**WEEK – 07**

**React**

**Exercise – 09**

**1. Features of ES6 (ECMAScript 2015)**

* Block-scoped variables using let and const
* Arrow functions () => {}
* Classes and inheritance
* Template literals using backticks (`Hello ${name}`)
* Destructuring assignment for arrays and objects
* Default, rest, and spread parameters
* Modules (import / export)
* Promise for asynchronous operations
* Map and Set data structures

**2. JavaScript let**

let is used to declare **block-scoped** variables. It allows re-assignment but not re-declaration in the same scope.

**Example:**

let name = "Alice";

name = "Bob"; // allowed

let name = "Charlie"; // error (same block)

**3. Difference Between var and let**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Redeclaration | Allowed | Not allowed |
| Hoisting | Hoisted (undefined) | Hoisted (TDZ\*) |
| Use in loops | Shared across loops | New scope per loop |

**4. JavaScript const**

* Used to declare block-scoped constants
* Cannot be reassigned
* However, object properties or array elements can still be modified

**Example:**

const x = 10;

x = 20; // Error

const arr = [1, 2];

arr.push(3); // allowed

**5. ES6 Class Fundamentals**

* Introduces class syntax for OOP
* Supports constructors, methods, and inheritance
* Improves readability over prototype-based code

**Example:**

class Person {

constructor(name) {

this.name = name;

}

greet() {

console.log(`Hello, ${this.name}`);

}

}

**6. ES6 Class Inheritance**

* extends keyword is used for inheritance
* super() calls the parent class constructor

**Example:**

class Employee extends Person {

constructor(name, id) {

super(name); // calls Person's constructor

this.id = id;

}

showId() {

console.log(`ID: ${this.id}`);

}

}

**7. ES6 Arrow Functions**

* Shorter syntax for function expressions
* Inherits this from surrounding scope (lexical binding)

**Example:**

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

**8. Set() and Map() in ES6**

**Set()**

* Collection of unique values
* No duplicates allowed

**Example:**

const s = new Set([1, 2, 2, 3]);

console.log(s); // Set {1, 2, 3}

**Map()**

* Stores key-value pairs
* Keys can be any type (even objects)

**Example:**

const m = new Map();

m.set("name", "Alice");

m.set(1, "One");

console.log(m.get("name")); // "Alice"

**CODE:**

**ListofPlayers.js:**

import React from 'react';

function ListofPlayers({ players }) {

return (

<ul>

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

<li key={index}>

{player.name} {player.score}

</li>

))}

</ul>

);

}

export default ListofPlayers;

**Scorebelow70.js:**

import React from 'react';

const Scorebelow70 = ({ players }) => {

const filteredPlayers = players.filter(player => player.score < 70);

return (

<ul>

{filteredPlayers.map((player, index) => (

<li key={index}>

{player.name} {player.score}

</li>

))}

</ul>

);

};

export default Scorebelow70;

**OddPlayers.js:**

import React from 'react';

const OddPlayers = ([first, second, third, fourth, fifth, sixth]) => {

return (

<ul>

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

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

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

</ul>

);

};

export default OddPlayers;

**EvenPlayers.js:**

import React from 'react';

const EvenPlayers = ([first, second, third, fourth, fifth, sixth]) => {

return (

<ul>

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

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

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

</ul>

);

};

export default EvenPlayers;

**ListofIndianPlayers.js:**

import React from 'react';

const ListofIndianPlayers = ({ IndianPlayers }) => {

return (

<ul>

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

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

))}

</ul>

);

};

export default ListofIndianPlayers;

**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';

function App() {

**const flag = true; // Change to false to see other output**

const players = [

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

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

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

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

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

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

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

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

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

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

{ name: 'Mr. Rohit', score: 80 },

];

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

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

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

const IndianPlayers = [...T20Players, ...RanjiTrophyPlayers]; // ES6 merge using spread operator

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>

);

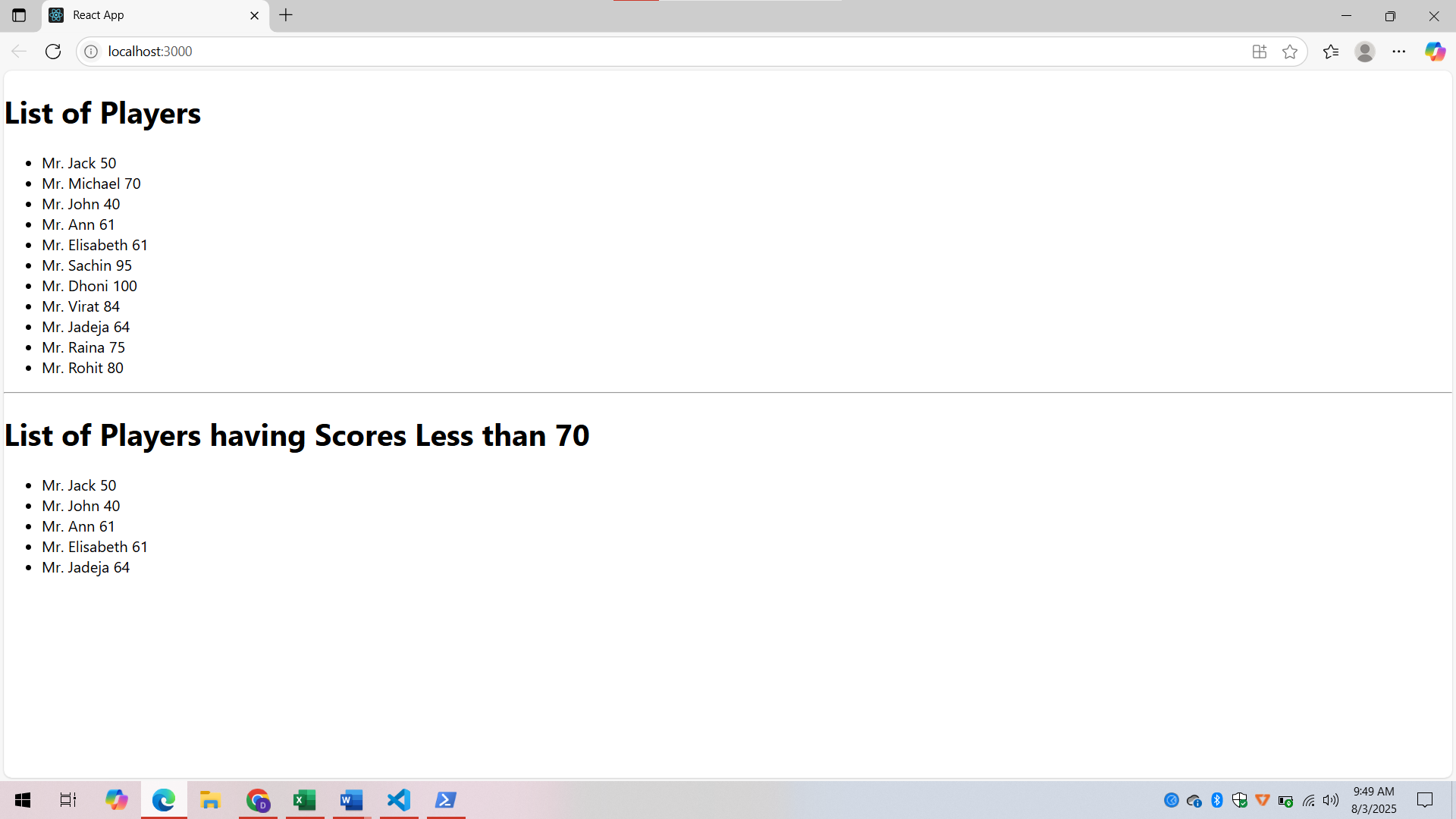
}

}

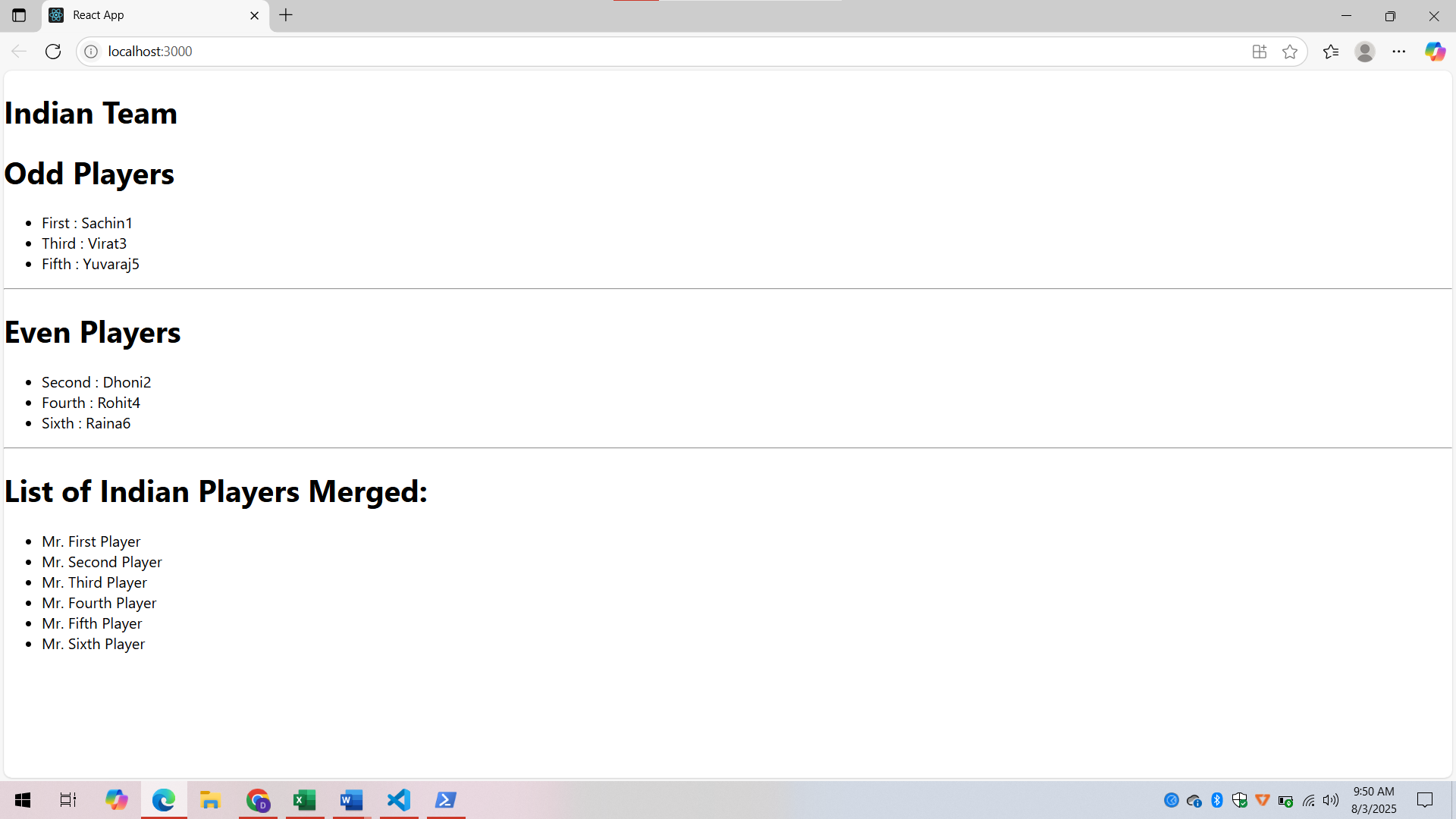
export default App;

**Output:**

**When flag=true,**



**When flag is false**



**Explanation:**

* An array of 11 players with names and scores is declared and displayed using map() from ES6.
* Players with scores below 70 are filtered using the ES6 arrow function and displayed.
* Two arrays T20Players and RanjiTrophyPlayers are merged using the ES6 spread operator.
* Odd and even team players are shown using array destructuring.
* A flag variable is used to conditionally render two different views using if-else.

**Exercise – 10**

**1. Define JSX**

JSX (JavaScript XML) is a syntax extension for JavaScript used in React. It allows writing HTML-like code within JavaScript, making it easier to describe the UI structure.

**2. Explain about ECMAScript**

ECMAScript (ES) is the official standard for JavaScript defined by ECMA International. Versions like ES5, ES6 (ES2015), etc., introduce new features to the language, such as let, const, arrow functions, classes, promises, and modules.

**3. Explain React.createElement()**

React.createElement() is a core method used by React to create virtual DOM elements. JSX code is internally converted into this method.

**Example:**

React.createElement('h1', null, 'Hello World');

**4. How to Create React Nodes with JSX**

React nodes can be created using JSX by writing HTML-like elements in JavaScript.

**Example:**

const element = <h1>Hello React</h1>;

**5. How to Render JSX to DOM**

JSX elements are rendered to the actual DOM using ReactDOM.render() method.

**Example:**

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

**6. How to Use JavaScript Expressions in JSX**

JavaScript expressions can be embedded inside JSX using curly braces {}.

**Example:**

const name = "React";

<h1>Hello, {name}!</h1>

**7. How to Use Inline CSS in JSX**

Inline CSS in JSX is written as a JavaScript object with camelCase property names.

**Example:**

<h1 style={{ color: 'blue', fontSize: '20px' }}>Styled Text</h1>

**CODE:**

**OfficeList.js:**

import React from 'react';

import './OfficeList.css'; // optional CSS file if needed

const OfficeList = () => {

  const element = "Office Space";

  const imageSrc = "https://images.unsplash.com/photo-1509395176047-4a66953fd231?auto=format&fit=crop&w=600&q=60";

  const offices = [

    { Name: "DBS", Rent: 50000, Address: "Chennai" },

    { Name: "WeWork", Rent: 70000, Address: "Bangalore" },

    { Name: "Regus", Rent: 45000, Address: "Mumbai" },

  ];

  return (

    <center>

      <h1><b>{element} , at Affordable Range</b></h1>

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

        let rentClass = office.Rent <= 60000 ? "textRed" : "textGreen";

        return (

          <div key={index}>

            <img src={imageSrc} width="25%" height="25%" alt="Office Space" />

            <h1>Name: {office.Name}</h1>

            <h3 className={rentClass}>Rent: Rs. {office.Rent}</h3>

            <h3>Address: {office.Address}</h3>

            <hr />

          </div>

        );

      })}

    </center>

  );

};

export default OfficeList;

**OfficeList.css:**

.textRed {

  color: red;

  font-weight: bold;

}

.textGreen {

  color: green;

  font-weight: bold;

}

**App.js:**

import React from 'react';

import OfficeList from './OfficeList';

function App() {

  return (

    <div className="App">

      <OfficeList />

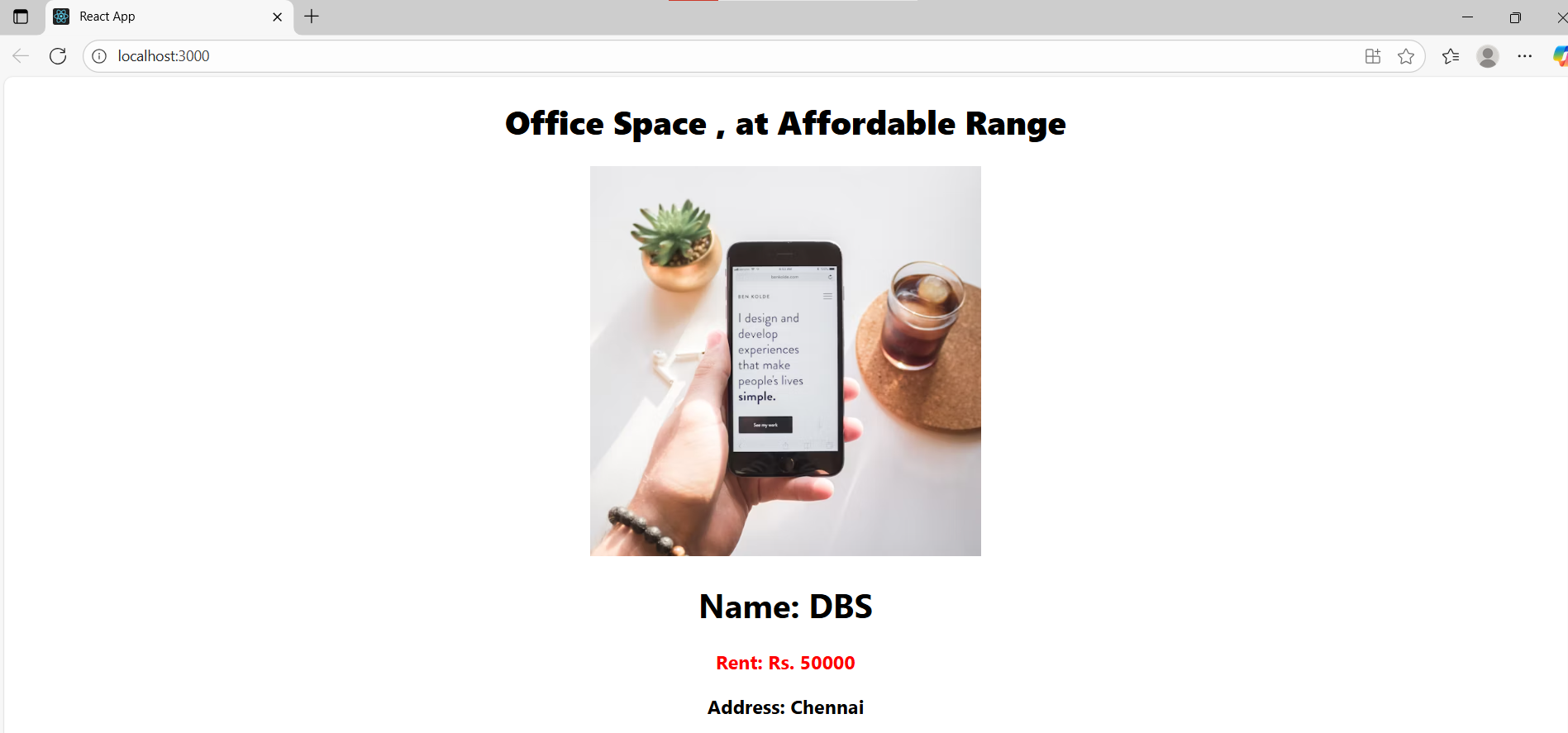
    </div>

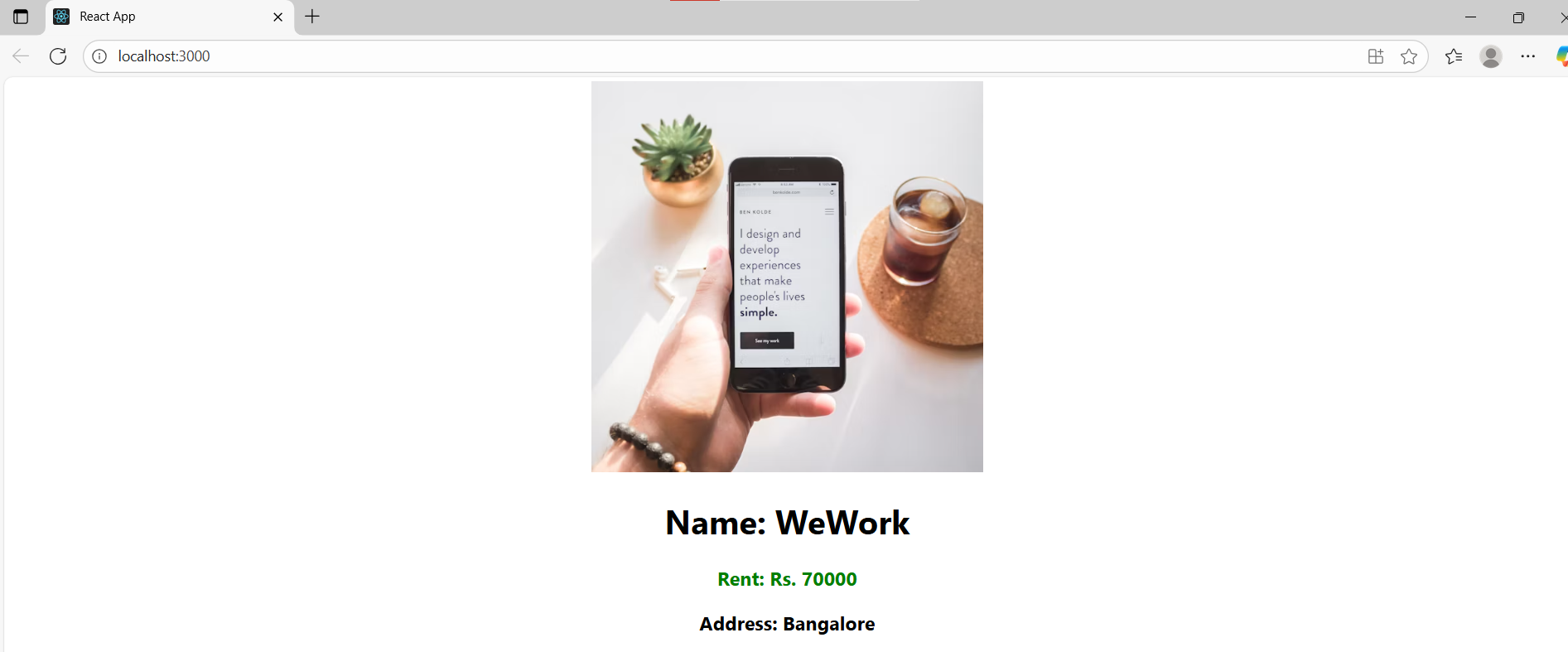
  );

}

export default App;

**Output:**





**Explanation:**

* JSX is used to create elements like headings, images, and office details inside the component.
* An object (ItemName) stores office data including name, rent, and address.
* Image attributes such as src, width, and alt are defined using JSX.
* Conditional styling is applied to the rent: red if it is less than or equal to ₹60,000 and green otherwise.
* Data is rendered dynamically using {} to insert JavaScript expressions inside JSX.

**Exercise – 11**

**1. Explain React Events**

React events are triggered by user interactions such as clicks, form submissions, or typing. React wraps native DOM events in a cross-browser wrapper called SyntheticEvent to ensure consistency across all browsers.

**2. Explain About Event Handlers**

Event handlers are functions that are called when a specific event occurs. In React, they are passed as props using camelCase syntax, e.g., onClick={handleClick}. They allow you to define logic that should run when an event happens.

**3. Define Synthetic Event**

A SyntheticEvent is a wrapper around the browser’s native event system in React. It provides a consistent and normalized interface to access event properties and behavior across different browsers.

**4. Identify React Event Naming Convention**

React uses camelCase for event names instead of lowercase. For example:

* onClick (React) vs onclick (HTML)
* onChange, onSubmit, onKeyDown are all camelCase in React.

**CODE:**

**Counter.js:**

import React, { Component } from 'react';

class Counter extends Component {

constructor() {

super();

this.state = { count: 0 };

}

increment = () => {

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

this.sayHello();

};

decrement = () => {

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

};

sayHello = () => {

alert("Hello! Count updated successfully.");

};

render() {

return (

<div>

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

<button onClick={this.increment}>Increase</button>

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

</div>

);

}

}

export default Counter;

**WelcomeButton.js:**

import React from 'react';

function WelcomeButton() {

const sayWelcome = (msg) => {

alert(msg);

};

return (

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

);

}

export default WelcomeButton;

**SyntheticEventButton.js:**

import React from 'react';

function SyntheticEventButton() {

const handleClick = (e) => {

e.preventDefault(); // Synthetic Event

alert("I was clicked");

};

return (

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

);

}

export default SyntheticEventButton;

**CurrencyConverter.js:**

import React, { useState } from 'react';

function CurrencyConverter() {

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

const [euros, setEuros] = useState('');

const handleSubmit = (e) => {

e.preventDefault();

const euroValue = (parseFloat(rupees) / 90).toFixed(2);

setEuros(euroValue);

};

return (

<div>

<h2>Currency Converter</h2>

<form onSubmit={handleSubmit}>

<input

type="number"

value={rupees}

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

placeholder="Enter amount in Rupees"

/>

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

</form>

{euros && <h3>Value in Euros: €{euros}</h3>}

</div>

);

}

export default CurrencyConverter;

**App.js:**

import React from 'react';

import Counter from './Counter';

import WelcomeButton from './WelcomeButton';

import SyntheticEventButton from './SyntheticEventButton';

import CurrencyConverter from './CurrencyConverter';

function App() {

return (

<div className="App">

<h1>Event Handling Examples</h1>

<Counter />

<br />

<WelcomeButton />

<br /><br />

<SyntheticEventButton />

<br /><br />

<CurrencyConverter />

</div>

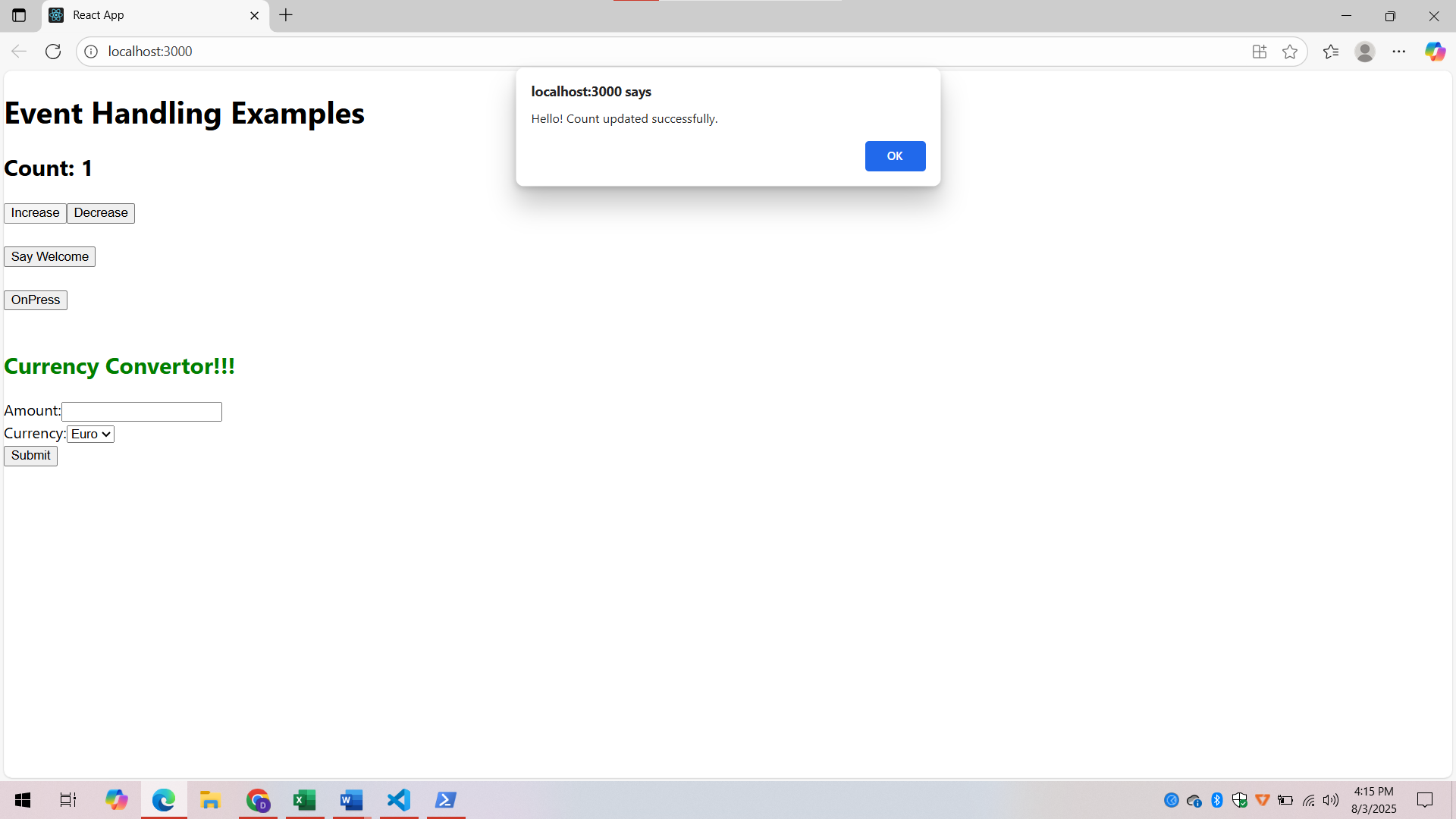
);

}

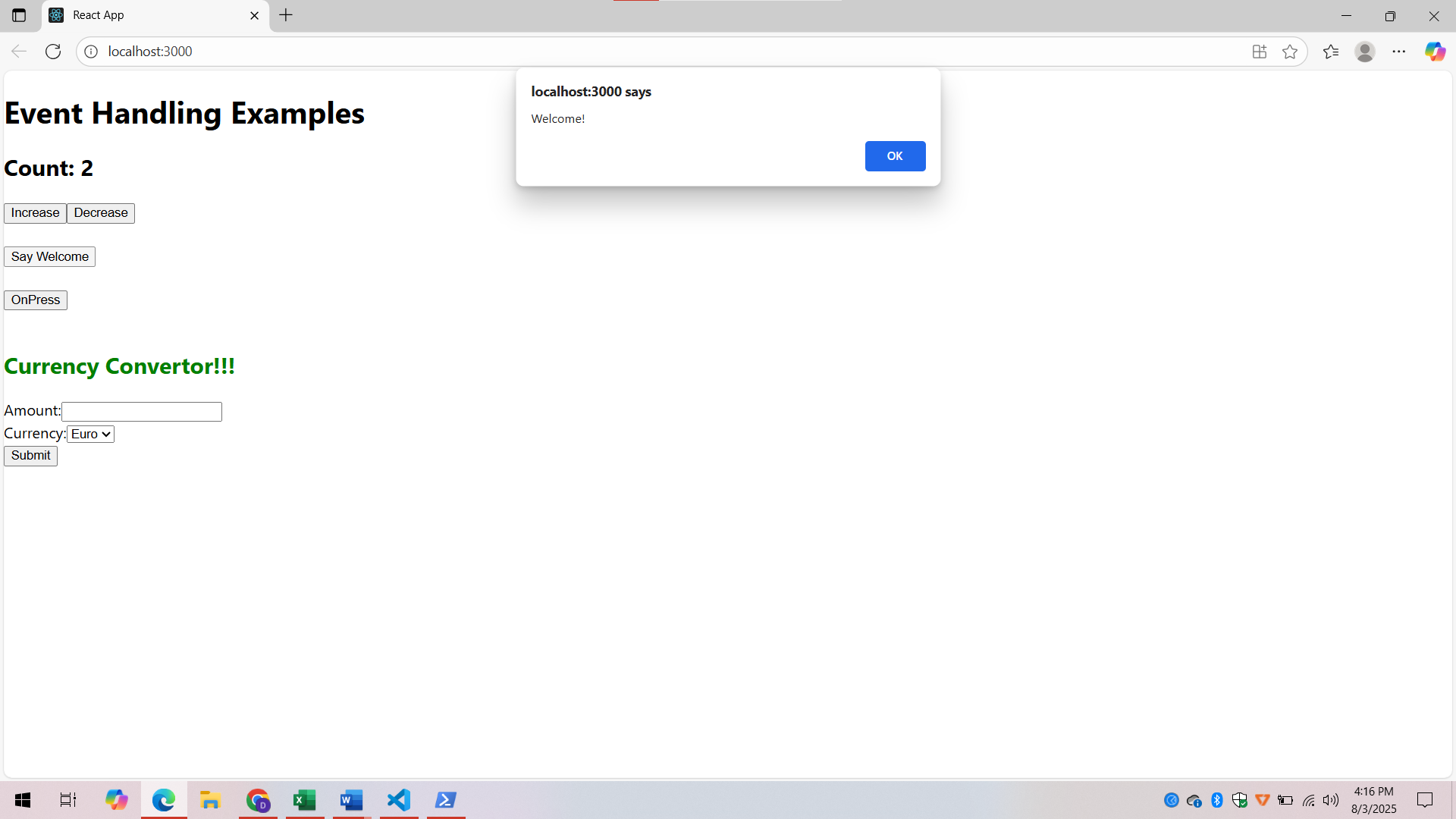
export default App;

**Output:**

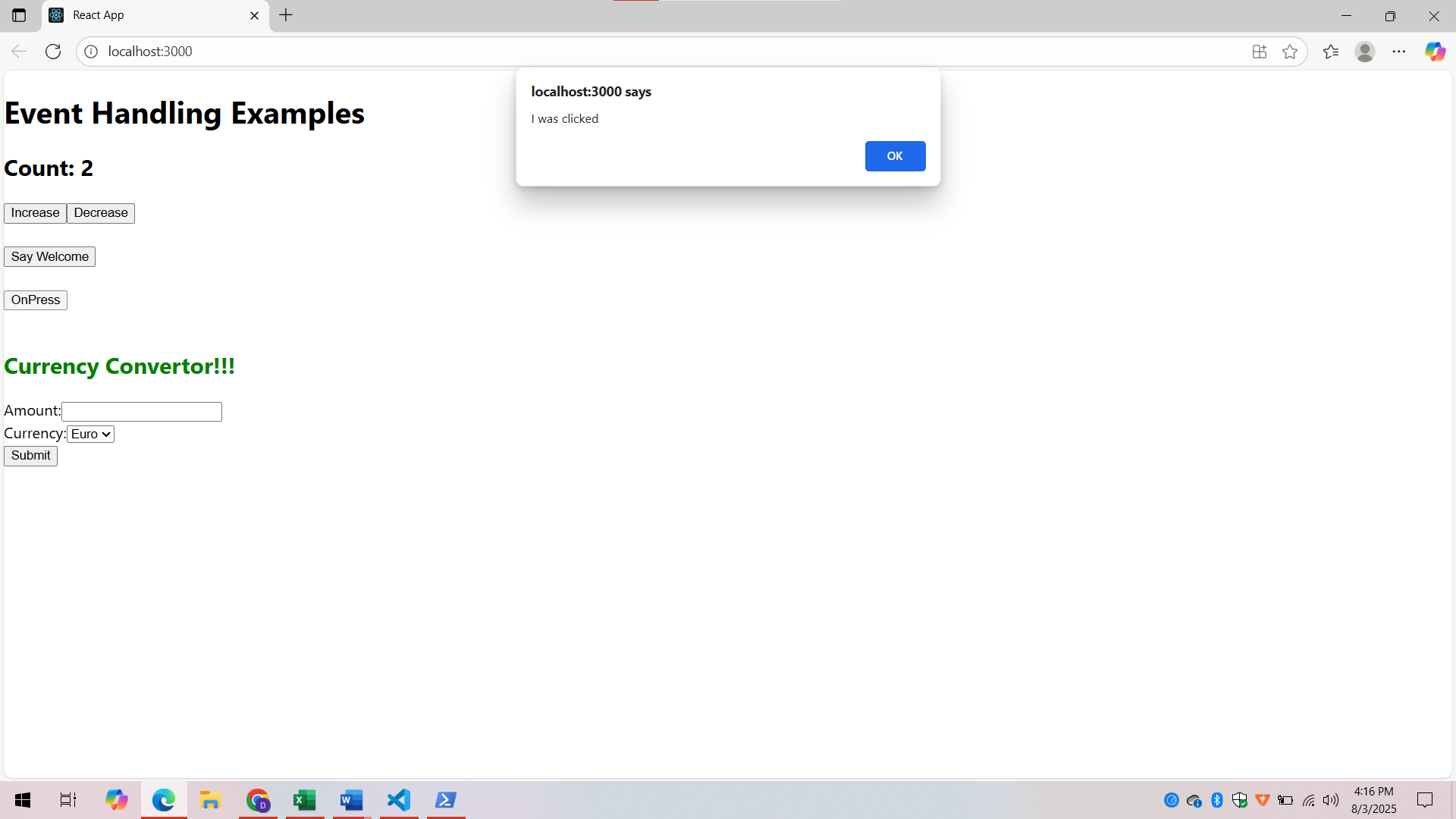
**When Currency is converted and increment is clicked,**



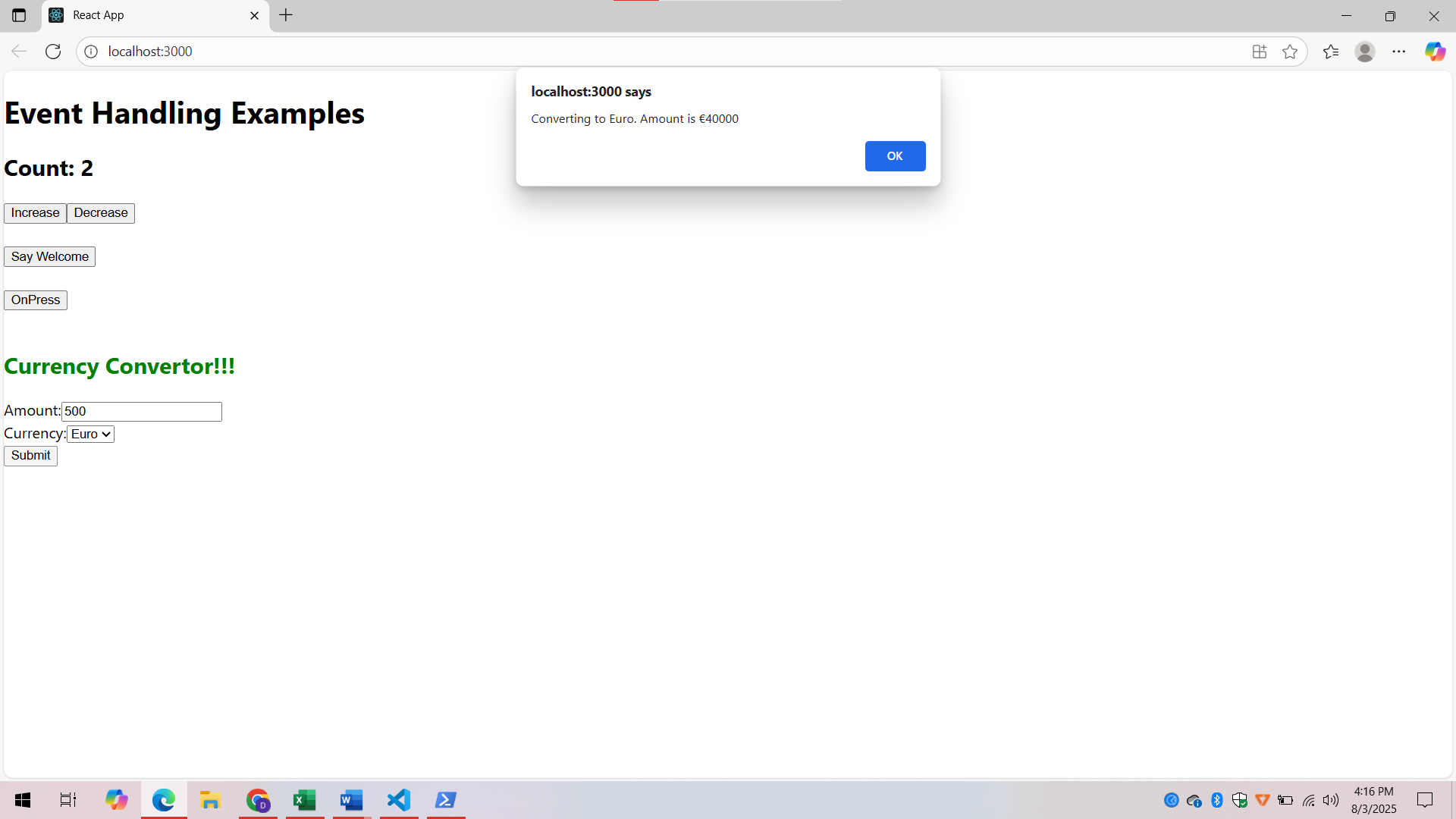
**When “Say Welcome” button is clicked,**



**When “OnPress” is clicked, (Named “OnPress” instead of “Click on me”)**



**When currency is converted into Euro,**



**Explanation:**

* A functional component CurrencyConverter is created using React hooks (useState) to manage user inputs.
* Users can enter an amount in Indian Rupees and choose the currency (Euro).
* On form submission, the handleSubmit function is called.
* It performs the currency conversion (₹1 = €0.0125).
* A popup alert is shown displaying the converted Euro amount using alert().

**Exercise – 12**

**1. Explain about Conditional Rendering in React**

Conditional rendering in React allows components to render different outputs based on a condition (e.g., login status). It works like JavaScript conditionals using if, ? : , or &&.

**Example:**

{isLoggedIn ? <UserPage /> : <GuestPage />}

**2. Define Element Variables**

Element variables are used to store JSX elements in a variable before including them in the return statement. This helps organize code when rendering conditionally. **Example:**

let greeting;

if (isLoggedIn) {

greeting = <h1>Welcome</h1>;

} else {

greeting = <h1>Please sign up</h1>;

}

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

**3. Explain How to Prevent Components from Rendering**

To prevent a component from rendering, use a condition that returns null. Returning null tells React to skip rendering that component without affecting the rest of the UI. **Example:**

function WarningBanner(props) {

if (!props.show) {

return null;

}

return <div className="warning">Warning!</div>;

}

**CODE:**

**App.js:**

import React, { useState } from 'react';

function UserGreeting() {

  return <h2>Welcome back!</h2>;

}

function GuestGreeting() {

  return <h2>Please sign up.</h2>;

}

function Greeting(props) {

  const isLoggedIn = props.isLoggedIn;

  if (isLoggedIn) {

    return <UserGreeting />;

  }

  return <GuestGreeting />;

}

function LoginButton(props) {

  return (

    <button onClick={props.onClick}>

      Login

    </button>

  );

}

function LogoutButton(props) {

  return (

    <button onClick={props.onClick}>

      Logout

    </button>

  );

}

function App() {

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

  const handleLoginClick = () => {

    setIsLoggedIn(true);

  };

  const handleLogoutClick = () => {

    setIsLoggedIn(false);

  };

  let button;

  if (isLoggedIn) {

    button = <LogoutButton onClick={handleLogoutClick} />;

  } else {

    button = <LoginButton onClick={handleLoginClick} />;

  }

  return (

    <div style={{ textAlign: 'center', marginTop: '50px' }}>

      <h1>Ticket Booking App</h1>

      <Greeting isLoggedIn={isLoggedIn} />

      {button}

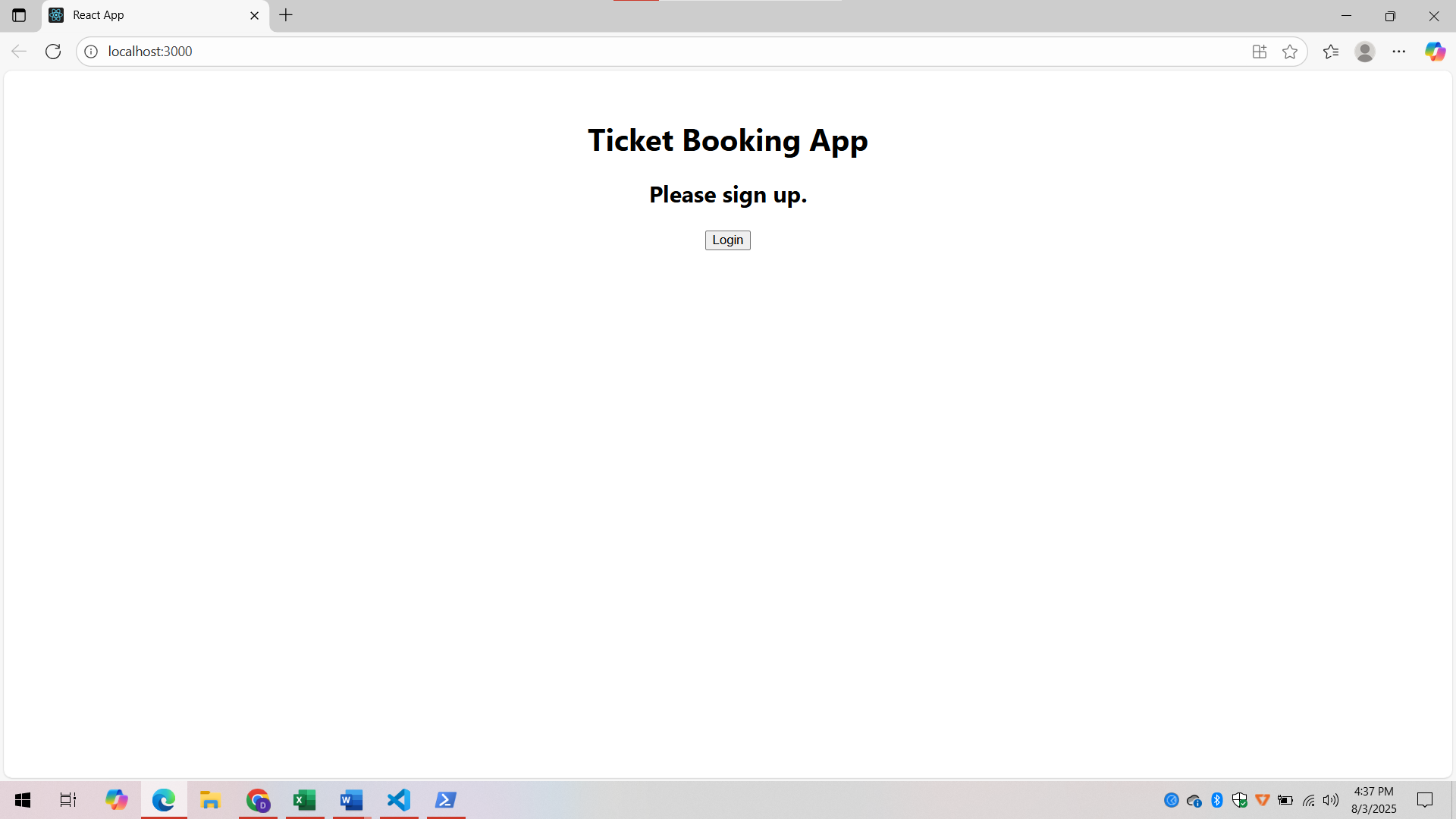
    </div>

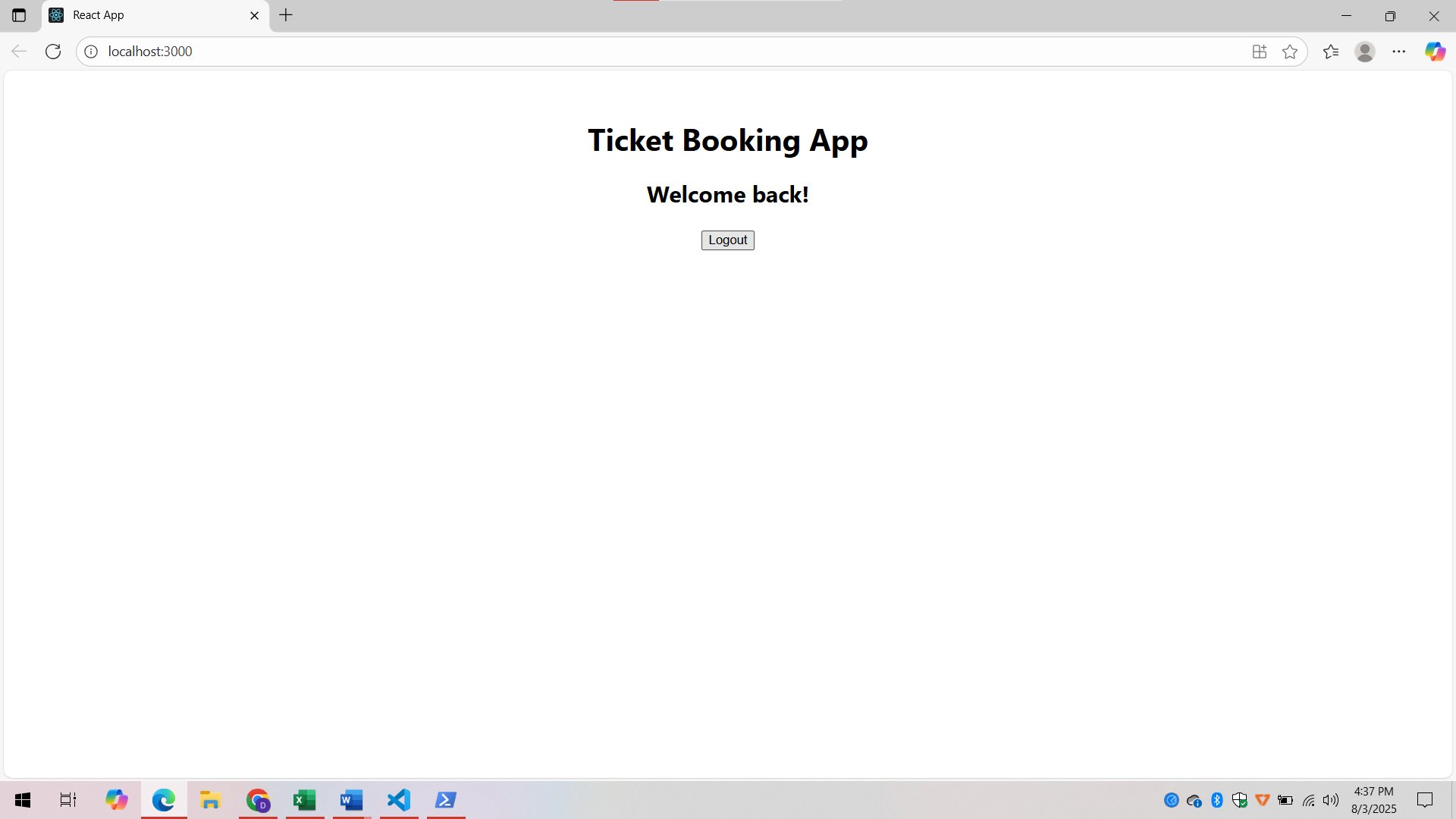
  );

}

export default App;

**Output:**

****

****

**Explanation:**

* This app uses conditional rendering to show different messages and buttons based on login status.
* Greeting component chooses between UserGreeting and GuestGreeting using a prop.
* LoginButton and LogoutButton components call the respective onClick function passed as props.
* useState() tracks whether the user is logged in (true) or not (false).
* Clean, modular design improves readability and reusability.

**Exercise – 13**

**1. Explain Various Ways of Conditional Rendering**

React supports multiple ways to conditionally render components:

* **If-else statements:** Used inside functions to assign JSX to variables.
* **Ternary operator (? :):** Inline conditional rendering.
* **Logical AND (&&) operator:** Render a component only if the condition is true.
* **Element variables:** Store conditional JSX in a variable and render it in return.

**2. Explain How to Render Multiple Components**

**Multiple components can be rendered in a single return block using JSX:**

return (

<div>

<Header />

<Main />

<Footer />

</div>

);

**Alternatively, use fragments (<></>) to avoid extra DOM nodes:**

<>

<Component1 />

<Component2 />

</>

**3. Define List Component**

A list component is used to render multiple similar items (like products, posts, users) by looping over an array and returning components dynamically.

**Example:**

function BookList({ books }) {

return (

<ul>

{books.map((book) => (

<li key={book.id}>{book.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, are added, or removed. Using stable and unique keys improves performance.

**{items.map(item => <li key={item.id}>{item.name}</li>)}**

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

In React, when rendering lists, it's a good practice to extract list items into their own components to improve readability and reusability. Even when extracting, the key prop must be assigned to the top-level element in the list, and not inside the child component.

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

React uses JavaScript’s map() function to loop over arrays and return JSX for each item.

**It's commonly used in lists:**

books.map(book => <li>{book.title}</li>);

Each returned element must have a key for efficient rendering.

**CODE:**

**Details.js:**

import React from 'react';

function Details(props) {

  const { books, blogs, courses, view } = props;

  if (view === 'books') {

    return (

      <div>

        <h1>Book Details</h1>

        <ul>

          {books.map((book) => (

            <div key={book.id}>

              <h3>{book.bname}</h3>

              <h4>{book.price}</h4>

            </div>

          ))}

        </ul>

      </div>

    );

  }

  if (view === 'blogs') {

    return (

      <div>

        <h1>Blog Details</h1>

        <ul>

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

            <div key={index}>

              <h3>{blog.title}</h3>

              <h4>{blog.author}</h4>

              <p>{blog.content}</p>

            </div>

          ))}

        </ul>

      </div>

    );

  }

  if (view === 'courses') {

    return (

      <div>

        <h1>Course Details</h1>

        <ul>

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

            <div key={index}>

              <h3>{course.name}</h3>

              <h4>{course.date}</h4>

            </div>

          ))}

        </ul>

      </div>

    );

  }

  return <p>No section selected</p>;

}

export default Details;

**Details.css:**

.container {

  display: flex;

  justify-content: space-around;

  margin: 20px;

}

.section {

  flex: 1;

  padding: 20px;

  border-right: 5px solid green;

}

.section:last-child {

  border-right: none;

}

.section h1 {

  font-size: 24px;

  font-weight: bold;

  text-align: center;

}

**App.js:**

import React, { useState } from 'react';

import Details from './Details';

function App() {

  const [view, setView] = useState('books'); // default view

  const books = [

    { id: 101, bname: 'Master React', price: 670 },

    { id: 102, bname: 'Deep Dive into Angular 11', price: 800 },

    { id: 103, bname: 'Mongo Essentials', price: 450 },

  ];

  const blogs = [

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

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

  ];

  const courses = [

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

    { name: 'React', date: '6/3/2021' },

  ];

  return (

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

      <h1>Blogger App</h1>

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

        <button onClick={() => setView('books')}>Show Books</button>{' '}

        <button onClick={() => setView('blogs')}>Show Blogs</button>{' '}

        <button onClick={() => setView('courses')}>Show Courses</button>

      </div>

      <Details books={books} blogs={blogs} courses={courses} view={view} />

    </div>

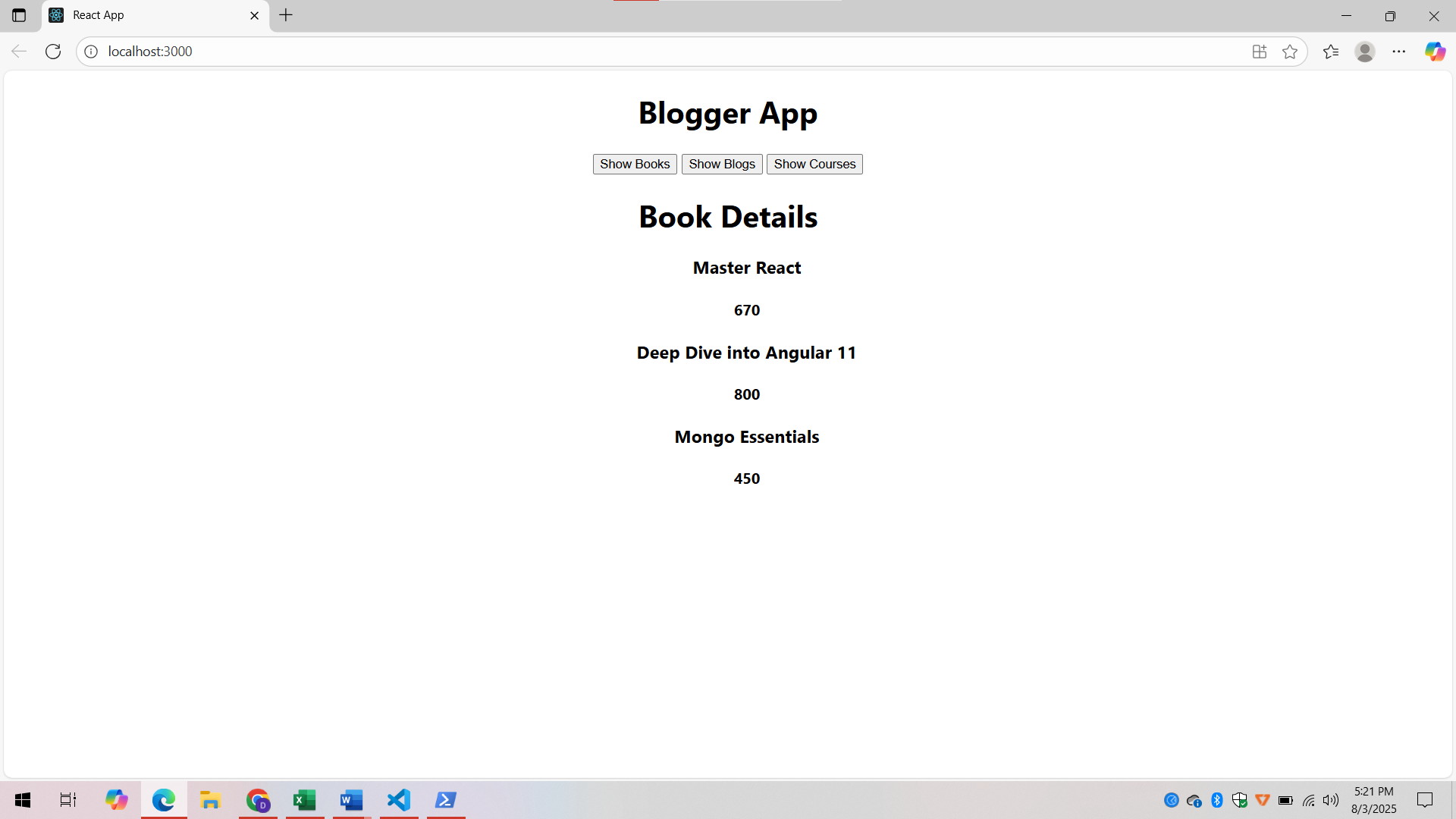
  );

}

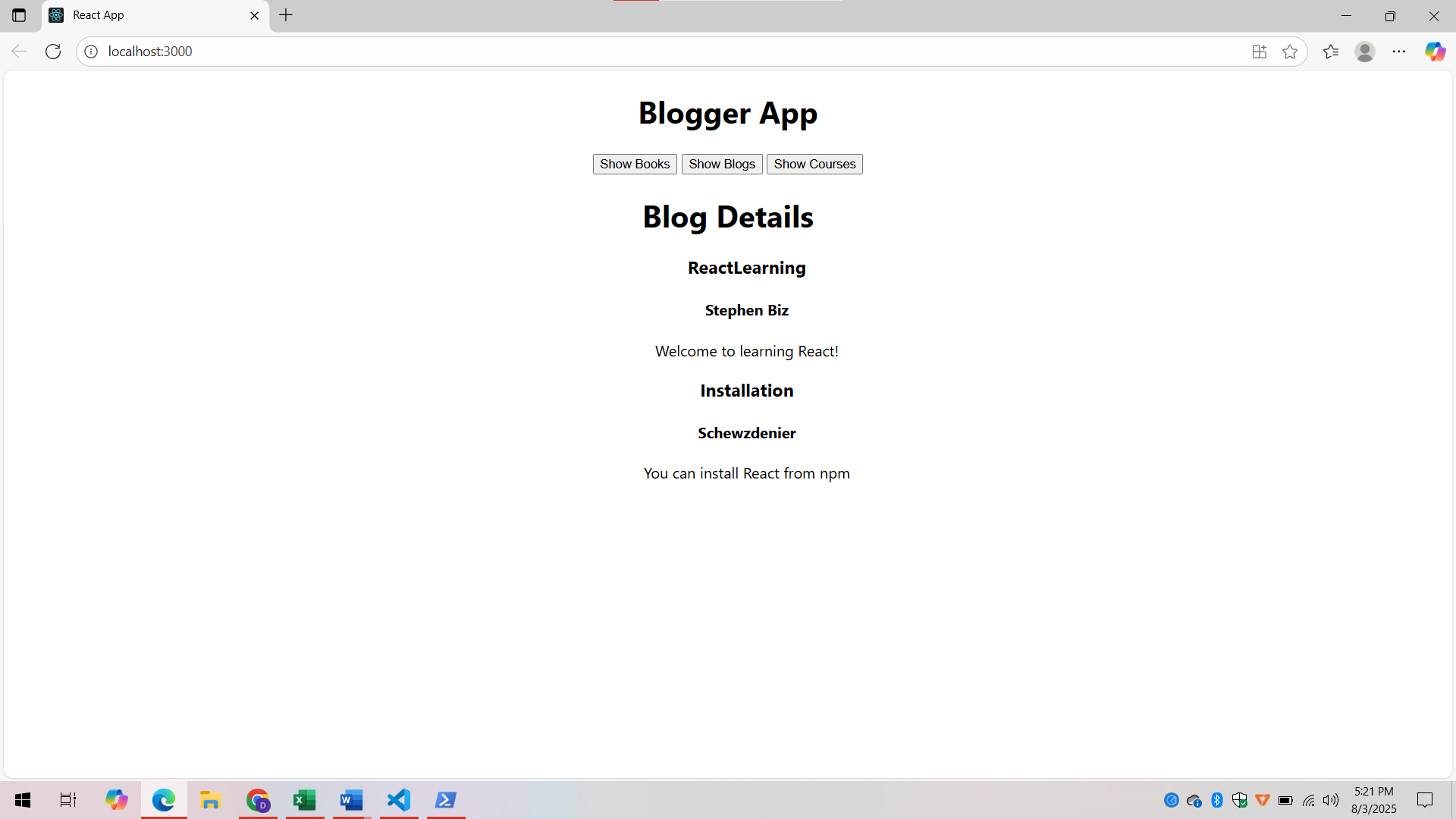
export default App;

**Output:**

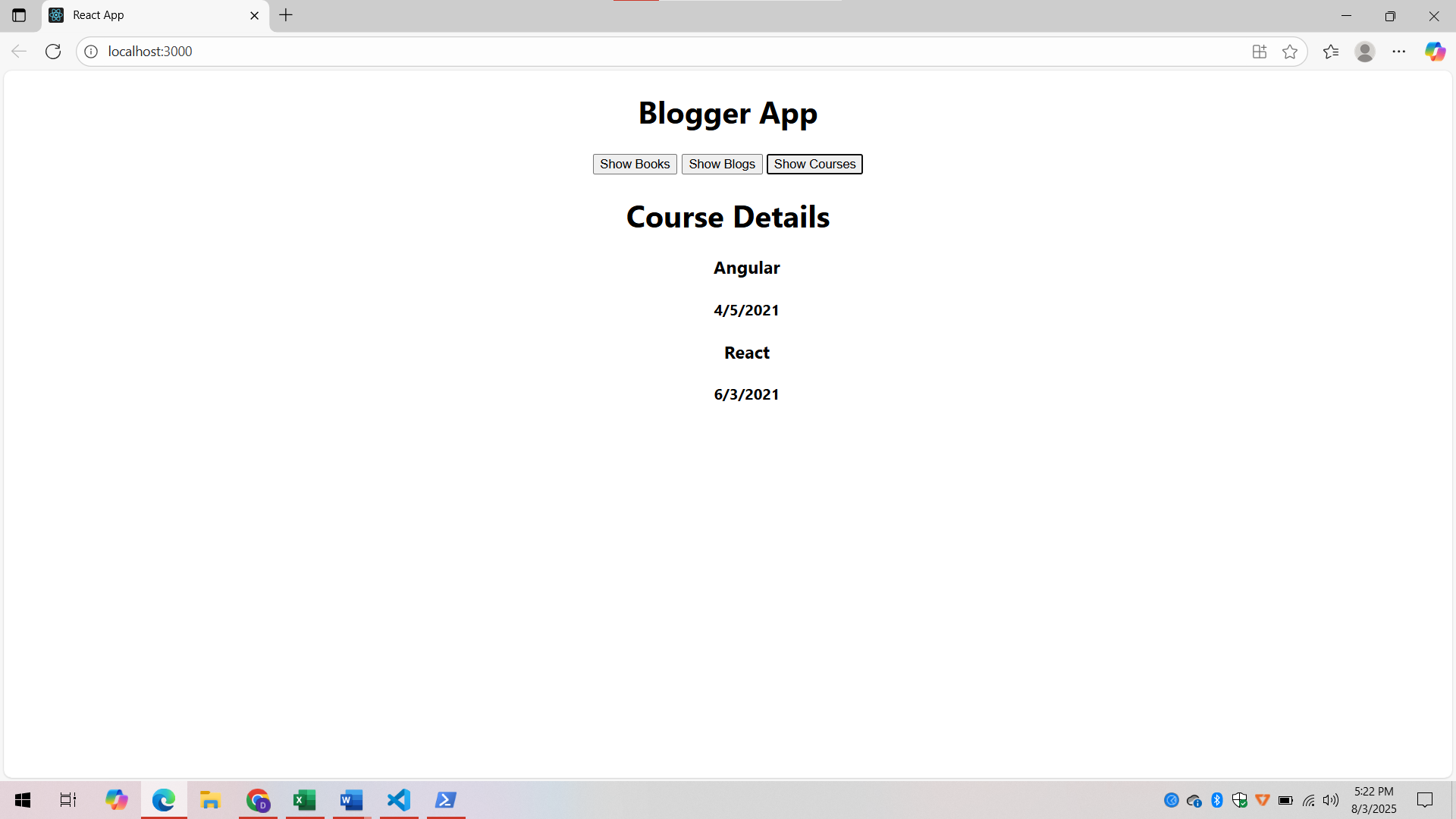
**When “Show Books” is clicked,**



**When “Show Blogs” is clicked,**



**When “Show Courses” is clicked,**



**Explanation:**

* The application uses conditional rendering to display one section at a time based on the user's selection.
* A state variable is used to track which component (Books, Blogs, or Courses) should be shown on the screen.
* if statements are implemented in the child component to control which JSX content is rendered.
* Buttons in the main component allow the user to change the view dynamically by updating the state.
* This approach enhances the user interface by avoiding unnecessary rendering and improving performance.