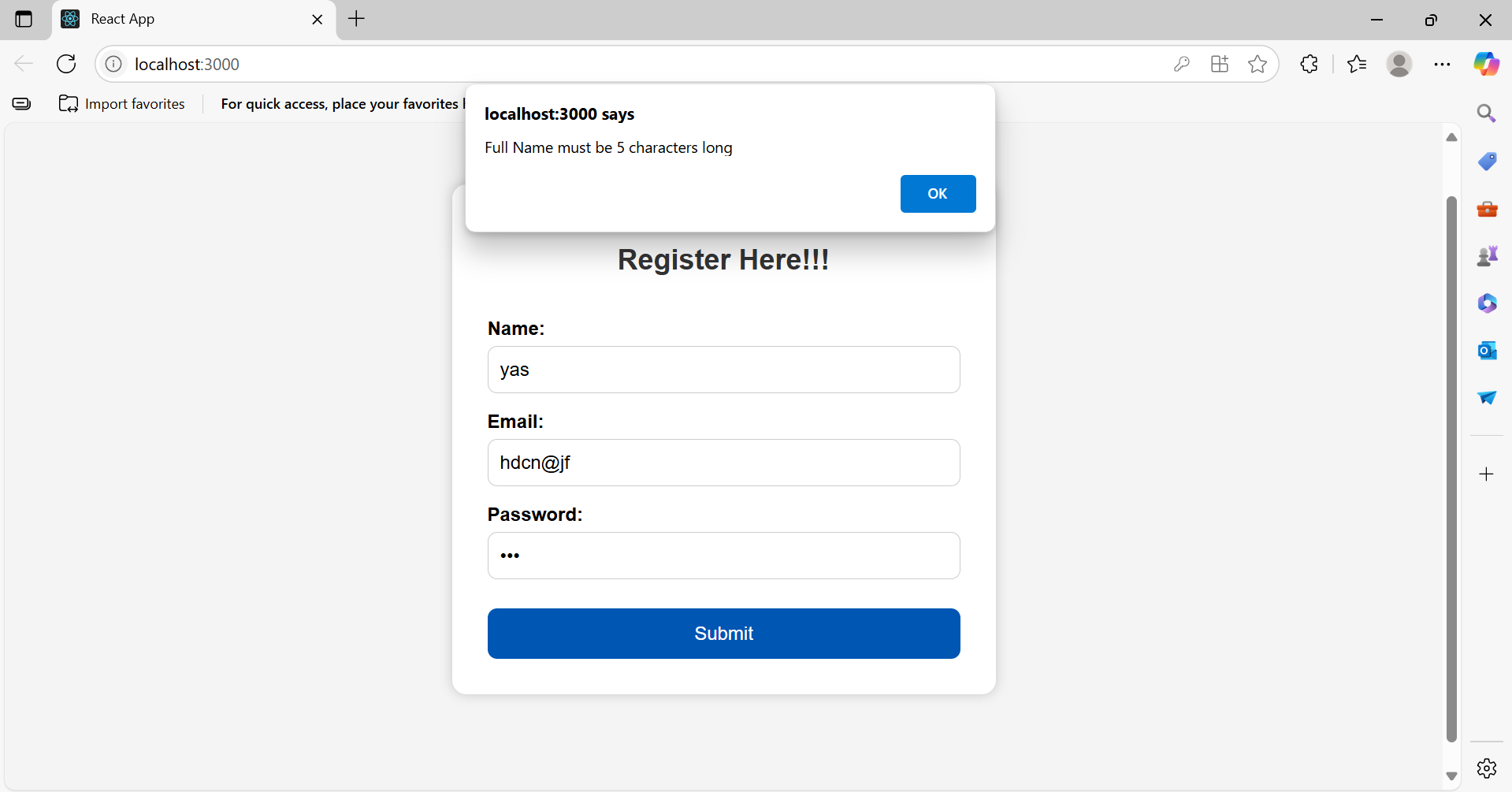
cursor: pointer;

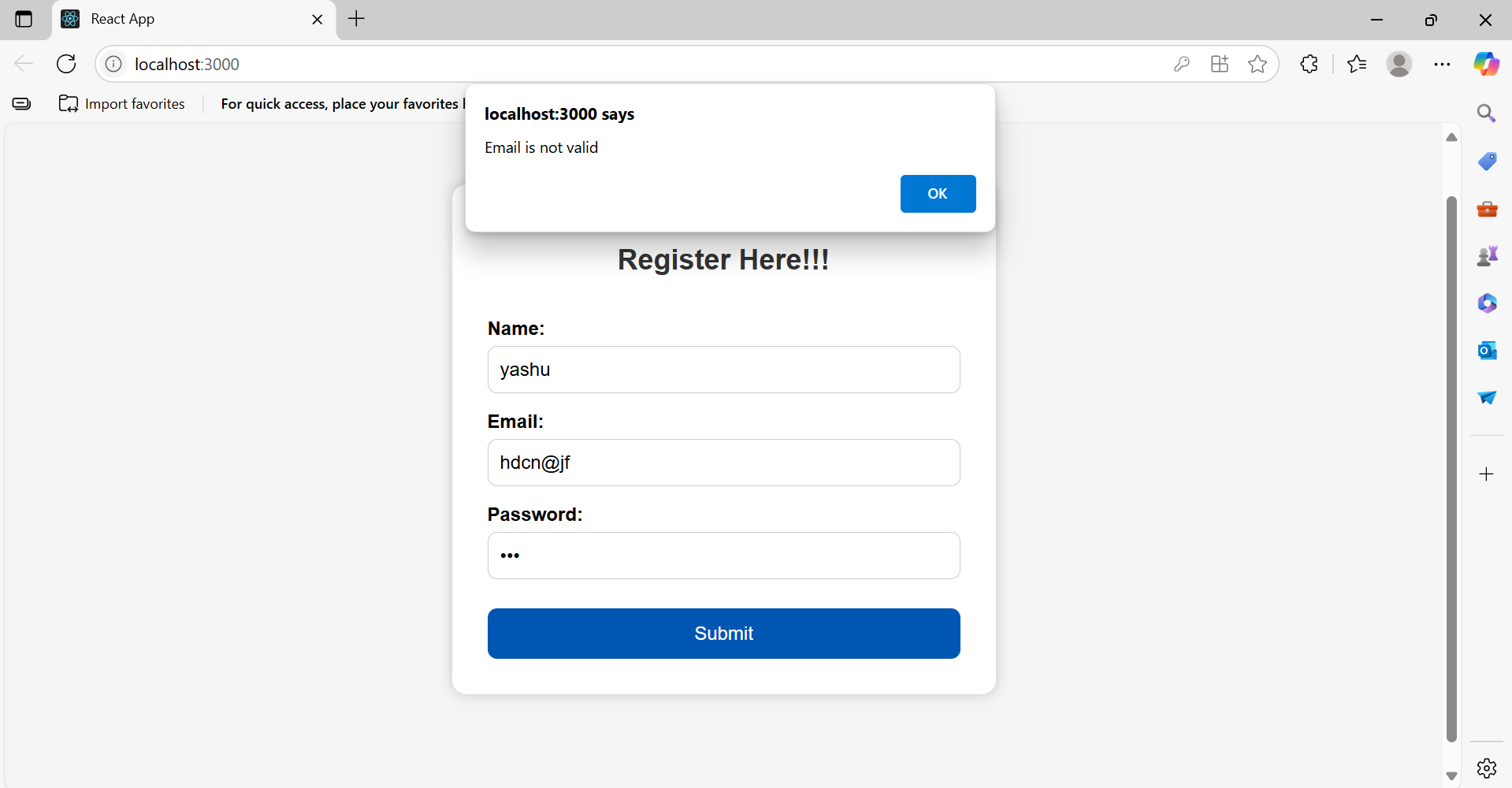
}

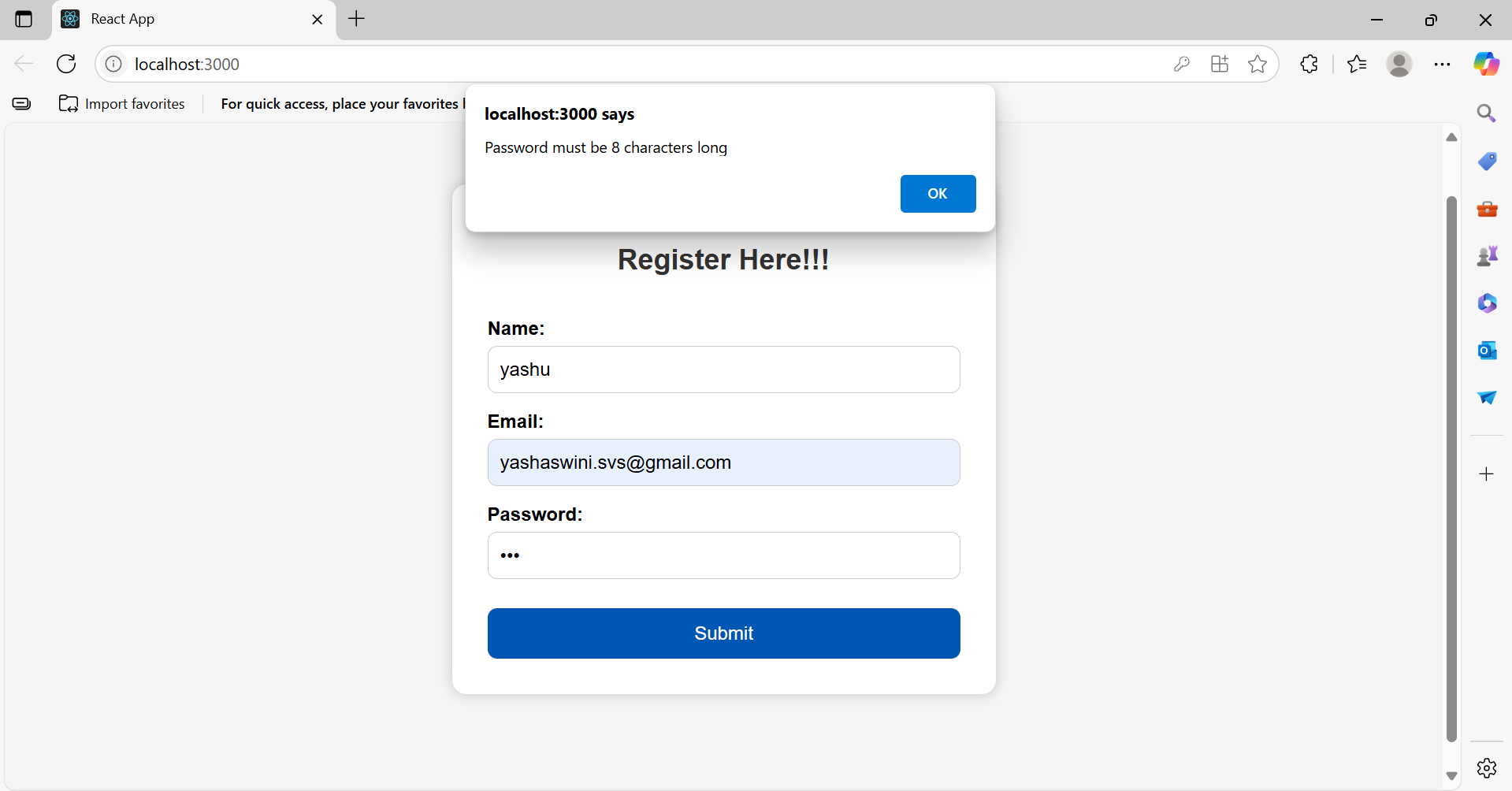
button:hover {

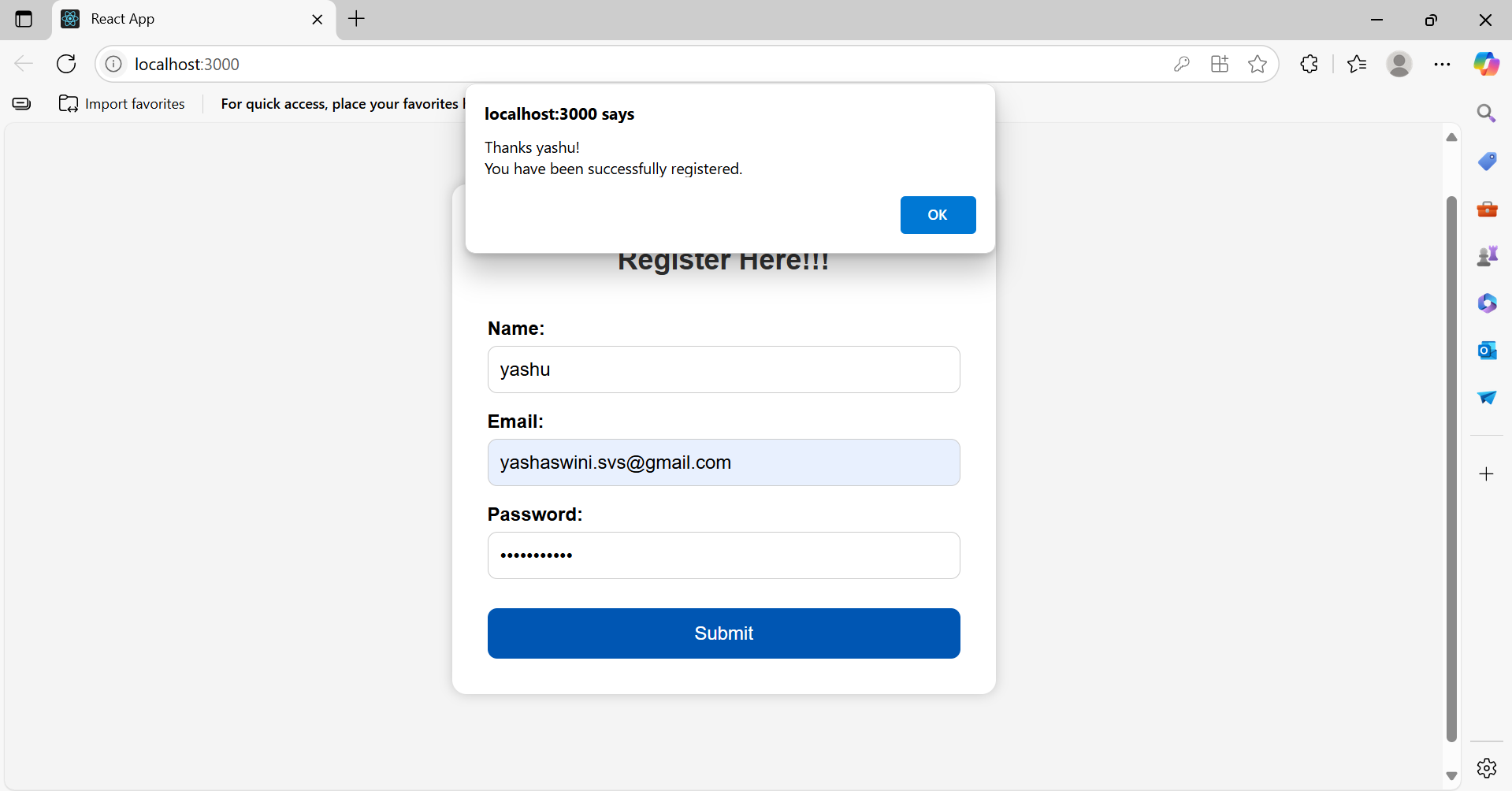
background-color: #0056b3;

}

**Output:  
**

****

****

****

**17.ReactJS\_HOL**

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

Answer:  
To consume a REST API in a React application:

1. Use fetch or axios to make HTTP requests.
2. Call the API in a lifecycle method (like componentDidMount() for class components or useEffect() for functional components).
3. Store the data in the component’s state using this.setState or the useState hook.
4. Render the data conditionally to handle loading and error states.

**Construct a React application that invokes the REST API and fetches data from the API**

Answer:  
We create a React app (named fetchuserapp) that:

* Uses componentDidMount() to call https://api.randomuser.me/
* Fetches a random user
* Stores their title, first name, and picture in state
* Displays the data in the browser

**Getuser.js**

import React, { Component } from 'react';

class Getuser extends Component {

constructor(props) {

super(props);

this.state = {

person: null,

loading: true,

};

}

async componentDidMount() {

const url = "https://api.randomuser.me/";

const response = await fetch(url);

const data = await response.json();

this.setState({ person: data.results[0], loading: false });

console.log(data.results[0]);

}

render() {

const { person, loading } = this.state;

if (loading) {

return <h3>Loading...</h3>;

}

return (

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

<h1>

{person.name.title} {person.name.first} {person.name.last}

</h1>

<img src={person.picture.large} alt="User" />

</div>

);

}

}

export default Getuser;

**App.js**

import React from 'react';

import Getuser from './Getuser';

function App() {

return (

<div className="App">

<Getuser />

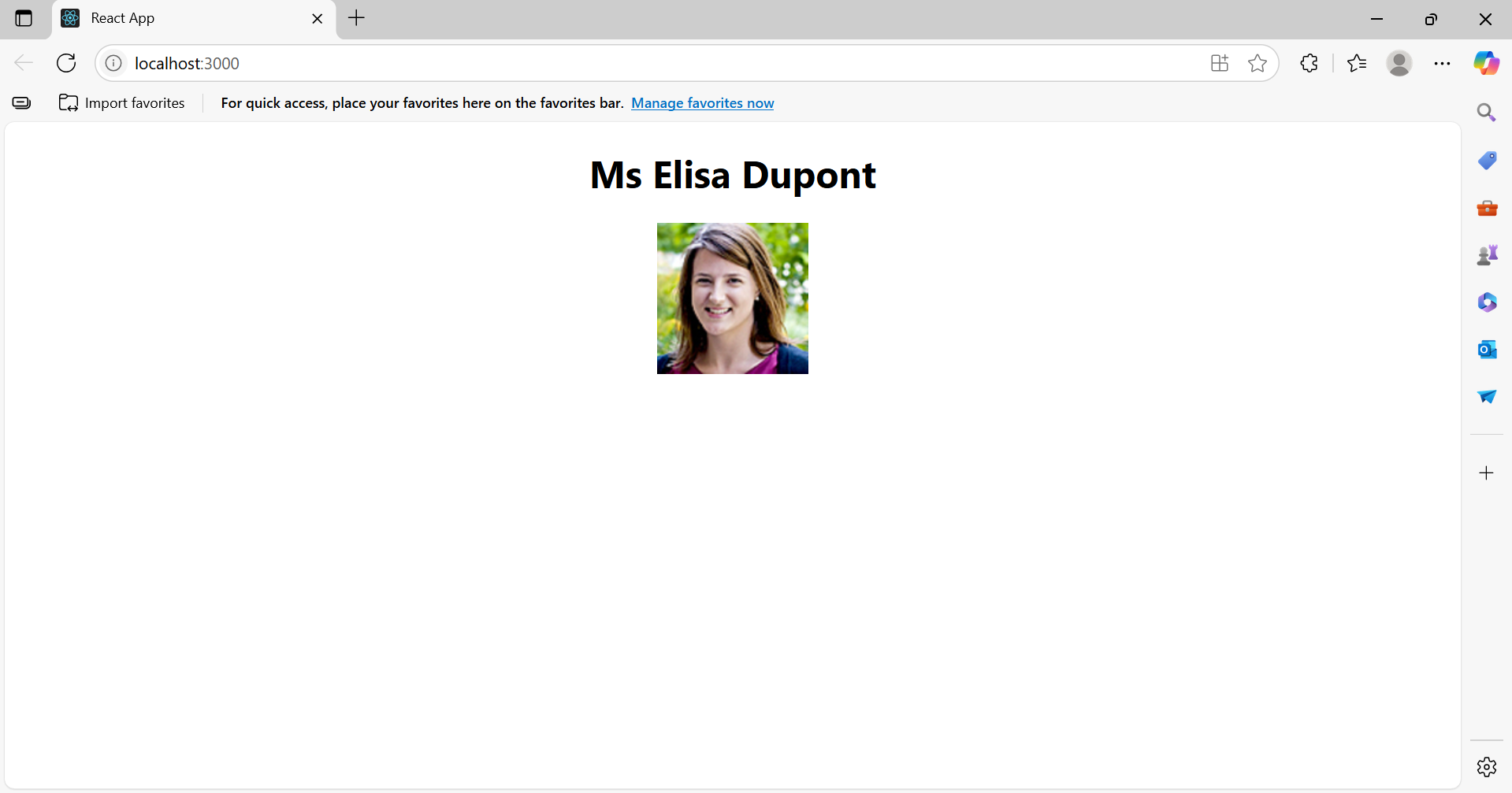
</div>

);

}

export default App;

**Output:**

****

**18.ReactJS\_HOL**

**1. Explain the need for Unit Testing in React**

Answer:  
Unit testing ensures that individual components in a React application work correctly in isolation. It helps:

* Catch bugs early
* Simplify refactoring
* Ensure code behaves as expected
* Improve confidence when making changes

**2. Working with Jest and Enzyme in React**

Answer:

* Jest is a JavaScript testing framework commonly used with React.
* Enzyme (by Airbnb) is a JavaScript testing utility that makes it easier to assert, manipulate, and traverse React components.

They are used together to:

* Mount components
* Simulate events
* Test props, state, and DOM structure

**3. List the types of Router Components in React Router**

Answer:  
Common router components from react-router-dom include:

* <BrowserRouter>: Uses HTML5 history API
* <HashRouter>: Uses hash portion of the URL (for legacy support)
* <Routes> / <Route>: For route definitions
* <Link> / <NavLink>: Navigation without page reload
* <Outlet>: Renders nested route elements
* <Navigate>: Redirects programmatically

**Cohort.js**

export const CohortData = [

{

id: 1,

cohortCode: 'CTS-2025-JUL',

description: 'Full Stack Training Batch',

status: 'Ongoing'

},

{

id: 2,

cohortCode: 'CTS-2025-MAY',

description: 'Frontend Basics',

status: 'Completed'

}

];

**CohortDetails.js**

import React from 'react';

function CohortDetails({ cohort }) {

if (!cohort) return <div>No cohort data available</div>;

return (

<div>

<h3>{cohort.cohortCode}</h3>

<p>{cohort.description}</p>

<p>Status: {cohort.status}</p>

</div>

);

}

export default CohortDetails;

**App.js**

import React from 'react';

import CohortDetails from './CohortDetails';

import { CohortData } from './Cohort';

function App() {

return (

<div className="App">

<h2>Cohort Dashboard</h2>

<CohortDetails cohort={CohortData[0]} />

</div>

);

}

export default App;

**App.test.js**

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

import App from './App';

test('renders cohort dashboard heading', () => {

render(<App />);

const headingElement = screen.getByText(/cohort dashboard/i);

expect(headingElement).toBeInTheDocument();

});

**CohortDetails.test.js**

import React from 'react';

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

import CohortDetails from './CohortDetails';

import { CohortData } from './Cohort';

describe('Cohort Details Component', () => {

test('should create the component and render no cohort message when no props', () => {

render(<CohortDetails />);

expect(screen.getByText(/no cohort data available/i)).toBeInTheDocument();

});

test('should initialize the props and render cohort details', () => {

const cohort = CohortData[0];

render(<CohortDetails cohort={cohort} />);

expect(screen.getByText(cohort.cohortCode)).toBeInTheDocument();

expect(screen.getByText(cohort.description)).toBeInTheDocument();

expect(screen.getByText(`Status: ${cohort.status}`)).toBeInTheDocument();

});

test('should display cohort code in an h3 tag', () => {

const cohort = CohortData[0];

render(<CohortDetails cohort={cohort} />);

const heading = screen.getByRole('heading', { level: 3 });

expect(heading).toHaveTextContent(cohort.cohortCode);

});

test('should match snapshot', () => {

const cohort = CohortData[0];

const { container } = render(<CohortDetails cohort={cohort} />);

expect(container).toMatchSnapshot();

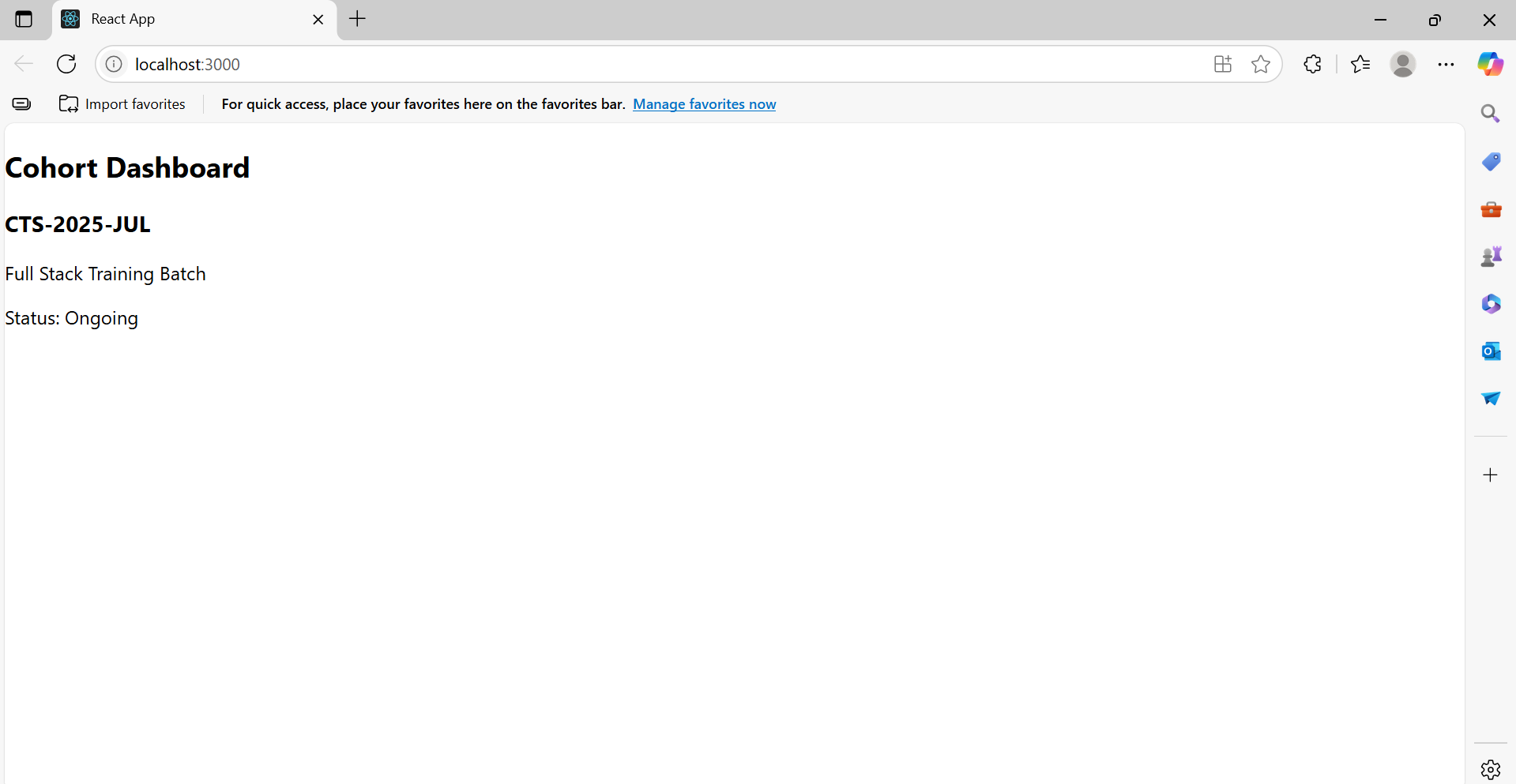
});

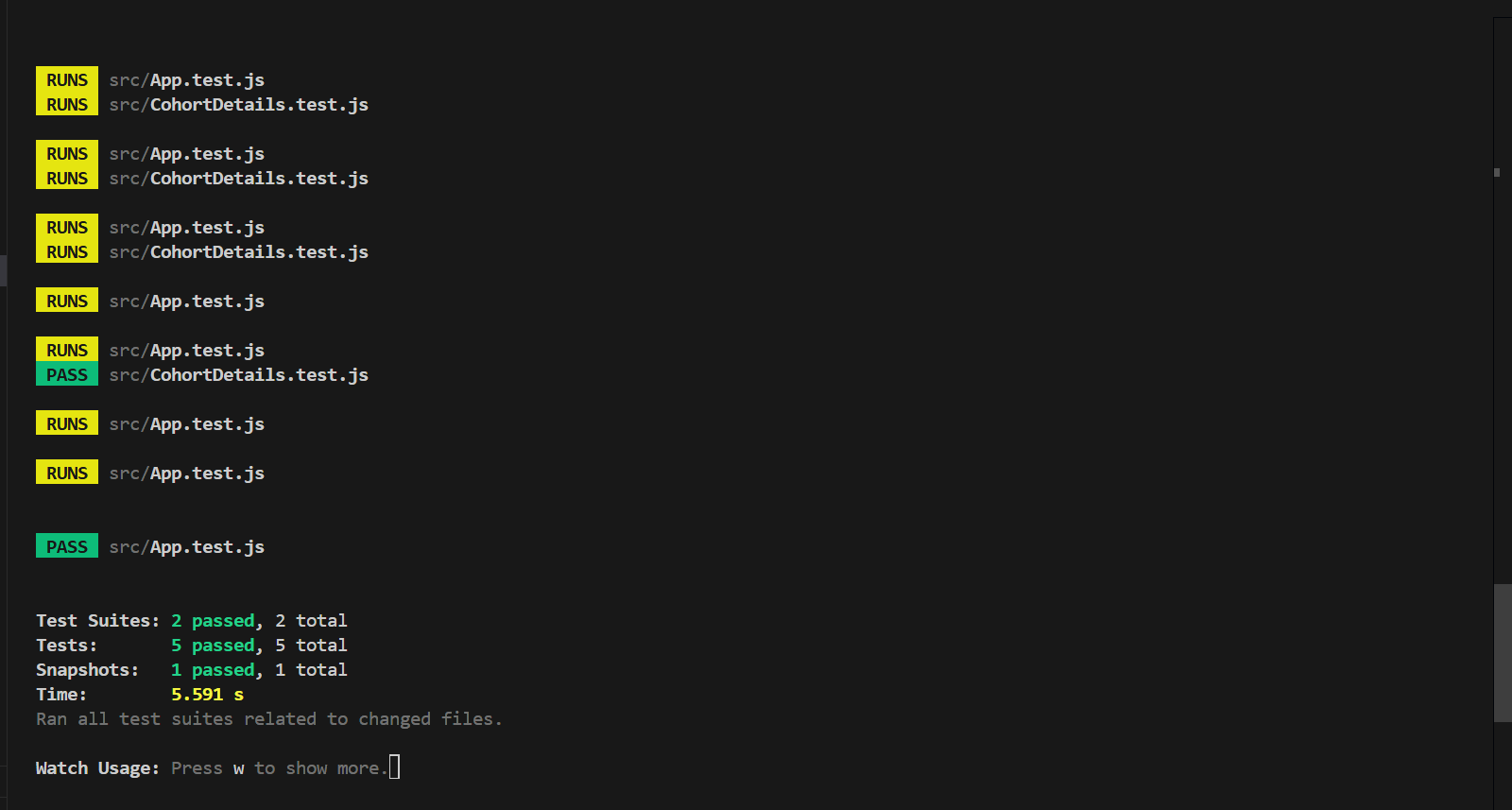
});

**setupTests.js**

import '@testing-library/jest-dom';

**Output:**

****

****

**19. ReactJS-HOL**

**1. Understanding need for isolation in testing**

* Isolation in testing means testing a component or module independently without relying on external dependencies like APIs or databases.
* It ensures that tests run quickly, consistently, and are not flaky due to network or third-party service issues.
* Isolated tests help pinpoint bugs within the module being tested, rather than in external systems.

**2. Understanding the concept of mocking**

* Mocking is the practice of replacing real dependencies with controlled, fake versions during testing.
* In the example, axios is mocked to simulate API responses without making actual HTTP requests.
* This lets you verify how your module behaves under various scenarios without depending on real external services.

**3. Using Jest for unit testing and mocking**

* Jest is a JavaScript testing framework commonly used with React.
* It provides built-in functions to write tests (describe, test), assertions (expect), and mocking capabilities (jest.mock).
* Using Jest, you can mock modules like axios and define expected return values to test your code’s logic in isolation.

**GitClient.js**

import axios from "axios";

class GitClient {

static async getRepositories(userName) {

const url = `https://api.github.com/users/${userName}/repos`;

const response = await axios.get(url);

return response.data.map((repo) => repo.name);

}

}

export default GitClient;

**App.js**

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

import GitClient from "./GitClient";

function App() {

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

useEffect(() => {

GitClient.getRepositories("TechieSyed").then((data) => {

setRepos(data);

});

}, []);

return (

<div className="App">

<h1>Git repositories of User - TechieSyed</h1>

<ul>

{repos.map((repo, index) => (

<li key={index}>{repo}</li>

))}

</ul>

</div>

);

}

export default App;

**GitClient.test.js**

import axios from "axios";

import GitClient from "./GitClient";

jest.mock("axios");

describe("Git Client Tests", () => {

test("should return repository names for techiesyed", async () => {

const dummyRepos = [

{ name: "repo-one" },

{ name: "repo-two" },

{ name: "repo-three" }

];

axios.get.mockResolvedValue({ data: dummyRepos });

const repos = await GitClient.getRepositories("TechieSyed");

expect(repos).toEqual(["repo-one", "repo-two", "repo-three"]);

});

});

**App.test.js**

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

import App from "./App";

test("renders GitClient header", () => {

render(<App />);

const header = screen.getByText(/Git repositories of User/i);

expect(header).toBeInTheDocument();

});

**package.json**

"scripts": {

"start": "react-scripts start",

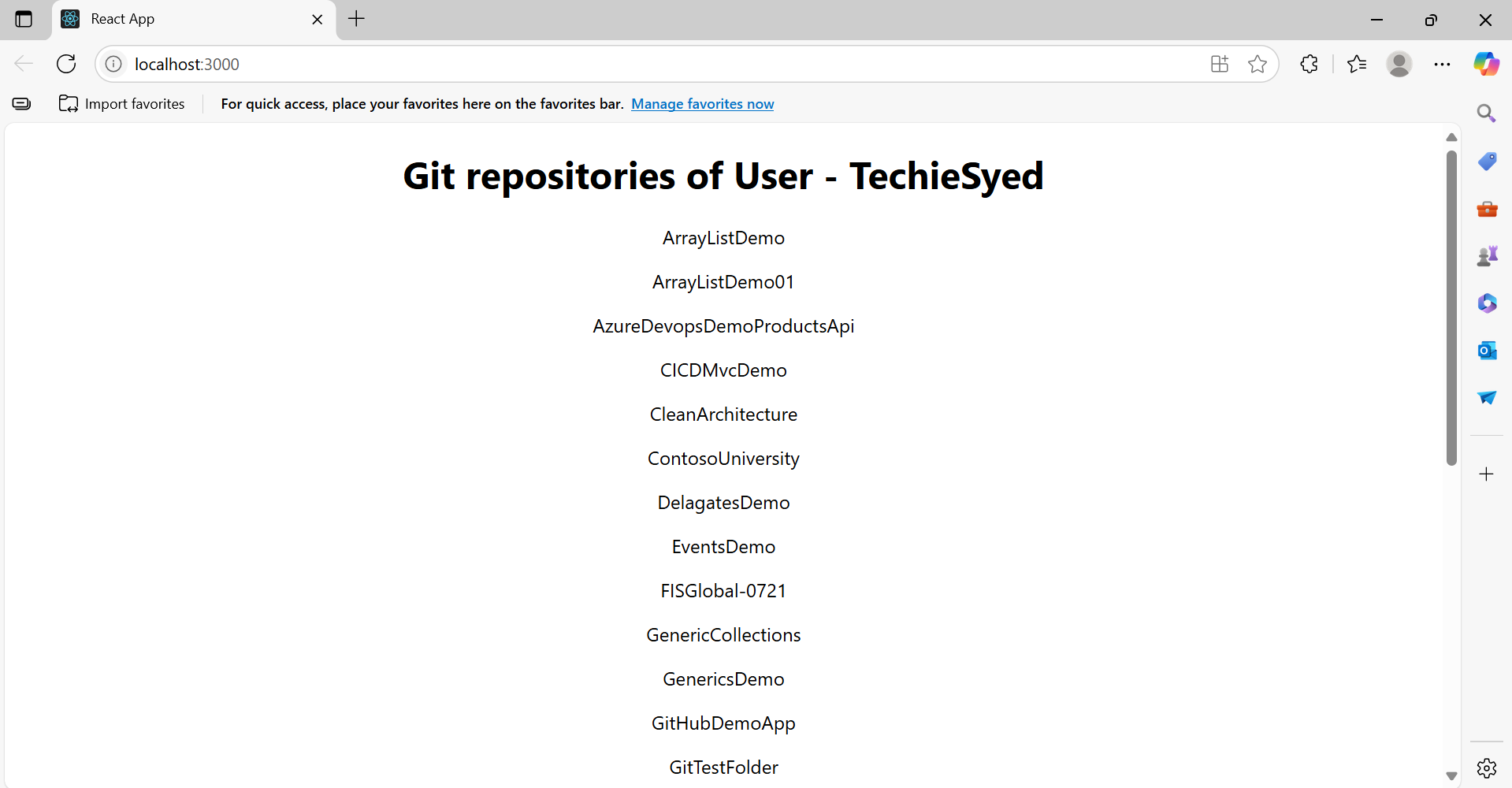
"build": "react-scripts build",

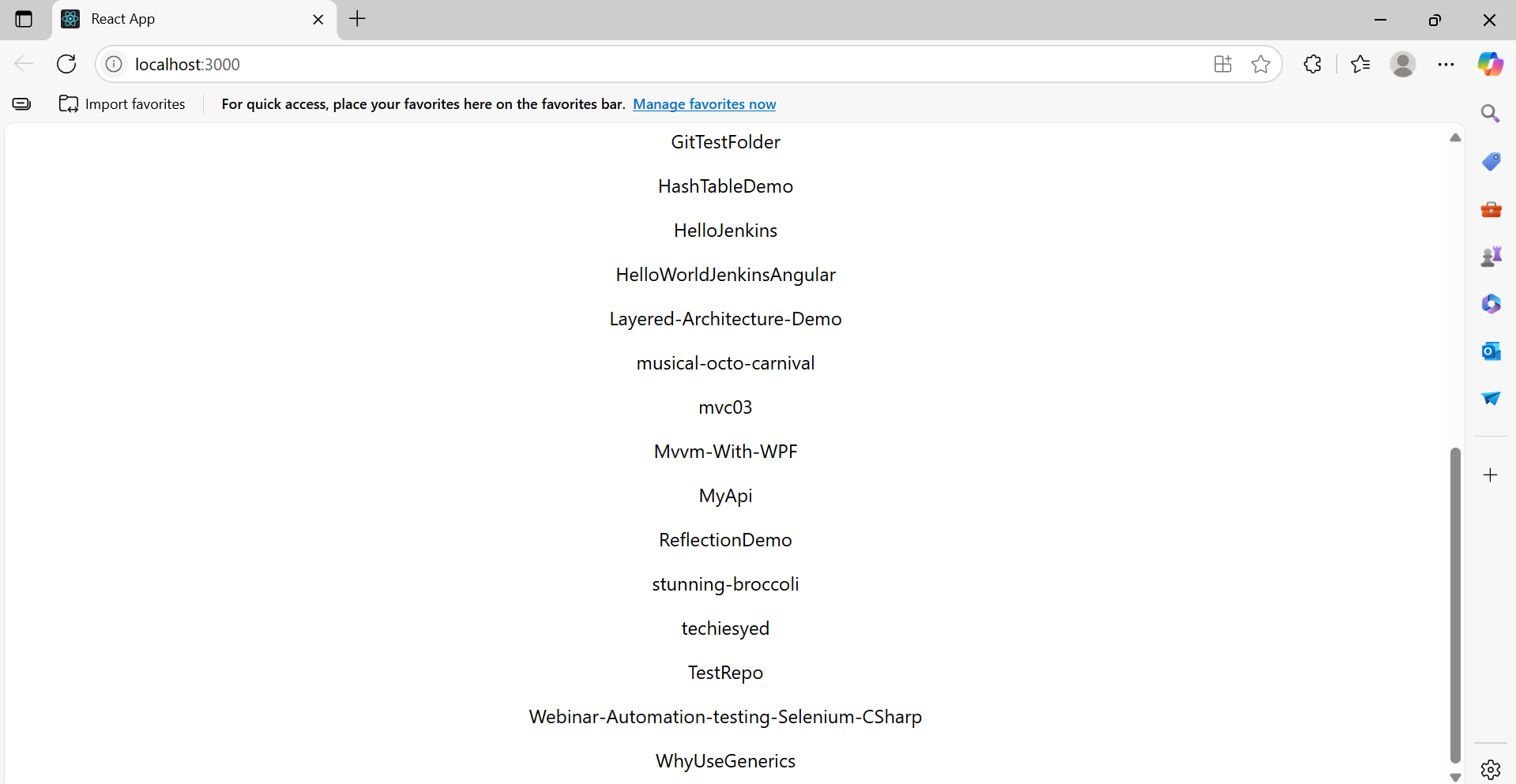
"test": "react-scripts test --transformIgnorePatterns \"node\_modules/(?!axios)/\"",

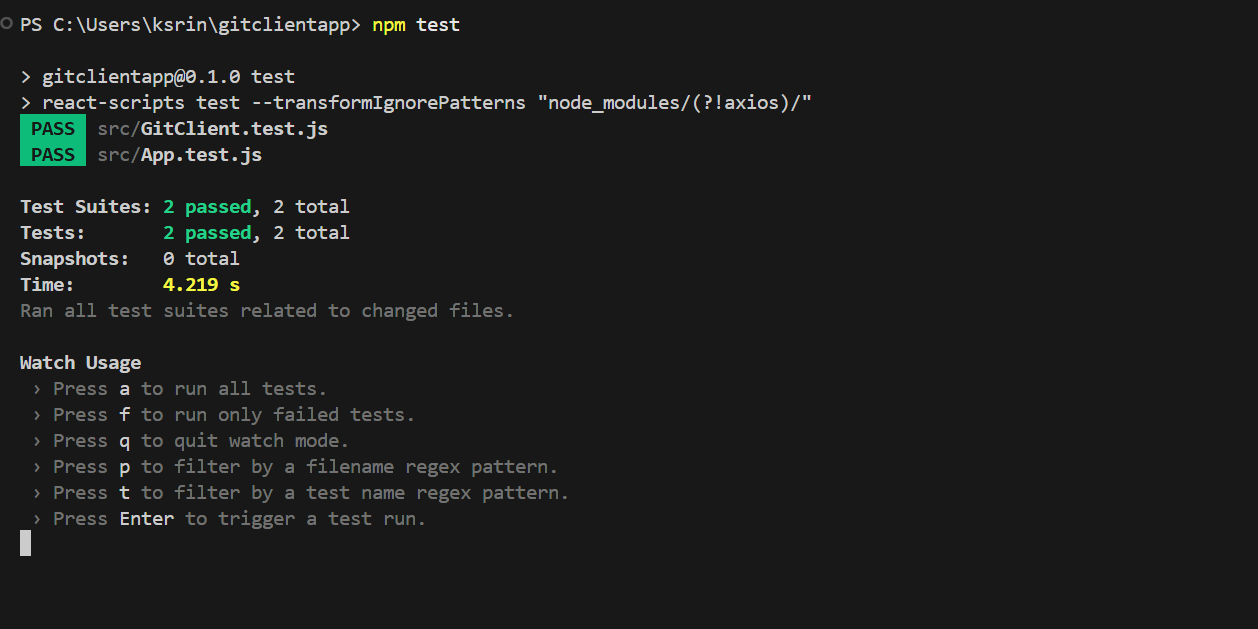
"eject": "react-scripts eject"

},

**Output:**

****

****

****