**1.ReactJS\_HOL**

**1. Define SPA and its Benefits**

SPA (Single-Page Application) is a web application that dynamically updates a single HTML page without reloading the whole page from the server.

Benefits:

* Faster page load after the first request
* Seamless user experience
* Reduces server load
* Uses AJAX and fetch APIs for background data

**2. Define React and Identify Its Working**

React is a JavaScript library developed by Facebook for building UI components. It is used to develop single-page applications by breaking UI into reusable components.

How React Works:

* Uses Virtual DOM for efficient rendering
* Detects changes and updates only the necessary parts of the real DOM
* Components manage their state and re-render when data changes

**3. Identify the Differences Between SPA and MPA**

| **Feature** | **SPA** | **MPA** |
| --- | --- | --- |
| Page Load | Single page loaded once | Multiple pages loaded |
| Speed | Faster after first load | Slower due to full reloads |
| Navigation | Handled via JavaScript (React) | Handled by server |
| SEO | Harder (needs SSR) | Easier |

**4. Pros and Cons of Single-Page Application**

**Pros:**

* Fast and responsive
* Better UX
* Efficient rendering using virtual DOM

**Cons:**

* SEO limitations
* Initial load can be large
* Needs more client-side coding

**5. Explain About React**

React is a component-based JavaScript library for building interactive UIs. It promotes a declarative programming style, using a virtual DOM to optimize performance and update the UI efficiently.

**6. Define Virtual DOM**

Virtual DOM is an in-memory representation of the real DOM. React uses it to:

* Detect changes via diffing algorithm
* Update only changed elements
* Avoid expensive full DOM manipulations

**7. Features of React**

* Component-based architecture
* One-way data binding
* Virtual DOM for faster updates
* JSX for writing HTML in JS
* Reusable components
* Hooks for managing state and lifecycle

**App.js**

import React from 'react';

function App() {

return (

<div>

<h1>Welcome to the first session of React</h1>

</div> ); }

export default App;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

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

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>React App</title>

</head>

<body>

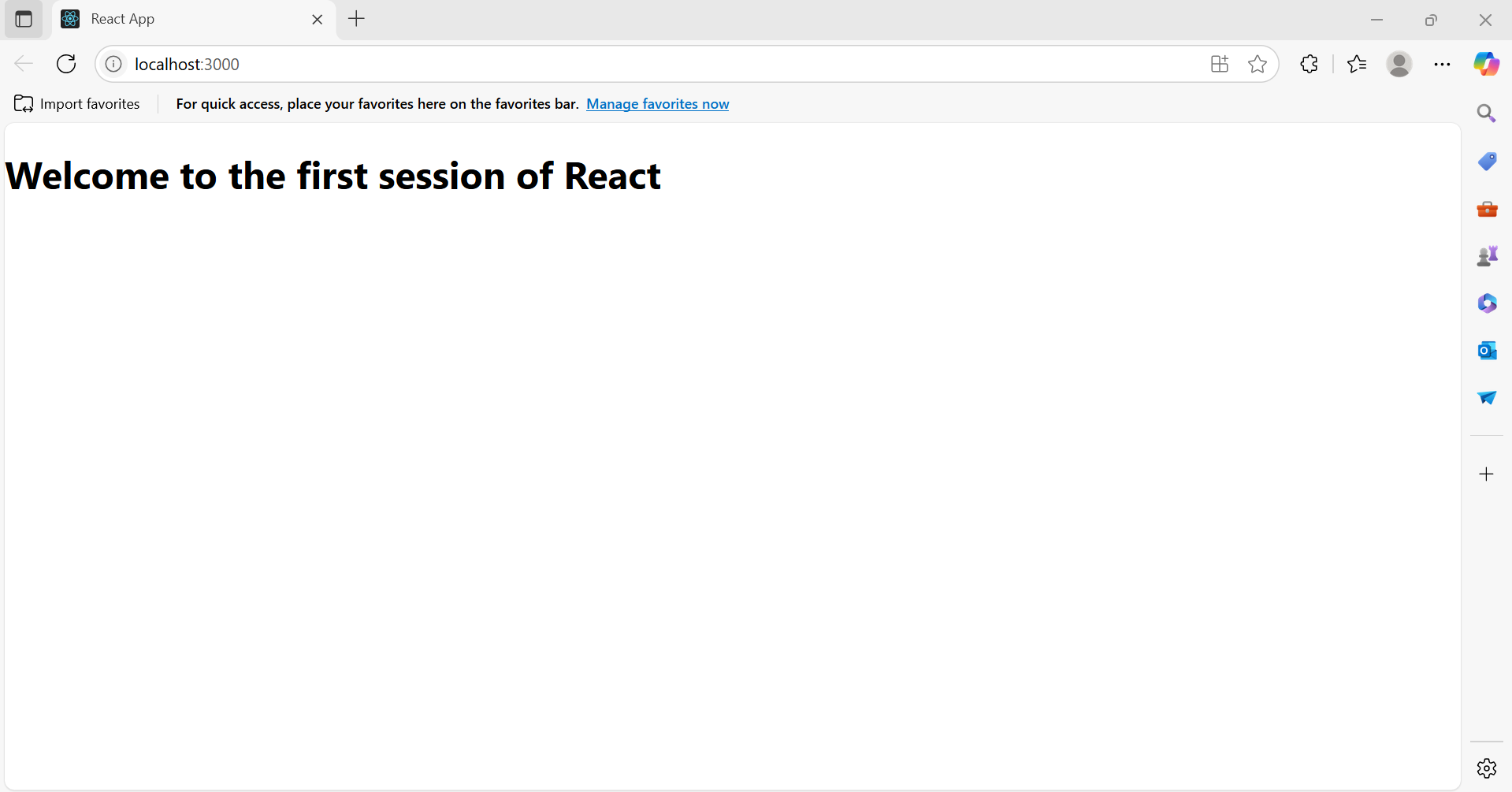
<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

</body>

</html>

**Output:**

****

**2.ReactJS\_HOL**

**1. Explain React Components**

React components are building blocks of any React application UI. Each component returns JSX that describes what should appear on the screen. They allow us to split the UI into independent, reusable pieces.

**2. Differences Between Components and JavaScript Functions**

| **Feature** | **JavaScript Function** | **React Component** |
| --- | --- | --- |
| Purpose | Performs a task | Returns JSX (UI) |
| Return Type | Any data | JSX (HTML-like code) |
| Naming Convention | camelCase | PascalCase (e.g., MyComp) |
| Usage | Called normally | Used like HTML tags <MyComp /> |
| React Features | Not supported | Can use useState, props, etc. |

**3. Types of Components**

* Function Component – A simple JS function returning JSX
* Class Component – Uses class keyword, has lifecycle methods and state

**4. Explain Class Component**

A class component is a component defined using ES6 class. It extends React.Component and must include a render() method that returns JSX.

class MyComponent extends React.Component {

render() {

return <h1>Hello from Class Component</h1>;

}

}

**5. Explain Function Component**

A function component is a simple JavaScript function that returns JSX.

function MyComponent() {

return <h1>Hello from Function Component</h1>;

}

**6. Define Component Constructor**

The constructor() method in class components is used to initialize state and bind event handlers.

constructor(props) {

super(props);

this.state = { message: "Hello" };

}

**7. Define render() Function**

The render() method in a class component returns JSX and describes what UI should be displayed.

**App.js**

import React from 'react';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

return (

<div>

<h1>Student Management Portal</h1>

<Home />

<About />

<Contact />

</div>

);

}

export default App;

**Home.js**

import React, { Component } from 'react';

class Home extends Component {

render() {

return (

<div>

<h2>Welcome to the Home page of Student Management Portal</h2>

</div>

);

}

}

export default Home;

**About.js**

import React, { Component } from 'react';

class About extends Component {

render() {

return (

<div>

<h2>Welcome to the About page of the Student Management Portal</h2>

</div>

);

}

}

export default About;

**Contact.js**

import React, { Component } from 'react';

class Contact extends Component {

render() {

return (

<div>

<h2>Welcome to the Contact page of the Student Management Portal</h2>

</div>

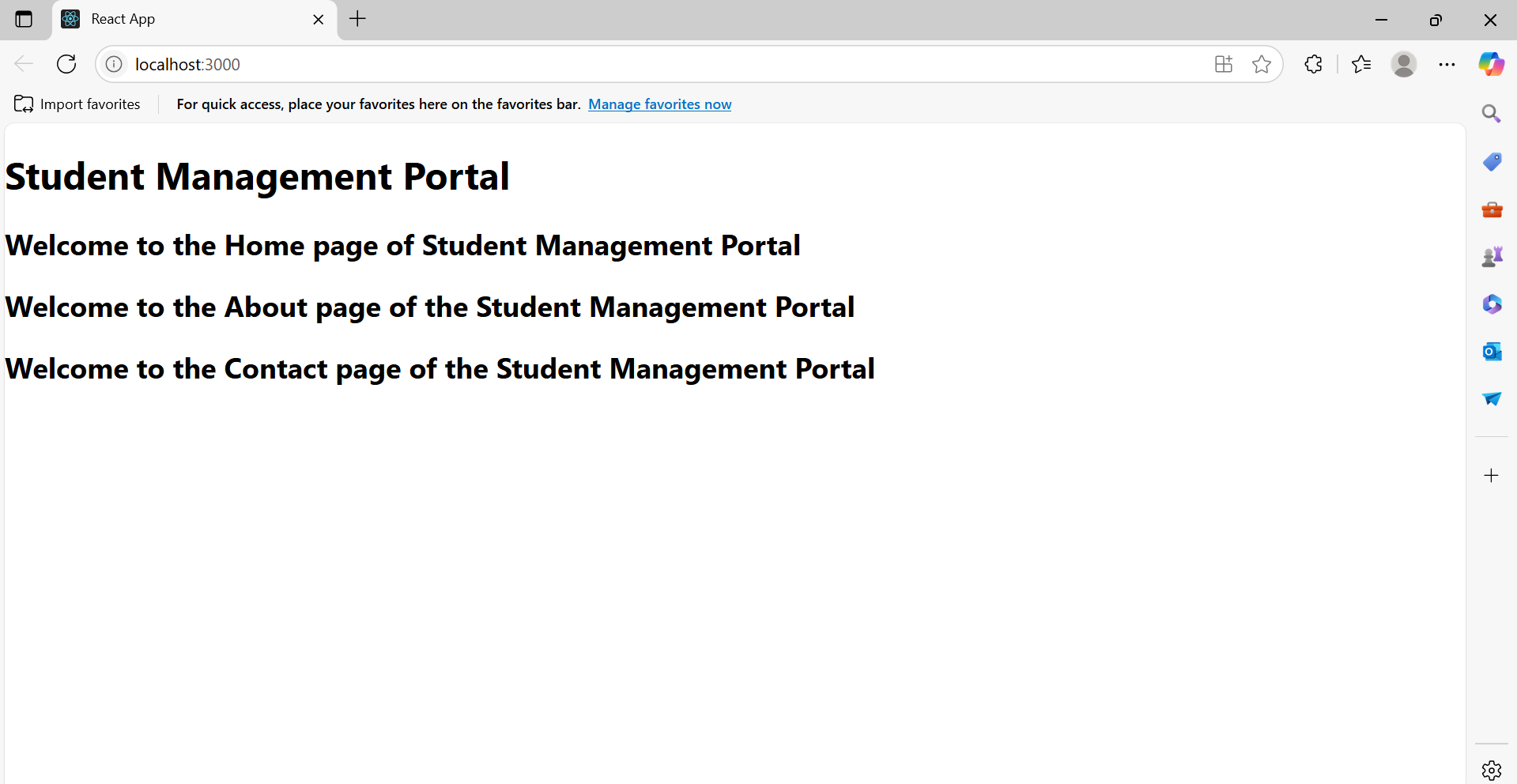
);

}

}

export default Contact;

**Output:**

****

**3.ReactJS\_HOL**

**CalculateScore.js**

import '../Stylesheets/mystyle.css';

const percentToDecimal = (decimal) => {

return (decimal.toFixed(2) + '%');

};

const calcScore = (total, goal) => {

return percentToDecimal(total / goal);

};

export const CalculateScore = ({ Name, School, total, goal }) => (

<div className="formatstyle">

<h1><font color="Brown">Student Details:</font></h1>

<div className="Name">

<b><span> Name: </span></b>

<span>{Name}</span>

</div>

<div className="School">

<b><span> School: </span></b>

<span>{School}</span>

</div>

<div className="Total">

<b><span>Total: </span></b>

<span>{total}</span> Marks

</div>

<div className="Score">

<b>Score: </b>

<span>{calcScore(total, goal)}</span>

</div>

</div>

);

**mystyle.css**

.Name {

font-weight: 300;

color: blue;

}

.School {

color: crimson;

}

.Total {

color: darkmagenta;

}

.formatstyle {

text-align: center;

font-size: large;

}

.Score {

color: forestgreen;

}

**App.js**

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

function App() {

return (

<div>

<CalculateScore

Name={"Steeve"}

School={"DNV Public School"}

total={284}

goal={3}

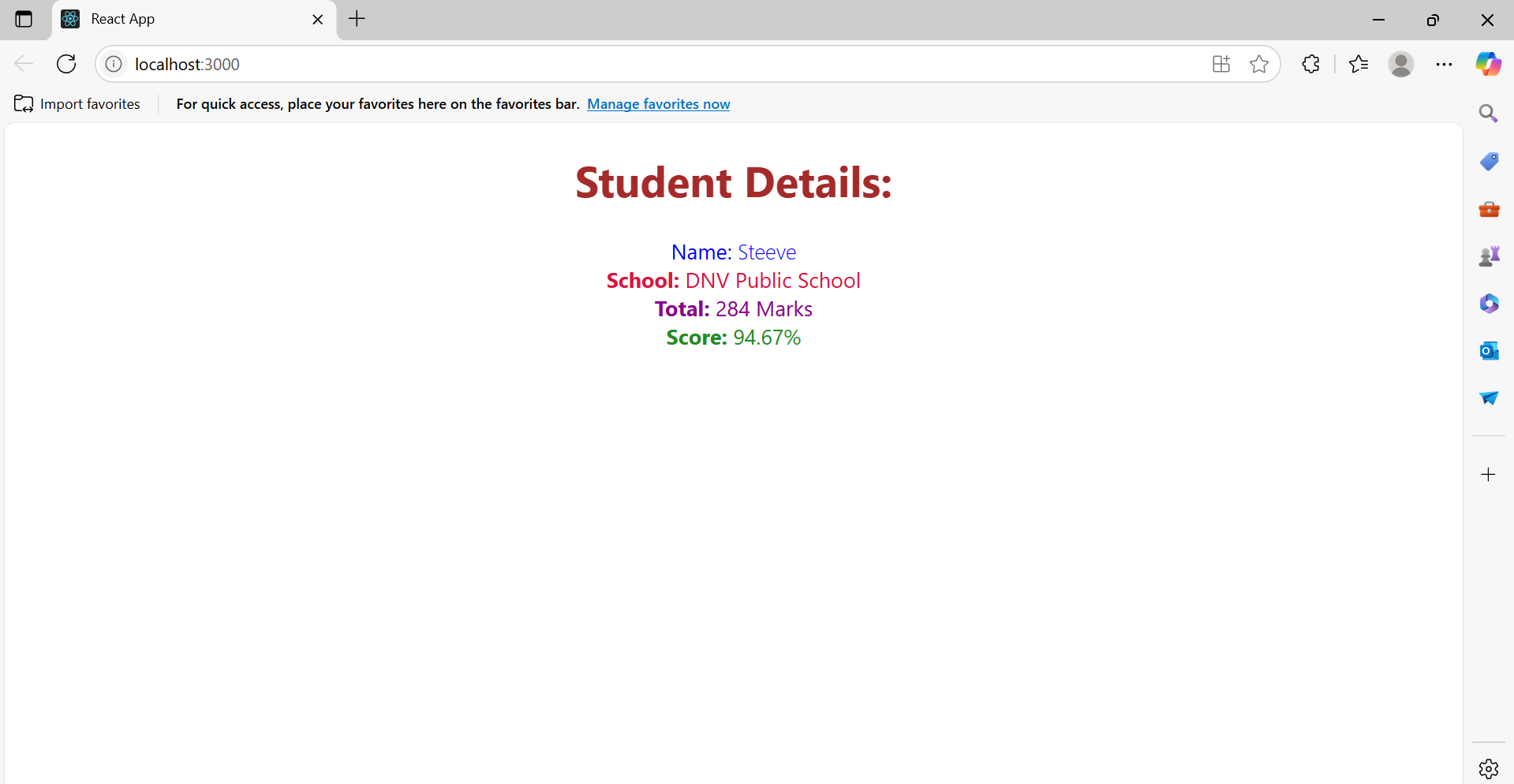
/>

</div>

);

}

export default App;

**Output:**

**4.ReactJS\_HOL**

**1. Explain the Need and Benefits of Component Life Cycle**

**Need:**The component life cycle in React helps manage the creation, updating, and destruction of components. It provides special methods (life cycle hooks) that allow developers to run code at specific stages of a component’s existence.

**Benefits:**

* Enables data fetching, DOM manipulation, and event handling at the right time.
* Helps in optimizing performance by controlling updates.
* Allows handling of errors gracefully.
* Improves code organization and reusability.

**2. Identify Various Life Cycle Hook Methods**

React class components have three main phases:

A. Mounting (component is being created and inserted into the DOM):

* constructor()
* static getDerivedStateFromProps()
* render()
* componentDidMount()

B. Updating (component is being re-rendered due to state or props change):

* static getDerivedStateFromProps()
* shouldComponentUpdate()
* render()
* getSnapshotBeforeUpdate()
* componentDidUpdate()

C. Unmounting (component is being removed from the DOM):

* componentWillUnmount()

D. Error Handling:

* componentDidCatch()
* static getDerivedStateFromError()

**3. List the Sequence of Steps in Rendering a Component**

**Mounting Phase:**

1. constructor()
2. static getDerivedStateFromProps()
3. render()
4. componentDidMount()

**Updating Phase:**

1. static getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

**Unmounting Phase:**

1. componentWillUnmount()

**Post.js**

class Post {

constructor(id, title, body) {

this.id = id;

this.title = title;

this.body = body;

}

}

export default Post;

**Posts.js**

import React from 'react';

import Post from './Post';

class Posts extends React.Component {

constructor(props) {

super(props);

this.state = {

posts: [],

hasError: false

};

}

loadPosts() {

fetch('https://jsonplaceholder.typicode.com/posts')

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

.then(data => {

const postList = data.map(p => new Post(p.id, p.title, p.body));

this.setState({ posts: postList });

})

.catch(error => {

console.error("Error fetching posts:", error);

this.setState({ hasError: true });

});

}

componentDidMount() {

this.loadPosts();

}

componentDidCatch(error, info) {

alert("An error occurred: " + error);

console.error("Error info:", info);

}

render() {

if (this.state.hasError) {

return <h2>Something went wrong!</h2>;

}

return (

<div>

<h1>Blog Posts</h1>

{this.state.posts.map(post => (

<div key={post.id}>

<h2>{post.title}</h2>

<p>{post.body}</p>

</div>

))}

</div>

);

}

}

export default Posts;

**App.js**

import React from 'react';

import Posts from './Posts';

function App() {

return (

<div className="App">

<Posts />

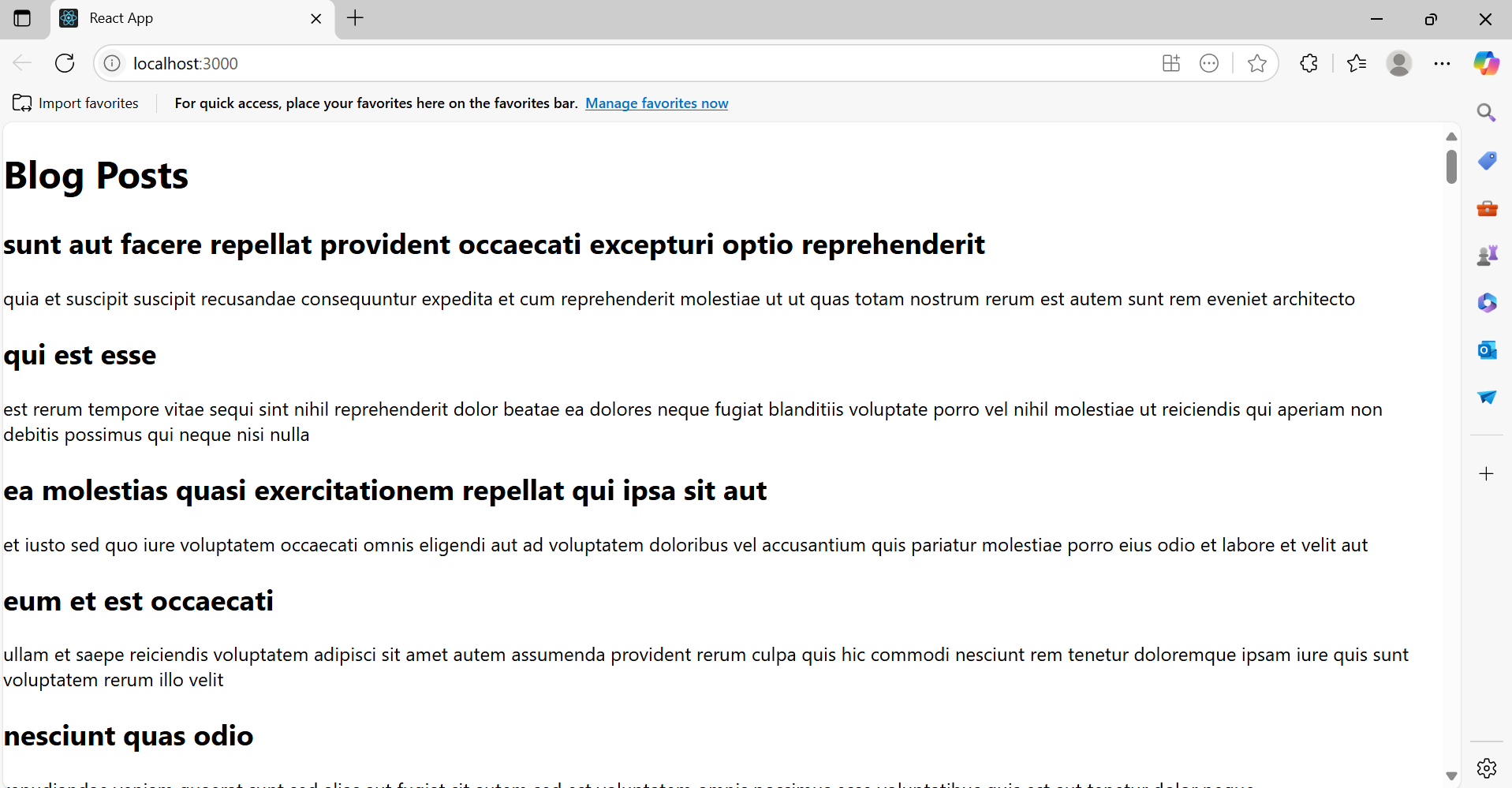
</div>

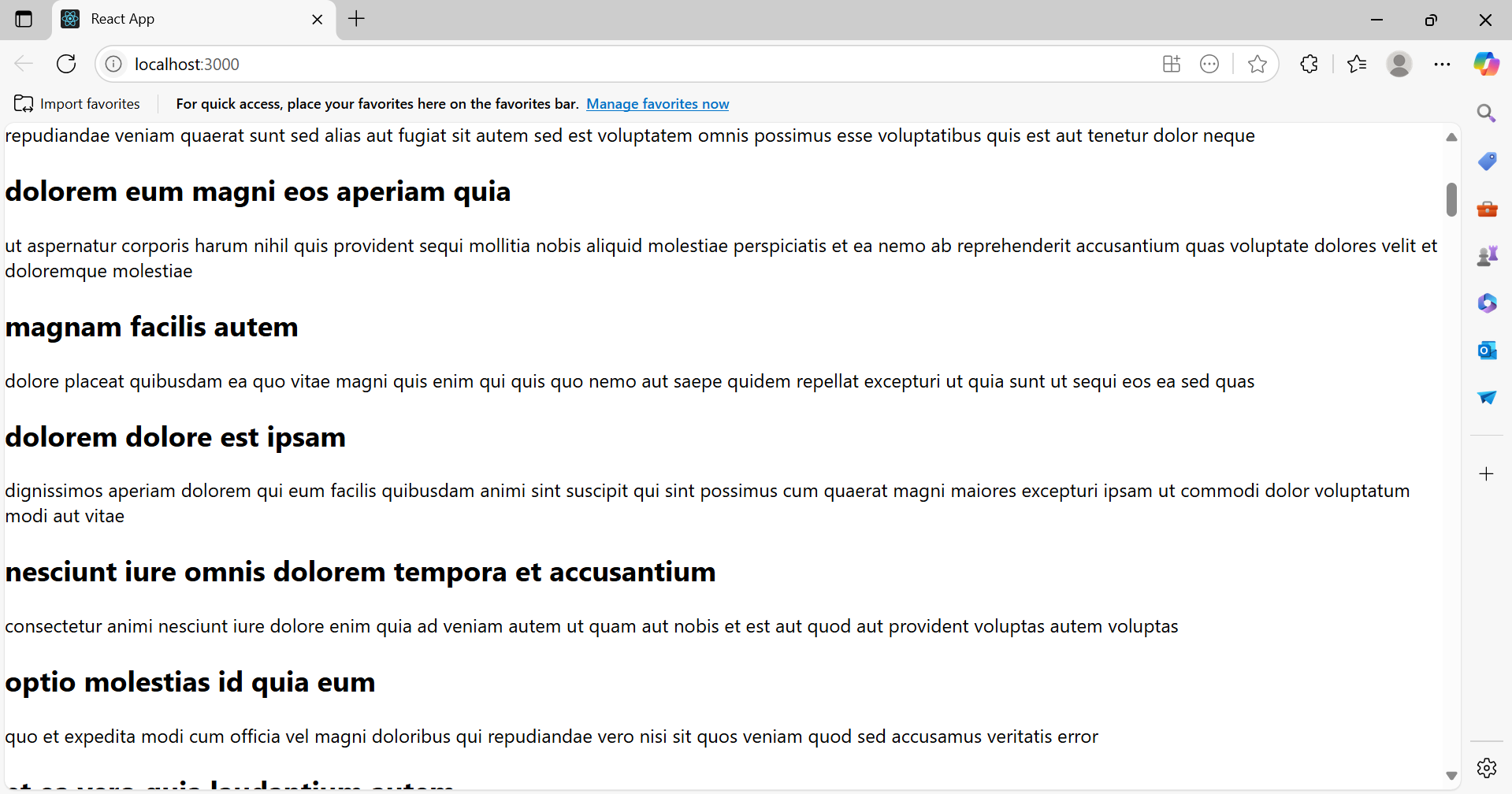
);

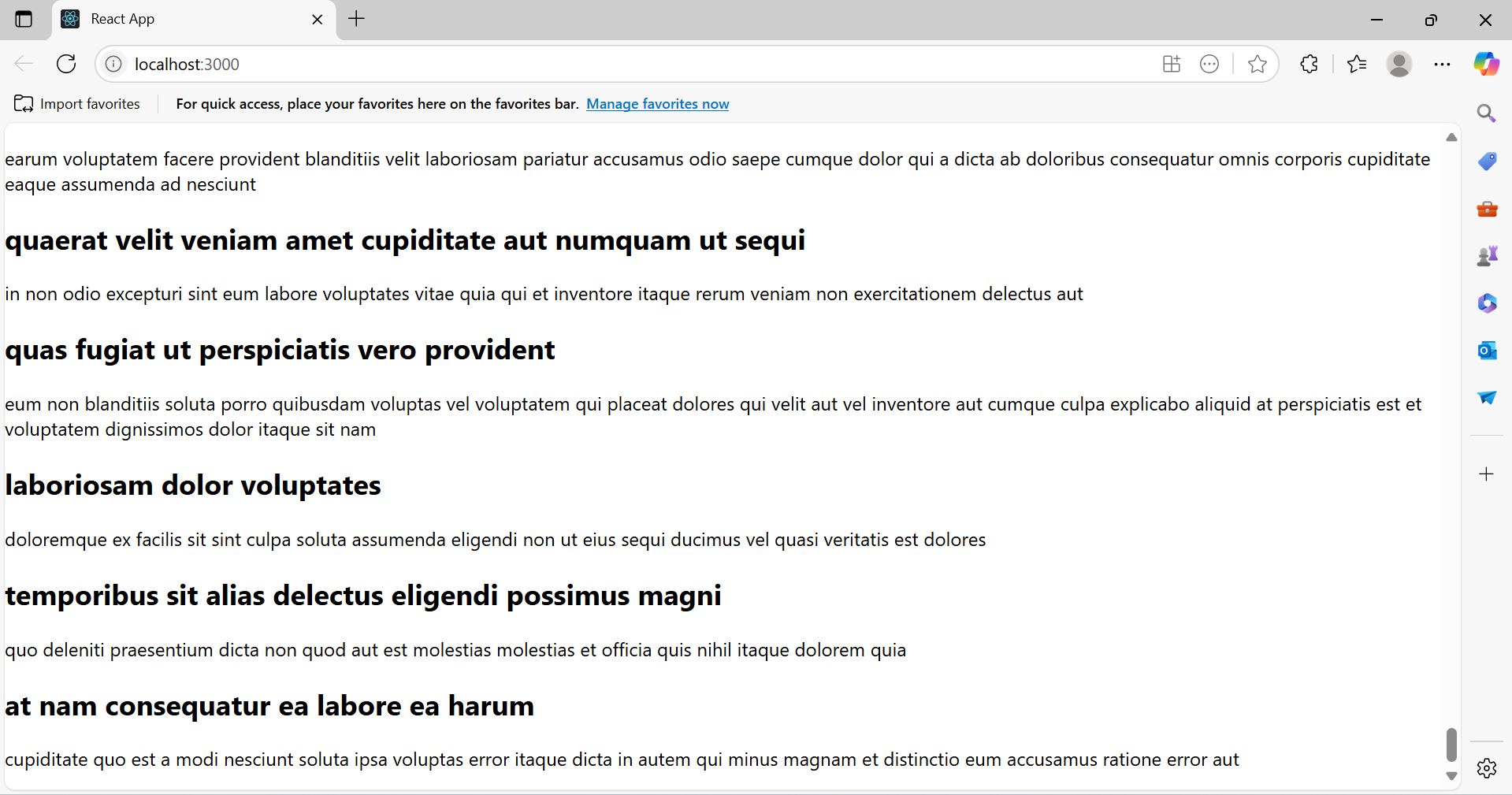
}

export default App;

**Output:**

****

****

****

**5.ReactJS\_HOL**

**Understanding the need for styling React components**

* Styling helps improve UI readability, usability, and branding consistency.
* In React, styling keeps components modular and scoped (especially with CSS Modules).

**Working with CSS Module and inline styles**

* CSS Modules: Allow scoped CSS (no conflicts between styles).
* Inline styles: Useful for conditional and dynamic styling directly in JSX.

**CohortDetails.jsx**

import React from 'react';

import styles from './CohortDetails.module.css';

function CohortDetails({ cohort }) {

const isOngoing = cohort.status.toLowerCase() === 'ongoing';

const headingStyle = {

color: isOngoing ? 'green' : 'blue'

};

return (

<div className={styles.box}>

<h3 style={headingStyle}>{cohort.name}</h3>

<dl>

<dt>Started On</dt>

<dd>{cohort.startedOn}</dd>

<dt>Current Status</dt>

<dd>{cohort.status}</dd>

<dt>Coach</dt>

<dd>{cohort.coach}</dd>

<dt>Trainer</dt>

<dd>{cohort.trainer}</dd>

</dl>

</div>

);

}

export default CohortDetails;

**CohortDetails.module.css**

.box {

width: 300px;

display: inline-block;

margin: 10px;

padding: 10px 20px;

border: 1px solid black;

border-radius: 10px;

}

dt {

font-weight: 500;

}

**App.js**

import React from 'react';

import CohortDetails from './components/CohortDetails';

const cohorts = [

{

name: 'INTADMDF10 - .NET FSD',

startedOn: '22-Feb-2022',

status: 'Scheduled',

coach: 'Aathma',

trainer: 'Jojo Jose'

},

{

name: 'ADM21JF014 - Java FSD',

startedOn: '10-Sep-2021',

status: 'Ongoing',

coach: 'Apoorv',

trainer: 'Elisa Smith'

},

{

name: 'CDBJF21025 - Java FSD',

startedOn: '24-Dec-2021',

status: 'Ongoing',

coach: 'Aathma',

trainer: 'John Doe'

}

];

function App() {

return (

<div>

<h2>Cohorts Details</h2>

{cohorts.map((cohort, index) => (

<CohortDetails key={index} cohort={cohort} />

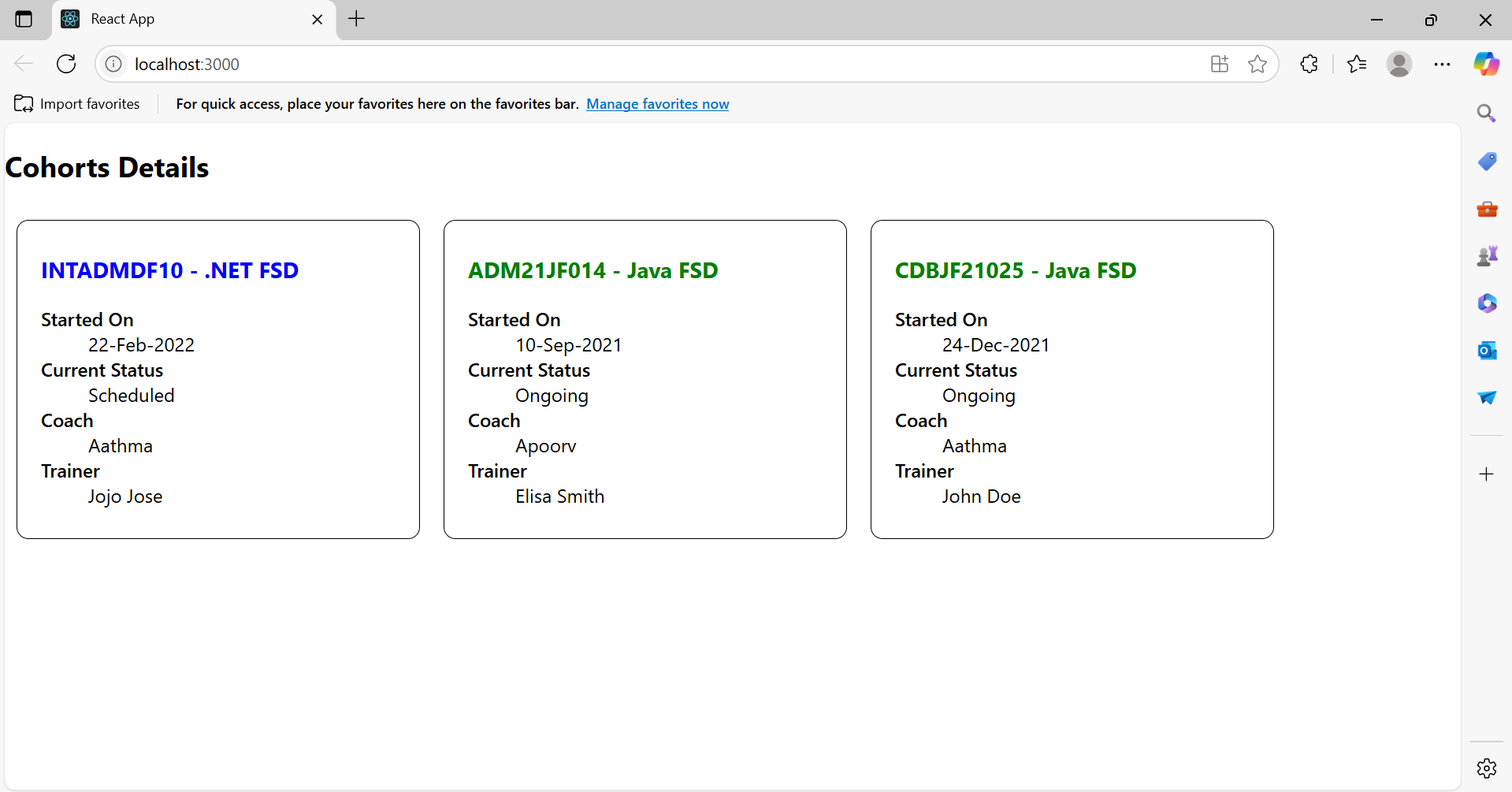
))}

</div>

);

}

export default App;

**Output:**

**6.ReactJS\_HOL**

**1. Need and Benefits of React Router**

* Enables building Single Page Applications (SPAs).
* Users can navigate without full page reload.
* Helps with route-based rendering of components.
* Improves performance and user experience.

**2. Components in React Router**

* BrowserRouter: Wraps the entire app to enable routing.
* Routes: Defines all route mappings.
* Route: Maps path to a component.
* Link: Enables navigation without page reload.
* useParams: Extracts route parameters (e.g., trainer ID).

**3. Types of Router Components**

* Routing Container: BrowserRouter
* Routes Declaration: Routes, Route
* Navigation: Link, NavLink
* Hooks: useParams, useNavigate, useLocation

**4. Parameter Passing via URL**

* Parameters like /trainers/:id allow dynamic routing.
* Use useParams() to extract them inside a component.

**Trainer.js**

class Trainer {

constructor(trainerId, name, email, phone, technology, skills) {

this.trainerId = trainerId;

this.name = name;

this.email = email;

this.phone = phone;

this.technology = technology;

this.skills = skills;

}

}

export default Trainer;

**TrainersMock.js**

import Trainer from "./trainer";

const trainersMock = [

new Trainer('t-syed8', 'Syed Khaleelullah', 'khaleelullah@cognizant.com', '97676516962', '.NET', ['C#', 'SQL Server', 'React', '.NET Core']),

new Trainer('t-jojo', 'Jojo Jose', 'jojo@cognizant.com', '9897199231', 'Java', ['Java', 'JSP', 'Angular', 'Spring']),

new Trainer('t-elisa', 'Elisa Jones', 'elisa@cognizant.com', '9871212235', 'Python', ['Python', 'Django', 'Angular']),

];

export default trainersMock;

**TrainerList.js**

import React from 'react';

import { Link } from 'react-router-dom';

function TrainerList({ trainers }) {

return (

<div>

<h3>Trainers List</h3>

<ul>

{trainers.map((trainer) => (

<li key={trainer.trainerId}>

<Link to={`/trainers/${trainer.trainerId}`}>{trainer.name}</Link>

</li>

))}

</ul>

</div>

);

}

export default TrainerList;

**Home.js**

import React from 'react';

function Home() {

return <h3>Welcome to My Academy trainers page</h3>;

}

export default Home;

**App.js**

import React from 'react';

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

import Home from './Home';

import TrainerList from './TrainerList';

import TrainerDetails from './TrainerDetails';

import trainersMock from './TrainersMock';

function App() {

return (

<BrowserRouter>

<div>

<h2>My Academy Trainers App</h2>

<nav>

<Link to="/">Home</Link> | <Link to="/trainers">Show Trainers</Link>

</nav>

<Routes>

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

<Route path="/trainers" element={<TrainerList trainers={trainersMock} />} />

<Route path="/trainers/:id" element={<TrainerDetails />} />

</Routes>

</div>

</BrowserRouter>

);

}

export default App;

**TrainerDetails.js**

import React from 'react';

import { useParams } from 'react-router-dom';

import trainersMock from './TrainersMock';

function TrainerDetails() {

const { id } = useParams();

const trainer = trainersMock.find(t => t.trainerId === id);

if (!trainer) {

return <p>Trainer not found!</p>;

}

return (

<div>

<h3>Trainers Details</h3>

<strong>{trainer.name} ({trainer.technology})</strong>

<p>{trainer.email}</p>

<p>{trainer.phone}</p>

<ul>

{trainer.skills.map((skill, i) => (

<li key={i}>{skill}</li>

))}

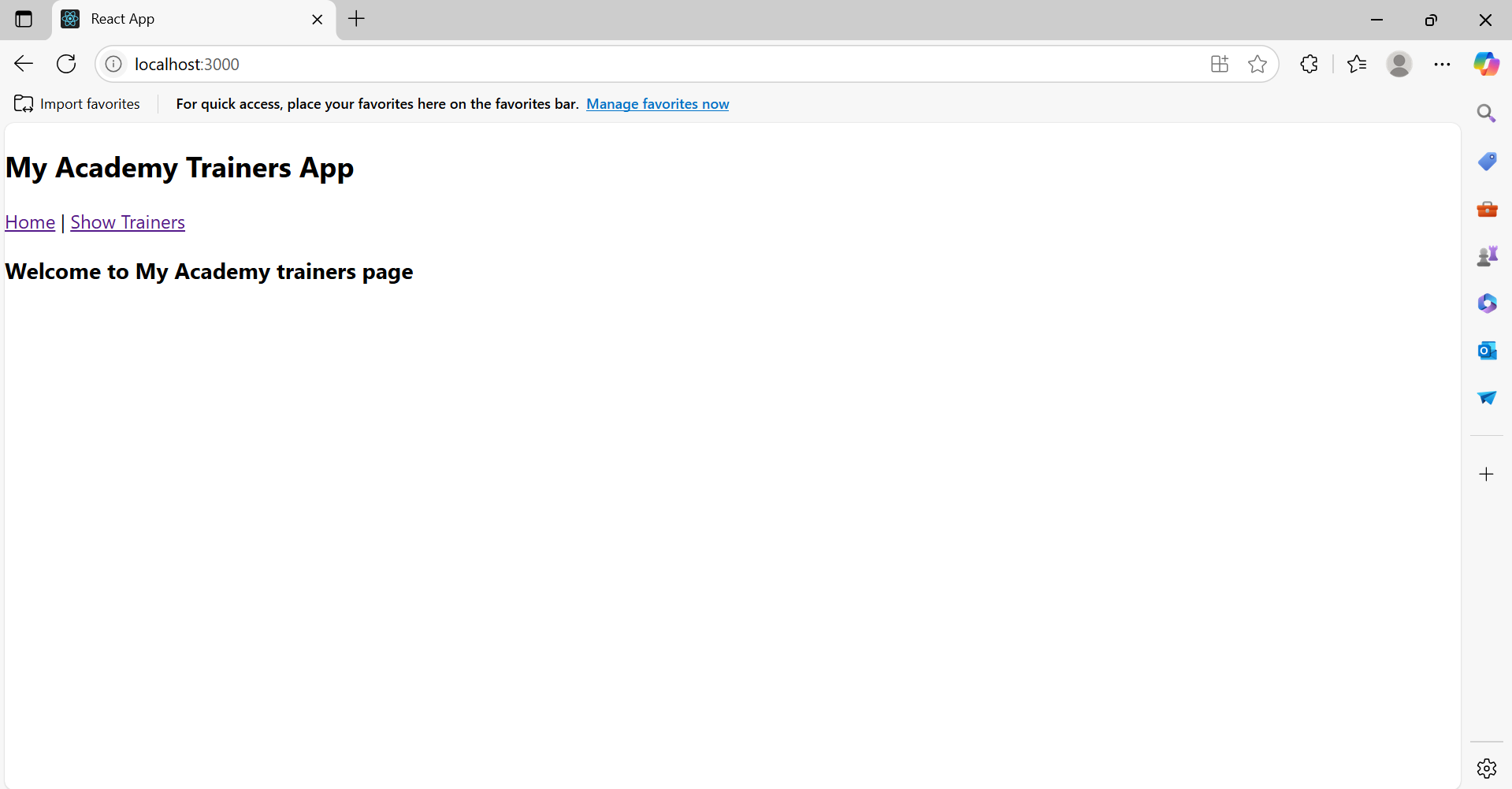
</ul>

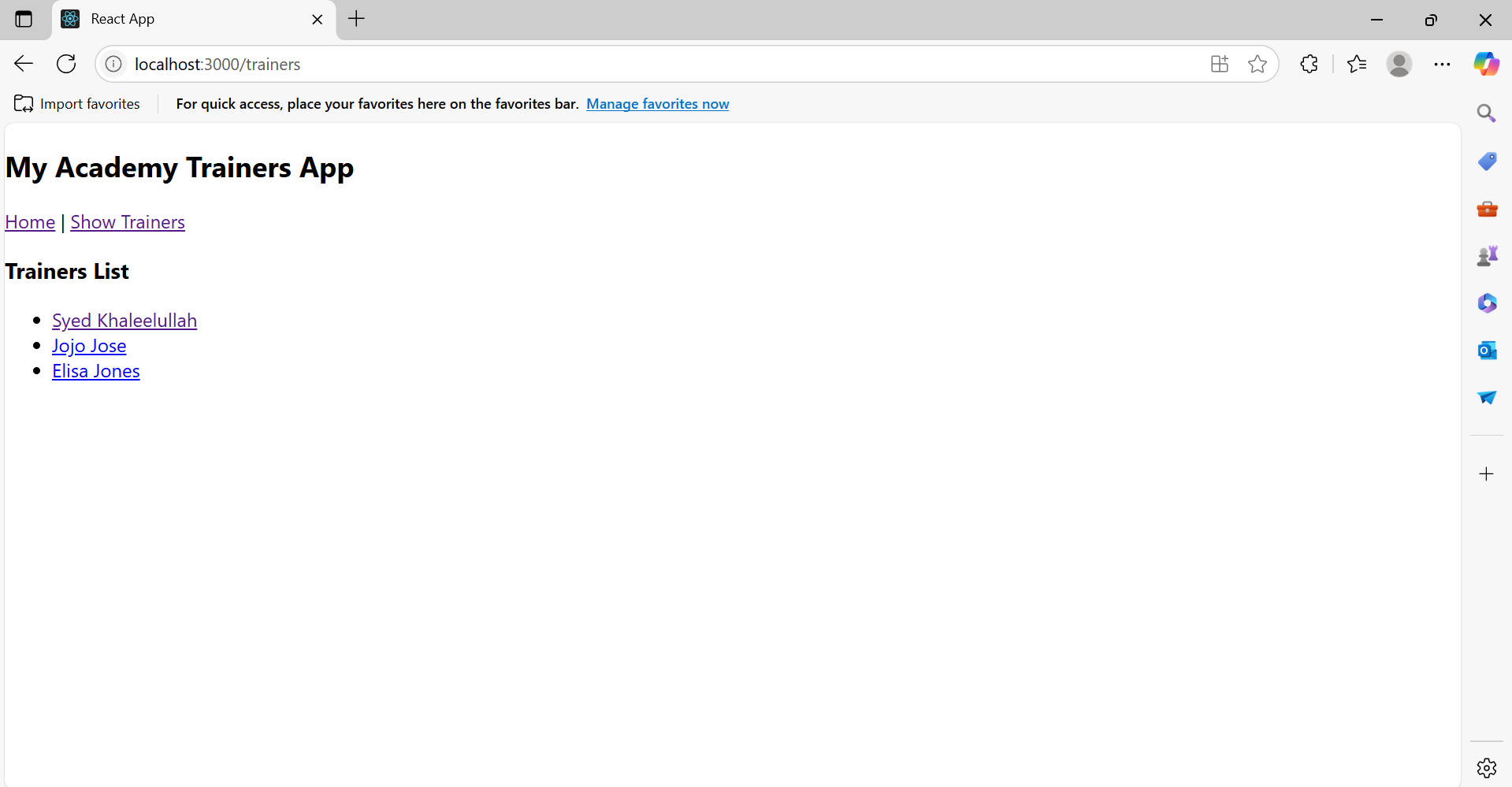
</div>

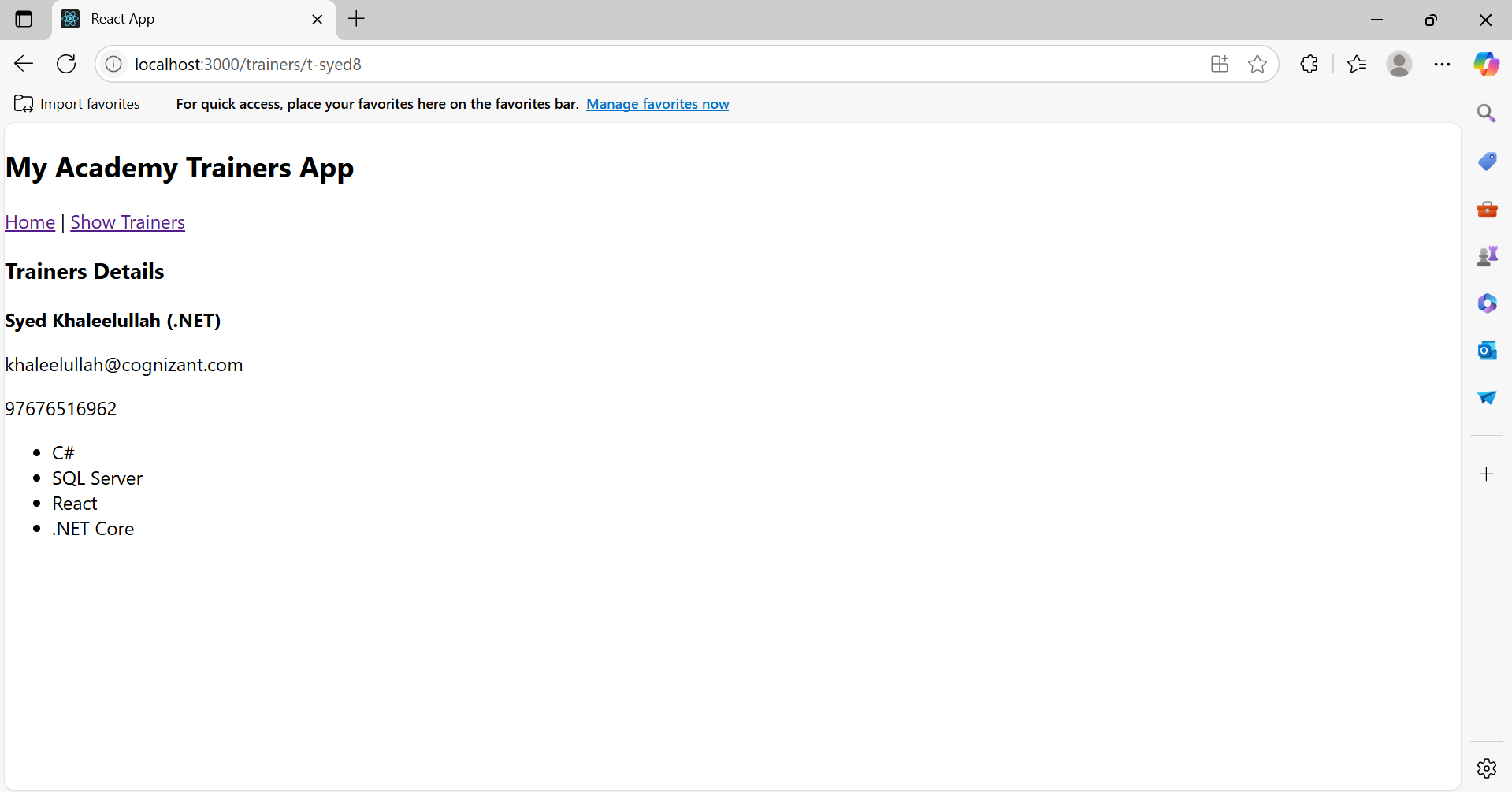
);

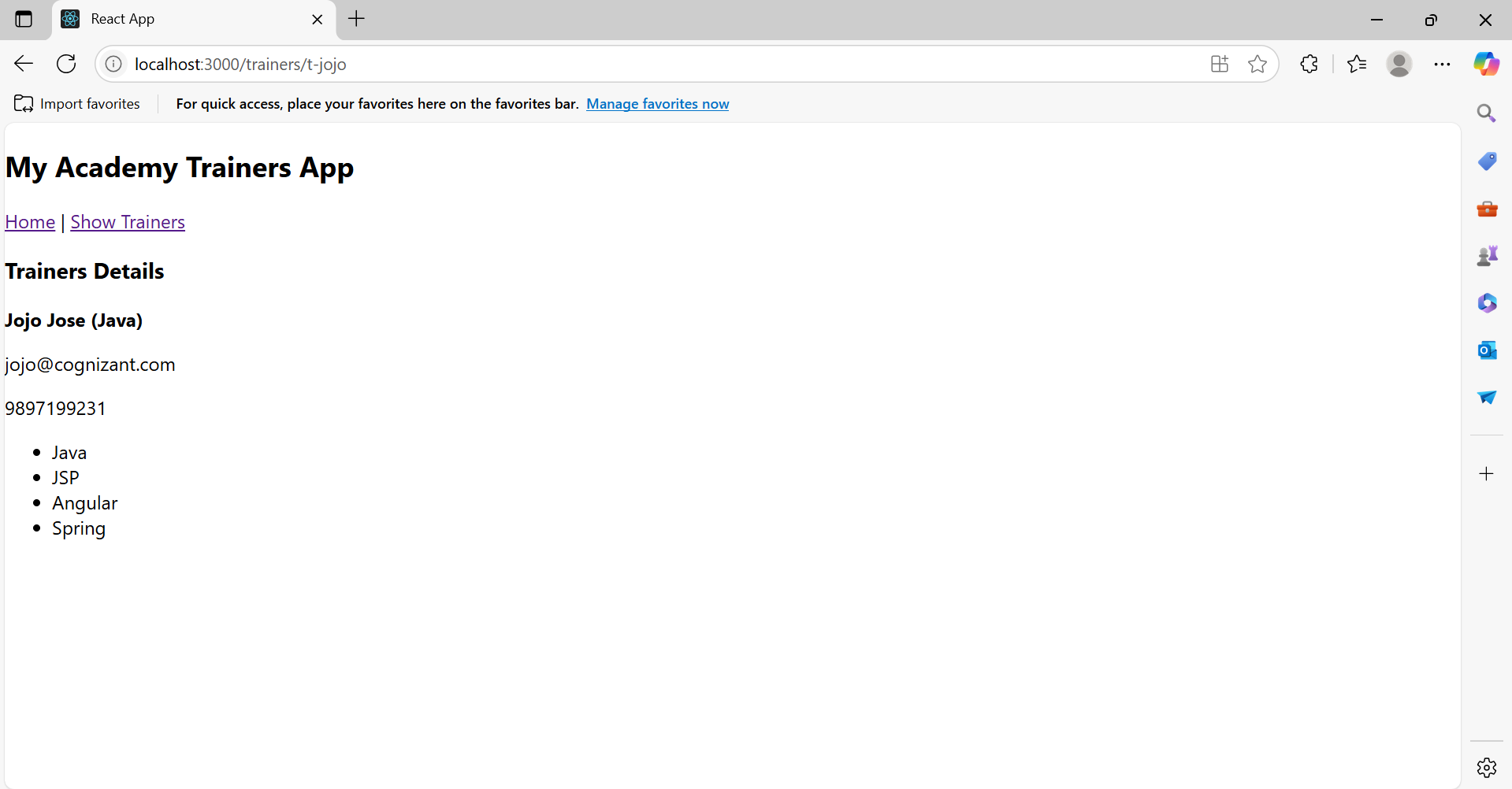
}

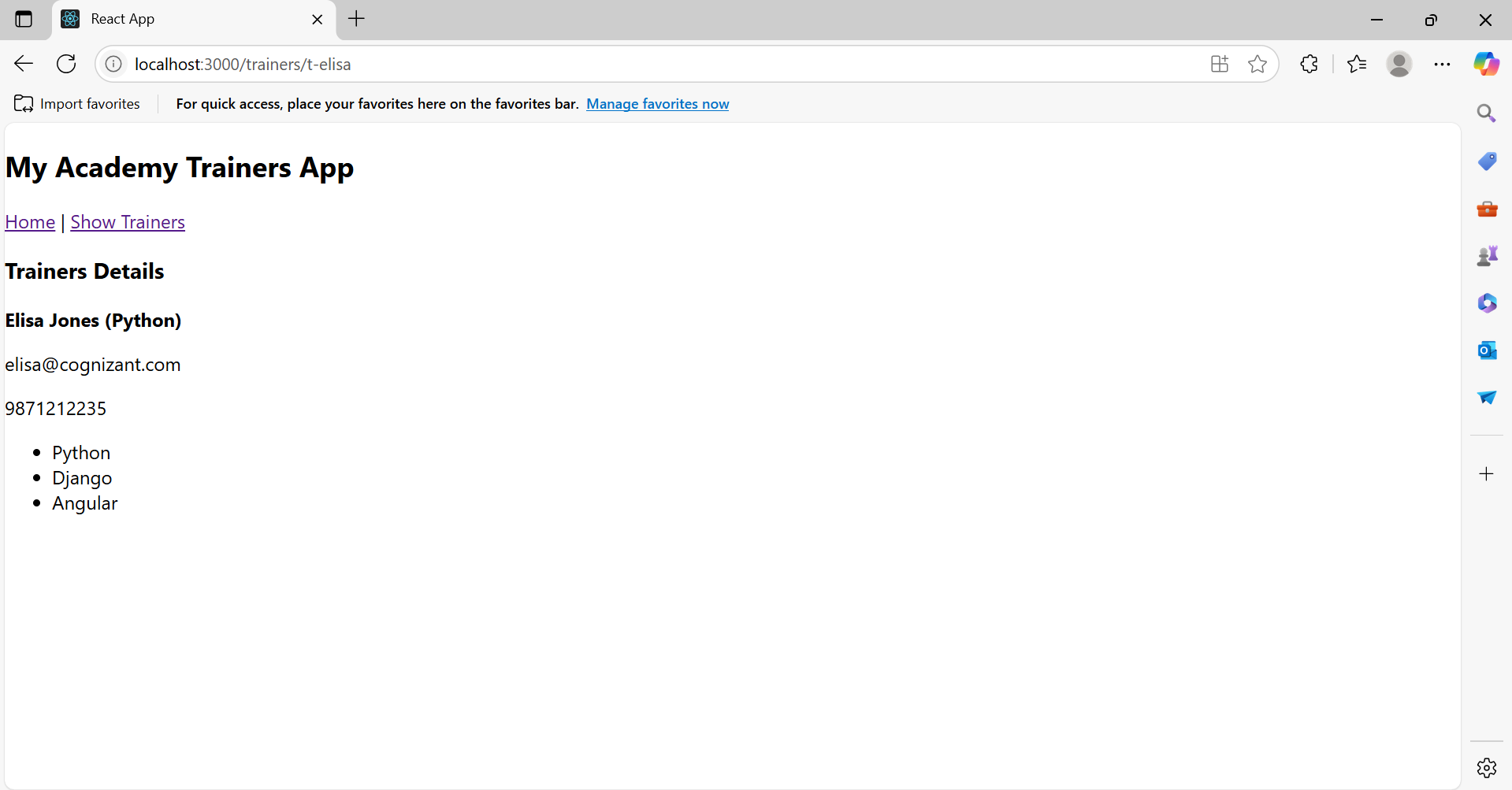
export default TrainerDetails;

**Output:**

****

****

****

****

**7.ReactJS\_HOL**

**1. Define Props**

Props (short for properties) are read-only attributes used in React to pass data from a parent component to a child component. They help in making components reusable and dynamic.

Example:

<Cart item={CartInfo} />

Here, CartInfo is passed as a prop to the Cart component.

**2. Explain Default Props**

Default Props allow you to assign default values to props in case none are provided.

Example:

Cart.defaultProps = {

item: [{ itemname: "Default Item", price: 0 }]

};

**3. Identify the Differences Between State and Props**

| **Props** | **State** |
| --- | --- |
| Passed from parent to child | Managed within the component |
| Immutable (read-only) | Mutable (can change) |
| Makes components reusable | Stores local data |
| Functional for displaying data | Used for interactivity |

**4. Explain ReactDOM.render()**

ReactDOM.render() is the method used to render a React element (component) into the DOM.

Syntax:

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

It tells React to display the component <App /> inside the HTML element with the id "root".

**Cart.js**

import React, { Component } from 'react';

export class Cart extends Component {

render() {

return (

<table border="1" align="center">

<thead>

<tr>

<th>Name</th>

<th>Price</th>

</tr>

</thead>

<tbody>

{this.props.item.map((item, index) => (

<tr key={index}>

<td>{item.itemname}</td>

<td>{item.price}</td>

</tr>

))}

</tbody>

</table>

);

}

}  
  
**OnlineShopping.js**

import React, { Component } from 'react';

import { Cart } from './Cart';

export class OnlineShopping extends Component {

render() {

const CartInfo = [

{ itemname: "Laptop", price: 80000 },

{ itemname: "TV", price: 120000 },

{ itemname: "Washing Machine", price: 50000 },

{ itemname: "Mobile", price: 30000 },

{ itemname: "Fridge", price: 70000 }

];

return (

<div className="mydiv" style={{ textAlign: 'center' }}>

<h1 style={{ color: 'green' }}>Items Ordered :</h1>

<Cart item={CartInfo} />

</div>

);

}

}

**App.js**

import React from 'react';

import './App.css';

import { OnlineShopping } from './OnlineShopping';

function App() {

return (

<div>

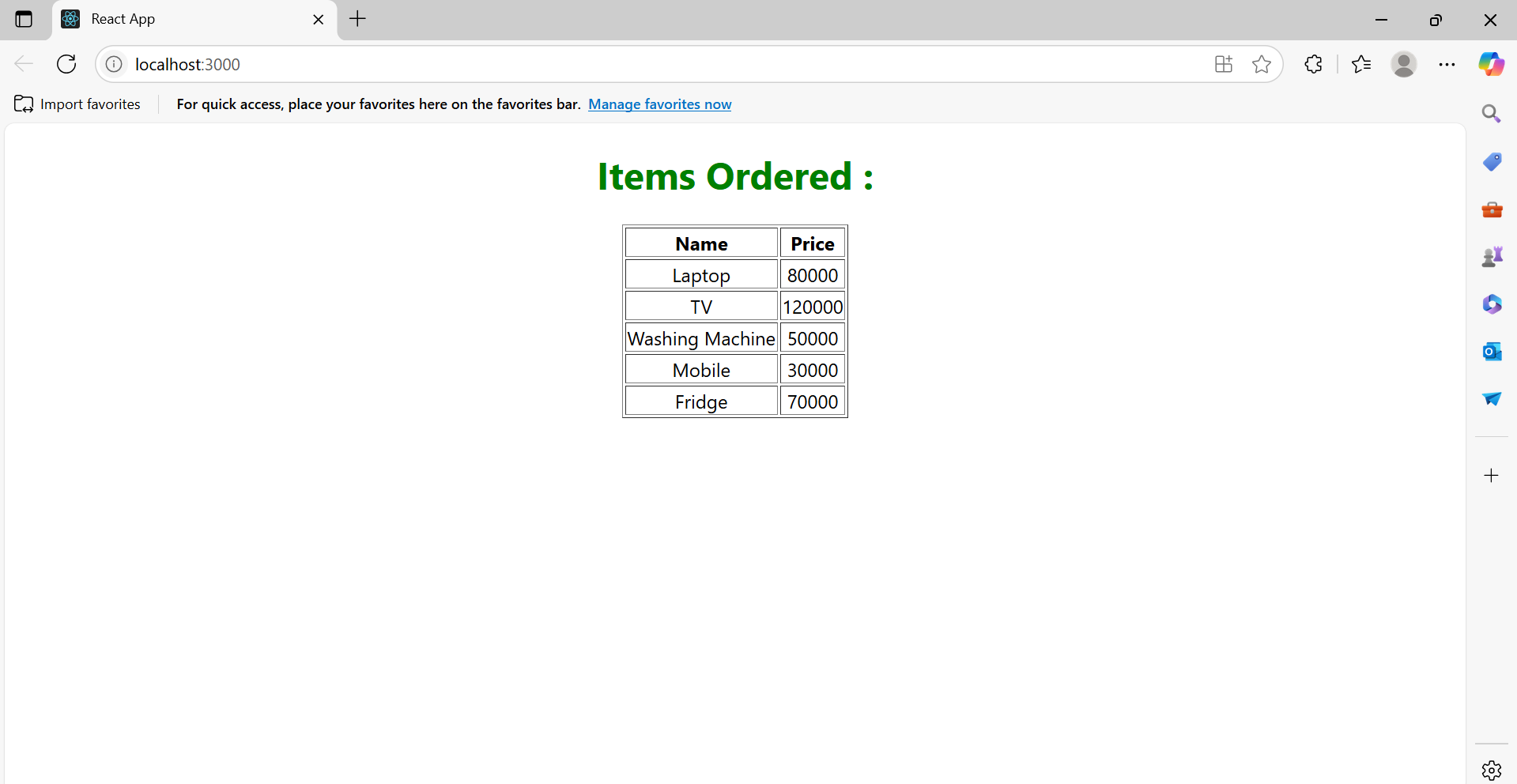
<OnlineShopping />

</div>

);

}

export default App;

**Output:  
**

**8.ReactJS\_HOL**

**What is React State?**

In React, state is an object that determines the behavior and rendering of a component. State allows components to create dynamic and interactive UIs. When the state changes, React automatically re-renders the component to reflect those changes.

**App.js**

import React from "react";

import CountPeople from "./CountPeople";

function App() {

return (

<div className="App">

<CountPeople />

</div>

);

}

export default App;

**CountPeople.js**

import React, { Component } from 'react';

class CountPeople extends Component {

constructor() {

super();

this.state = {

entrycount: 0,

exitcount: 0

};

}

updateEntry = () => {

this.setState(prevState => ({

entrycount: prevState.entrycount + 1

}));

}

updateExit = () => {

this.setState(prevState => ({

exitcount: prevState.exitcount + 1

}));

}

render() {

return (

<div style={{ textAlign: 'center', marginTop: '100px' }}>

<button

onClick={this.updateEntry}

style={{ backgroundColor: 'lightgreen', marginRight: '20px', padding: '10px 20px' }} >

Login

</button>

<span style={{ marginRight: '40px' }}>{this.state.entrycount} People Entered!!!</span>

<button

onClick={this.updateExit}

style={{ backgroundColor: 'lightgreen', padding: '10px 20px' }} >

Exit

</button>

<span style={{ marginLeft: '20px' }}>{this.state.exitcount} People Left!!!</span>

</div>

);

}

}

export default CountPeople;

**index.js**

import React from 'react';

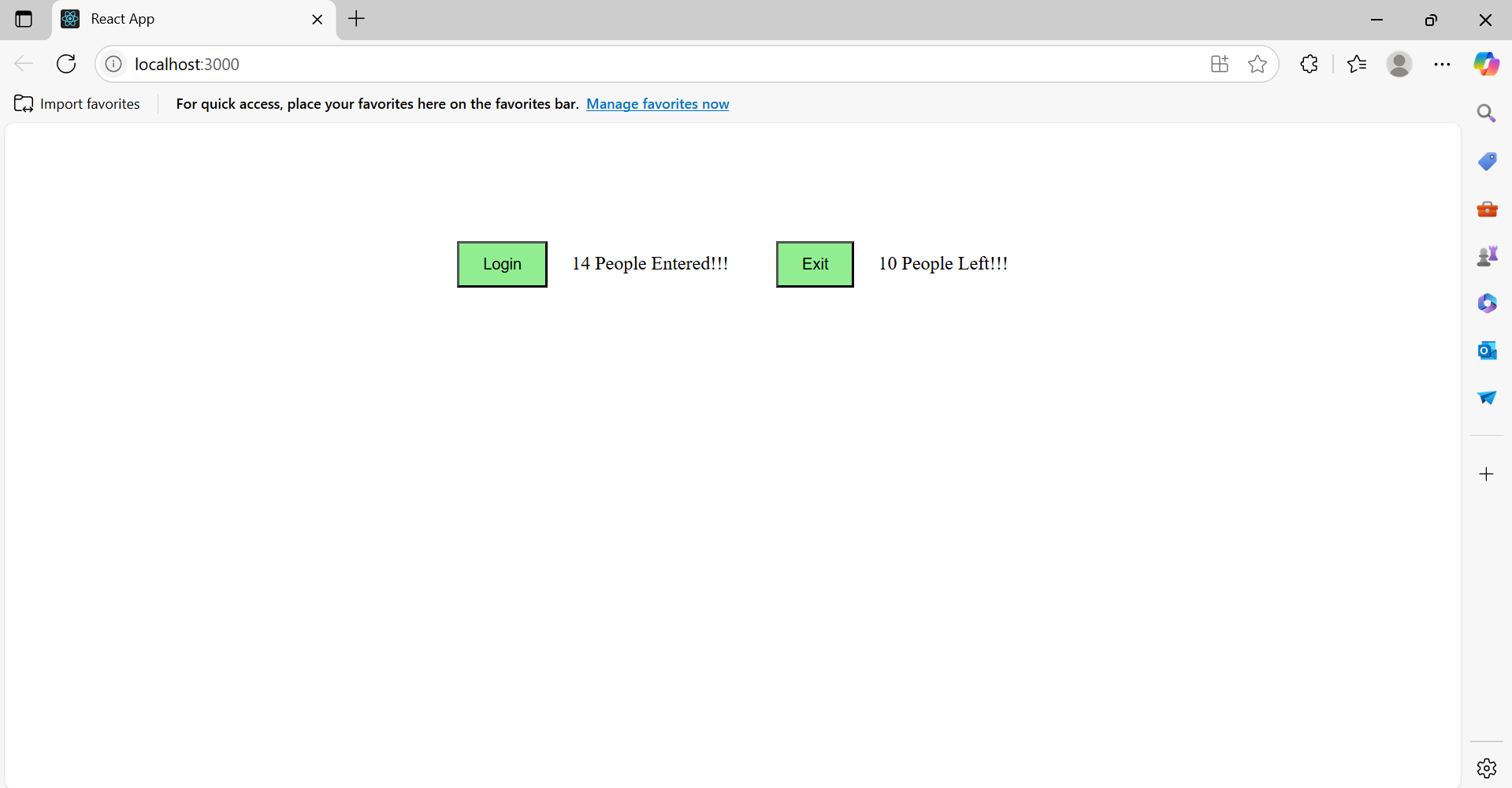
import ReactDOM from 'react-dom/client';

import App from './App';

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

root.render(<App />);

**Output:**

****