**Week 07**

**React**

**Exercise - 09**

**1. List the features of ES6:**

\* Arrow functions

\* `let` and `const` declarations

\* Classes and inheritance

\* Template literals

\* Default, rest, and spread parameters

\* Destructuring assignment

\* Modules (`import` / `export`)

\* Promises

\* Enhanced object literals

\* `Map`, `Set`, `WeakMap`, `WeakSet`

**2. Explain JavaScript `let`:**

\* `let` allows you to declare block-scoped variables. Unlike `var`, it's not hoisted to the top of its enclosing block and prevents redeclaration within the same scope.

**3. Identify the differences between `var` and `let`:**

| **Feature** | **‘var’** | **‘let’** |
| --- | --- | --- |
| Scope | Funtion-scoped | Block-scoped |
| Hoisting | Hoisted and initialized as `undefined` | Hoisted but not initialized |
| Redeclaration | Allowed within the same scope | Not allowed within the same scope |

**4. Explain JavaScript `const`:**

\* `const` is used to declare block-scoped constants. The value assigned cannot be reassigned, though for objects and arrays, their contents can still be modified.

**5. Explain ES6 class fundamentals:**

\* ES6 introduces the `class` keyword to define object-oriented structures.

\* It supports constructors, methods, and encapsulation.

\* Syntax is more intuitive and closer to classical OOP languages.

**6. Explain ES6 class inheritance:**

\* Inheritance is achieved using `extends`.

\* The `super()` keyword is used to call the constructor or methods of the parent class.

**7. Define ES6 arrow functions**:

\* A concise way to write functions using `=>`.

\* Lexically binds `this`, making them useful in callbacks.

\* Syntax: `(args) => expression` or `arg => { statements }`

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

\* `Set` is a collection of unique values (no duplicates).

\* `Map` stores key-value pairs, where keys can be of any type and maintains insertion order.

**Explanation:**

* An array containing 11 players' names and scores is declared.
* The map() function is used to display each player's details.
* Players scoring below 70 are filtered using an arrow function.
* Two arrays, T20Players and RanjiTrophyPlayers, are combined using the spread operator.
* Team members are divided into odd and even groups using array destructuring.
* A flag variable determines which of the two UI views to render using a conditional if-else block.

**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>

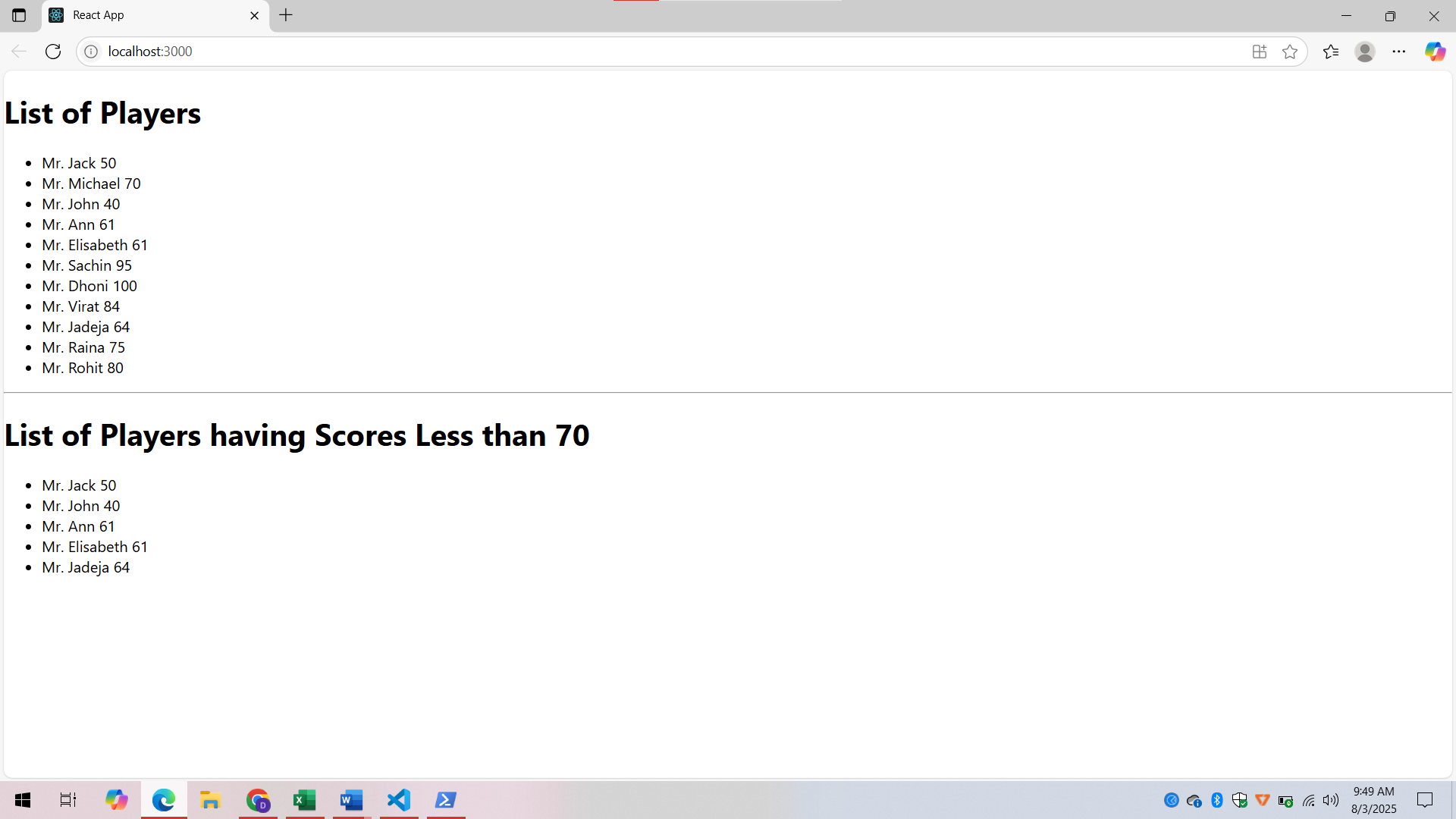
</div>

); }}

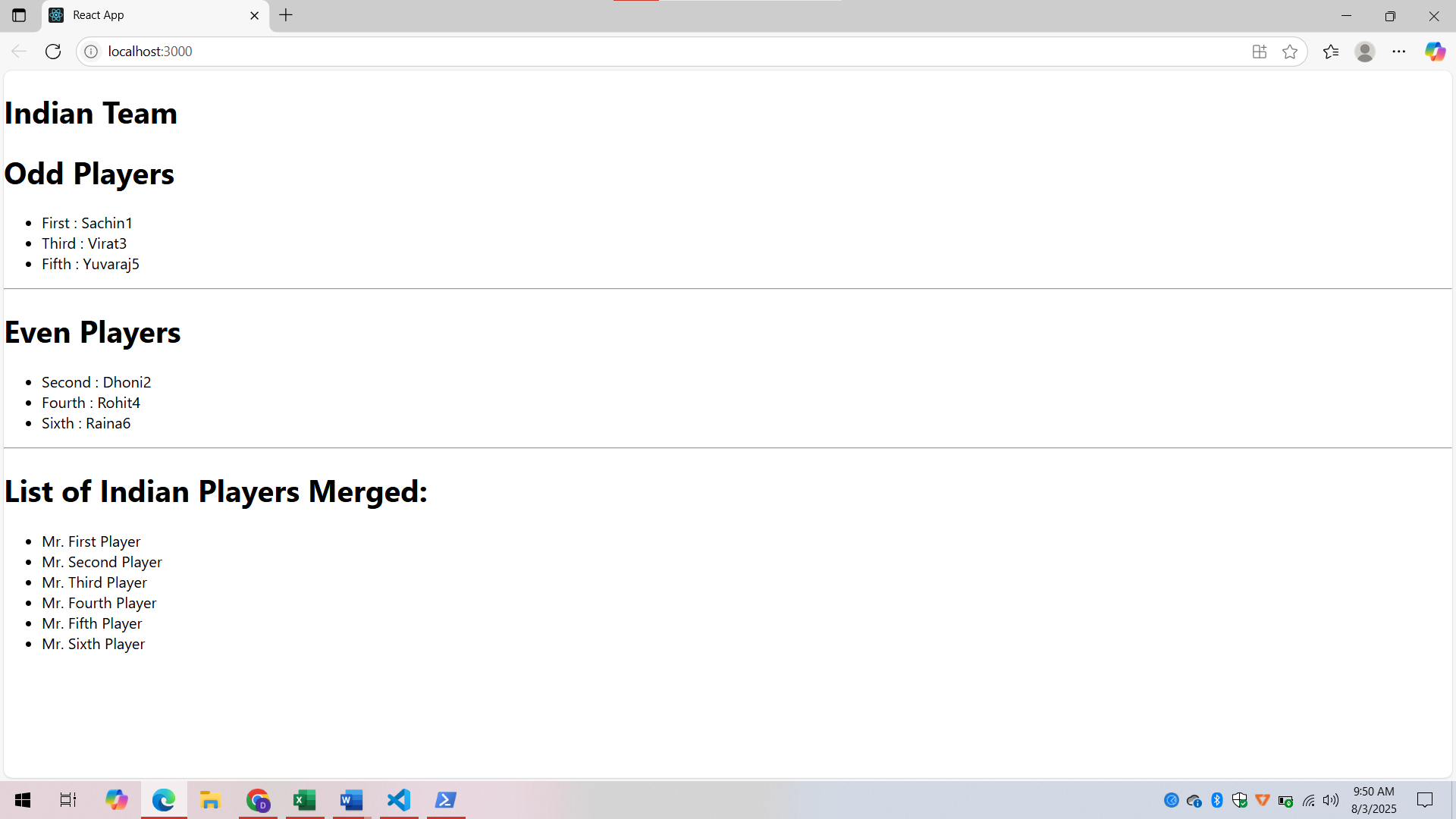
export default App;

**Output:**

When flag=true,



When flag is false



**Exercise - 10**

**1. Define JSX:**

\* JSX (JavaScript XML) is a syntax extension for JavaScript used with React.

\* It allows writing HTML-like code within JavaScript to describe the UI.

**2. Explain about ECMA Script:**

\* ECMAScript is the standard specification for scripting languages like JavaScript.

\* ES6 (ECMAScript 2015) introduced modern features like `let`, `const`, arrow functions, classes, modules, and more.

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

\* A core method in React to create virtual DOM elements.

\* Syntax: `React.createElement(type, props, children)`

\* Used internally by JSX during transpilation.

**4. Explain how to create React nodes with JSX:**

\* Use HTML-like tags directly in JavaScript:

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

\* These compile into `React.createElement()` calls.

**5. Define how to render JSX to the DOM:**

\* Use `ReactDOM.render()` or `createRoot().render()`:

import ReactDOM from 'react-dom/client';

const root =

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

root.render(<App />);

**6. Explain how to use JavaScript expressions in JSX:**

\* Use curly braces `{}` to embed JS expressions inside JSX:

const name = "Deepika";

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

**7. Explain how to use inline CSS in JSX:**

\* Provide a style object using double curly braces:

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

<h1 style={style}>Styled Text</h1>

**Explanation:**

\* Run `npx create-react-app officespacerentalapp` to create a new React app and open it in VS Code.

\* Use JSX to add a heading element and an image with `src` and `alt` attributes.

\* Define an `office` object containing `name`, `rent`, and `address`, and display the details using JSX.

\* Create an array of office objects and use the `map()` function to render each office's details.

\* Apply conditional styling to the `rent` field: show it in red if it's below `60000`, otherwise in green.

**Code**

**App.js**

import React from 'react';

import './App.css';

function App() {

const heading = <h1>🏢 Office Space Rental App</h1>;

const imageURL = "https://via.placeholder.com/400x200.png?text=Office+Space";

const office1 = {

name: "Prestige Tech Park",

rent: 55000,

address: "Bangalore, Karnataka"

};

const officeList = [

{

name: "WeWork Residency",

rent: 45000,

address: "Mumbai, Maharashtra"

},

{

name: "IndiQube Alpha",

rent: 65000,

address: "Chennai, Tamil Nadu"

},

{

name: "Smartworks HQ",

rent: 72000,

address: "Hyderabad, Telangana"

},

{

name: "91Springboard",

rent: 58000,

address: "Delhi NCR"

}

];

return (

<div className="App">

{heading}

<img src={imageURL} alt="Office" width="400" height="200" />

<h2>Featured Office</h2>

<p><strong>Name:</strong> {office1.name}</p>

<p><strong>Rent:</strong> <span className={office1.rent > 60000 ? 'high' : 'low'}>{office1.rent}</span></p>

<p><strong>Address:</strong> {office1.address}</p>

<h2>Available Office Spaces</h2>

<ul>

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

<li key={index}>

<p><strong>Name:</strong> {office.name}</p>

<p><strong>Rent:</strong> <span className={office.rent > 60000 ? 'high' : 'low'}>{office.rent}</span></p>

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

<hr />

</li>

))}

</ul>

</div>

);

}

export default App;

**App.css**

.App {

font-family: Arial, sans-serif;

padding: 20px;

background-color: #f3f3f3;

}

h1,

h2 {

color: #333;

}

img {

margin: 20px 0;

border-radius: 10px;

}

.low {

color: red;

font-weight: bold;

}

.high {

color: green;

font-weight: bold;

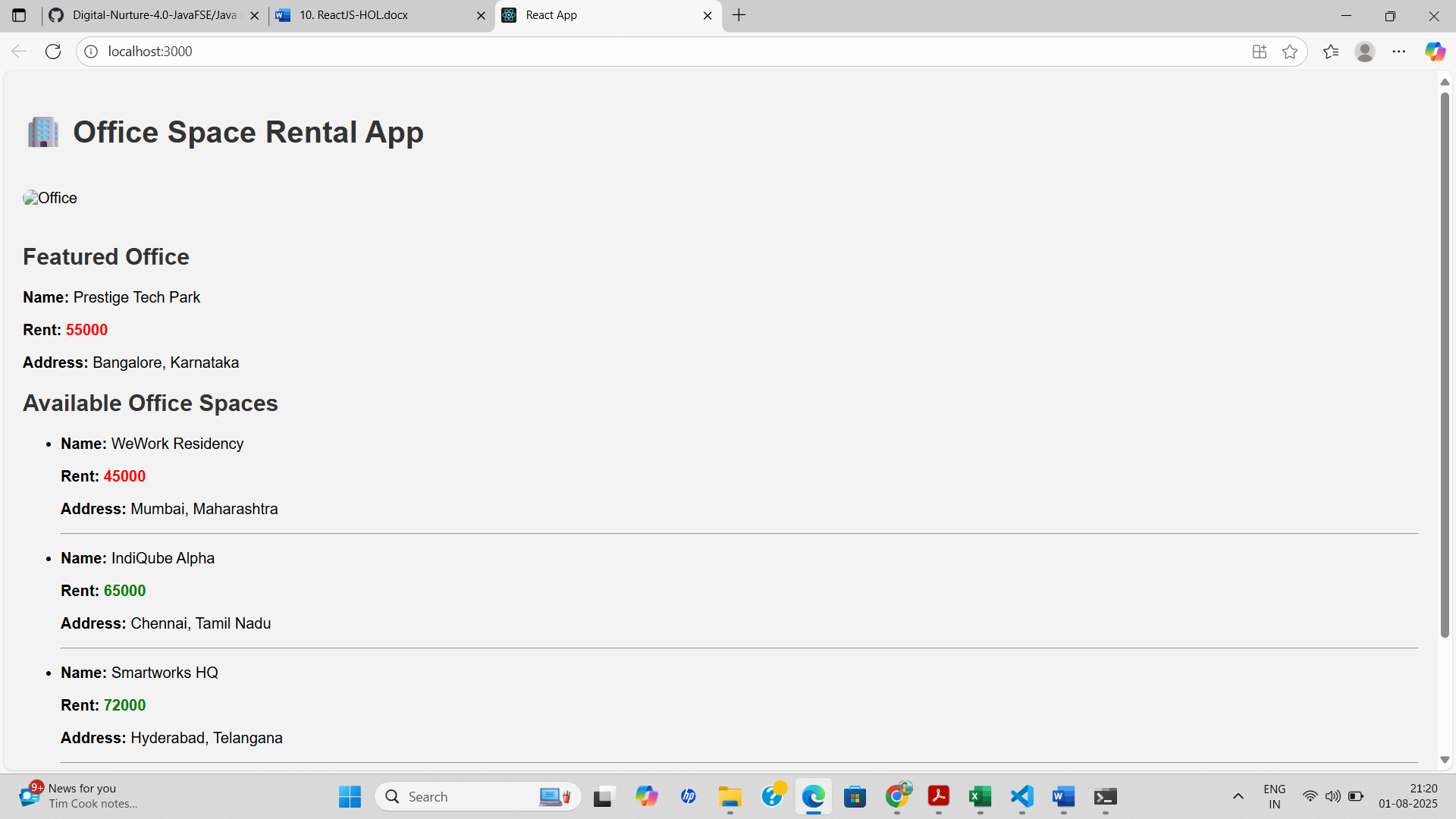
}

li {

margin-bottom: 15px;

}

**Output:**



**Exercise - 11**

**1. React events:**

* React events are JavaScript events wrapped in a cross-browser wrapper called SyntheticEvent.
* They behave similarly to native DOM events but work consistently across all browsers.

**2. Event handlers:**

* Event handlers are functions that are triggered in response to events like clicks, form submissions, keypresses, etc.
* In React, they are usually passed as props to elements, e.g., onClick={handleClick}.

**3. Synthetic event:**

* A SyntheticEvent is React’s wrapper around the native browser event.
* It normalizes event behavior across different browsers and provides the same interface for all events.

**4. Identify React event naming convention:**

* Event names in React use camelCase (not lowercase as in HTML).
* Example: onClick, onChange, onSubmit
* Event handlers are typically passed as function references (not strings).

**Explanation**

* A functional component named CurrencyConverter is created using React hooks (useState) to manage form data.
* The user inputs an amount in Indian Rupees and selects a target currency (Euro).
* Upon submitting the form, a function handleSubmit is triggered.
* Inside this function, the conversion is calculated using the fixed rate (₹1 = €0.0125).
* A popup is shown using alert() to display the converted Euro amount to the user.

**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 />

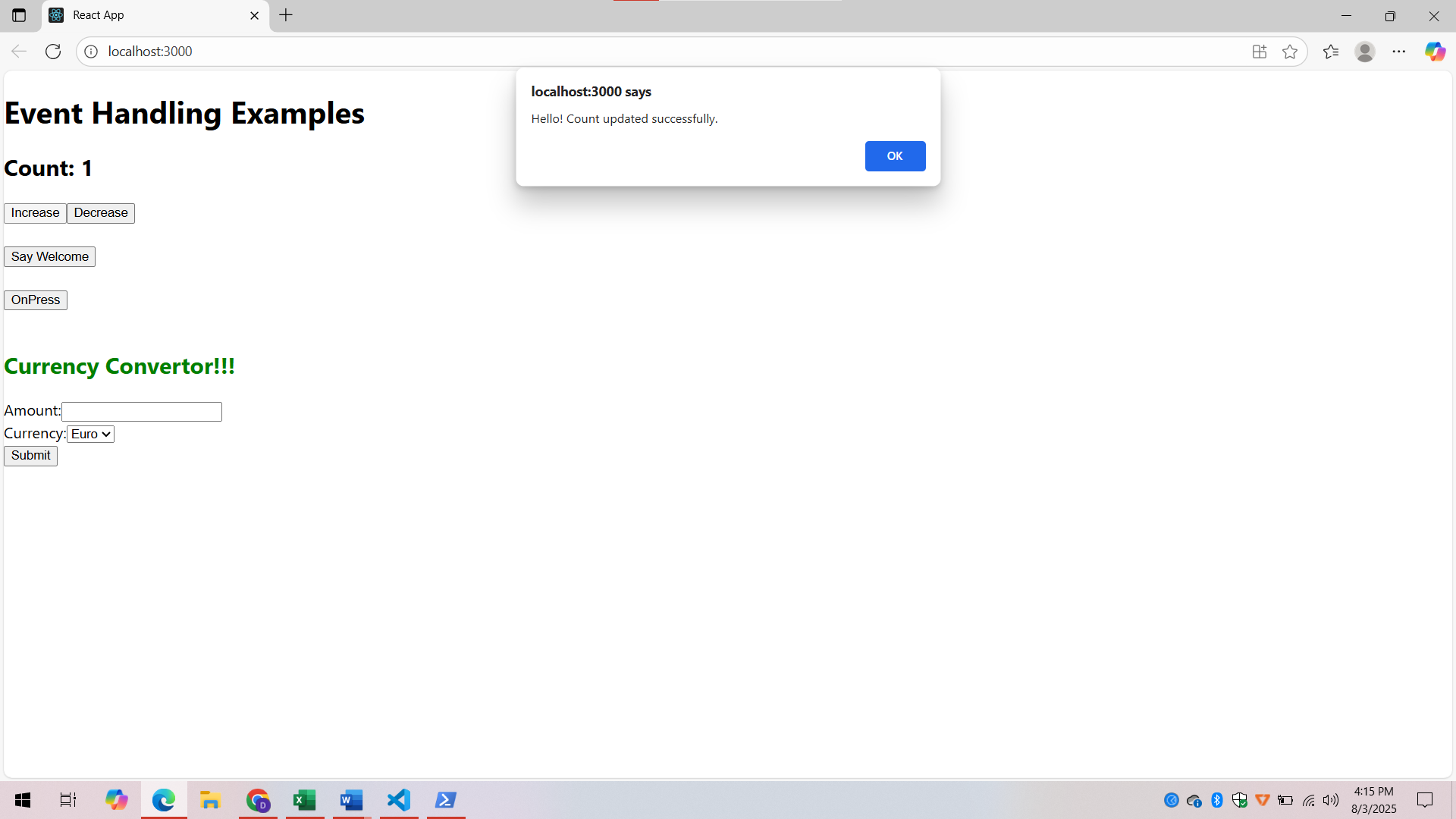
</div>

);

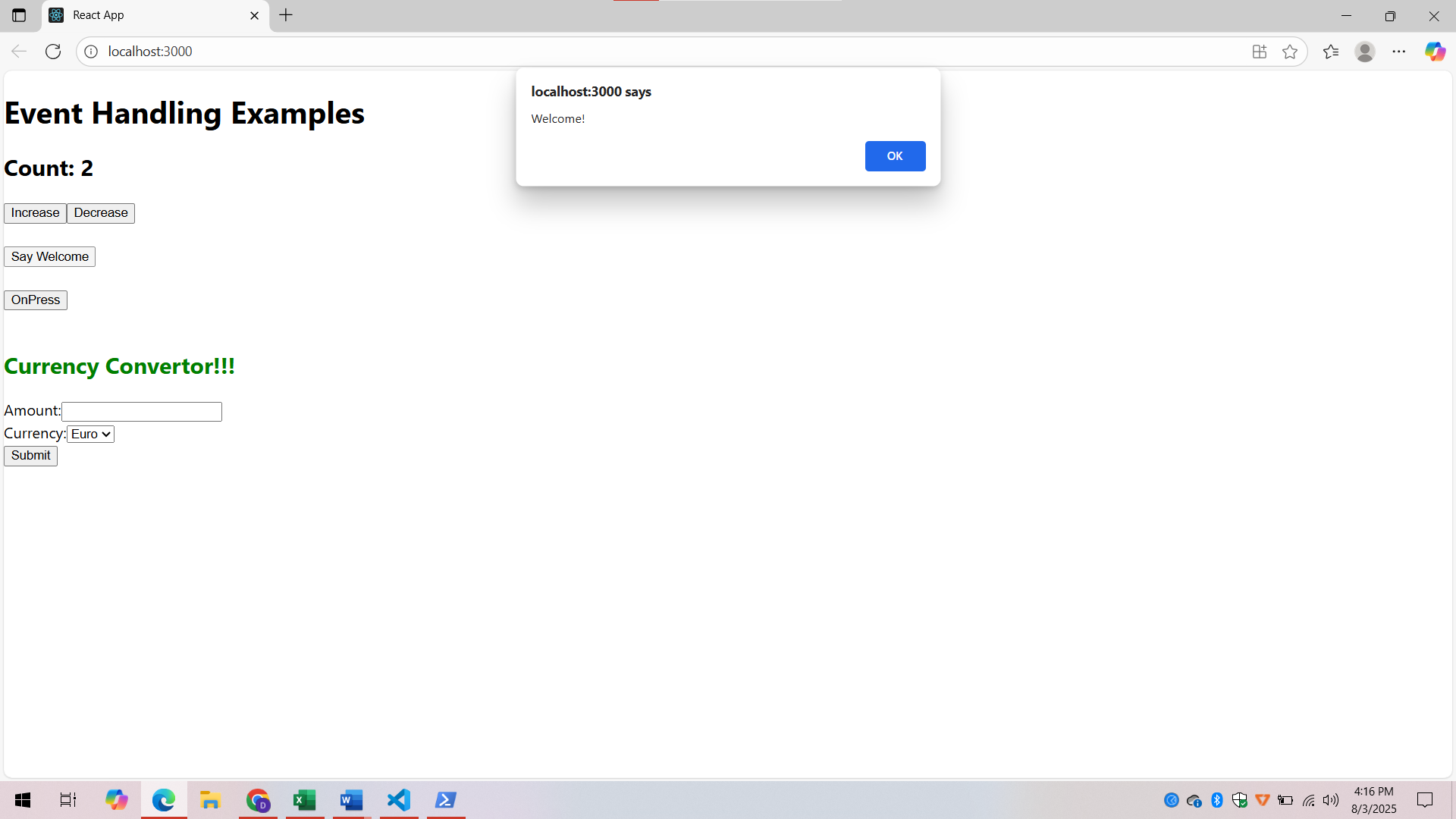
}

export default App;

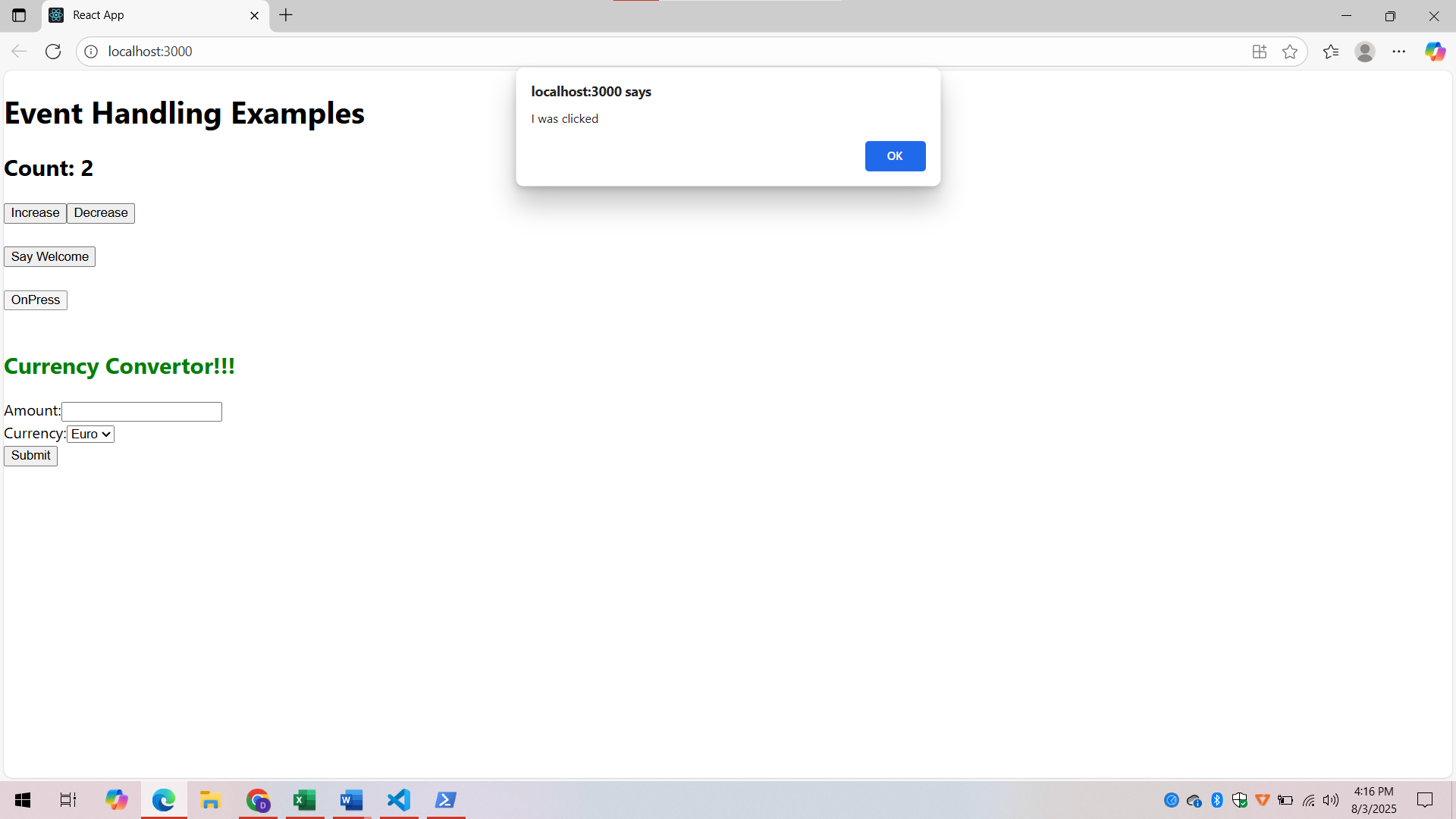
**Output:**

When Currency is converted and increment is clicked,

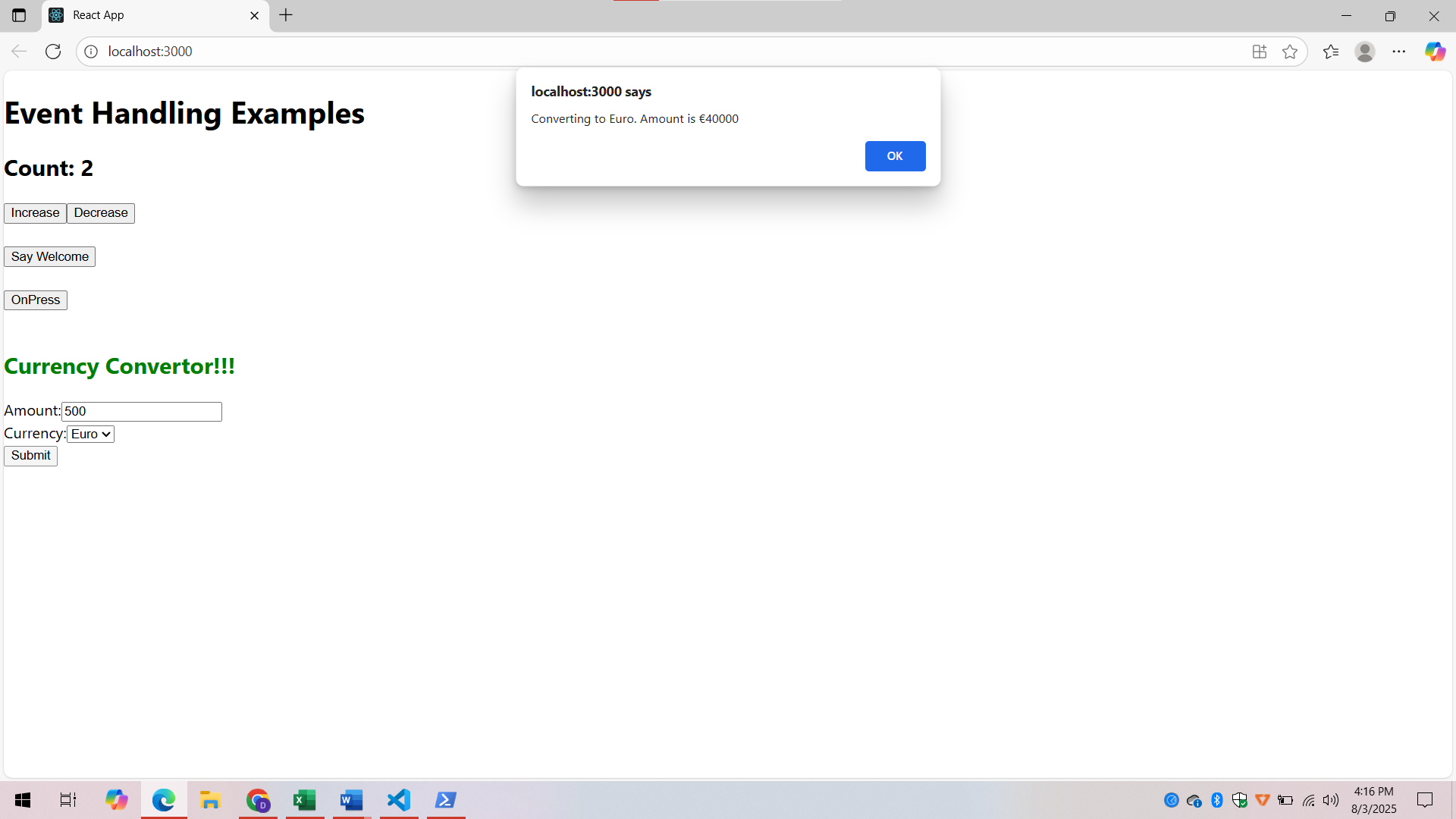
When “Say Welcome” button is clicked,



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



When currency is converted into Euro,



**Exercise - 12**

**1. Conditional rendering in React:**

* Conditional rendering is the technique of rendering different UI elements based on specific conditions (like login status or data availability).
* It uses JavaScript expressions such as ternary (? :), logical AND (&&), or if-else inside functions or components.

**2. Element variables:**

* Element variables are variables that can store JSX elements.
* You can use them to conditionally assign different JSX content and render them.

let message;

if (isLoggedIn) {

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

} else {

message = <h1>Please Sign In</h1>;

}

**3. Prevent components from rendering:**

* You can return null from a component to prevent it from rendering any UI.

if (!showComponent) {

return null;

}

* Alternatively, use conditional logic to exclude components from being part of the render tree.

**Explanation:**

* Run npx create-react-app ticketbookingapp to set up the React project and open it in VS Code.
* Build two components: GuestPage to display flight details and UserPage for booking tickets.
* Use the useState hook to manage a login state variable called isLoggedIn.
* Add Login and Logout buttons to toggle the isLoggedIn value.
* Conditionally render either the UserPage or GuestPage based on login status using the expression: {isLoggedIn ? <UserPage /> : <GuestPage />}.

**Code**

**Guest.js**

import React from 'react';

import FlightList from './FlightList';

function Guest({ onLogin }) {

return (

<div>

<h2>Welcome, Guest!</h2>

<button onClick={onLogin}>Login</button>

<FlightList />

</div>

);

}

export default Guest;

**User.js**

import React from 'react';

import FlightList from './FlightList';

function User({ onLogout }) {

return (

<div>

<h2>Welcome, User!</h2>

<button onClick={onLogout}>Logout</button>

<FlightList />

<button style={{ backgroundColor: "green", color: "white" }}>Book Ticket</button>

</div>

);

}

export default User;

**FlightList.js**

import React from 'react';

const flights = [

{ id: 1, name: 'Air India', from: 'Delhi', to: 'Mumbai' },

{ id: 2, name: 'IndiGo', from: 'Chennai', to: 'Bangalore' },

{ id: 3, name: 'SpiceJet', from: 'Hyderabad', to: 'Kolkata' },

];

function FlightList() {

return (

<div>

<h3>Flight Details</h3>

<ul>

{flights.map(flight => (

<li key={flight.id}>

✈ {flight.name} — {flight.from} to {flight.to}

</li>

))}

</ul>

</div>

);

}

export default FlightList;

**App.js**

import React, { useState } from 'react';

import Guest from './components/Guest';

import User from './components/User';

function App() {

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

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

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

return (

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

<h1>✈️ Ticket Booking App</h1>

{isLoggedIn ? (

<User onLogout={handleLogout} />

) : (

<Guest onLogin={handleLogin} />

)}

</div>

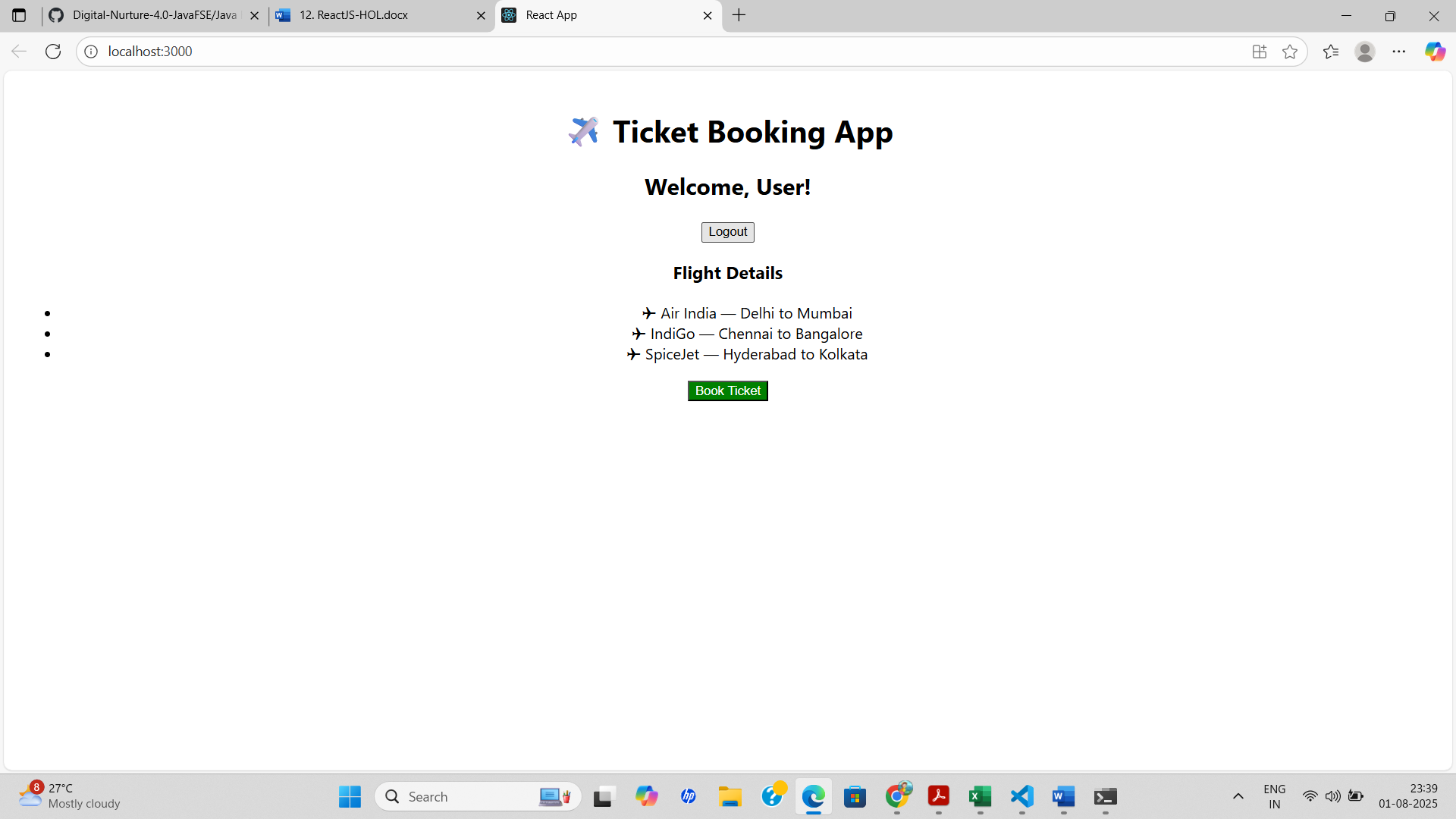
);

}

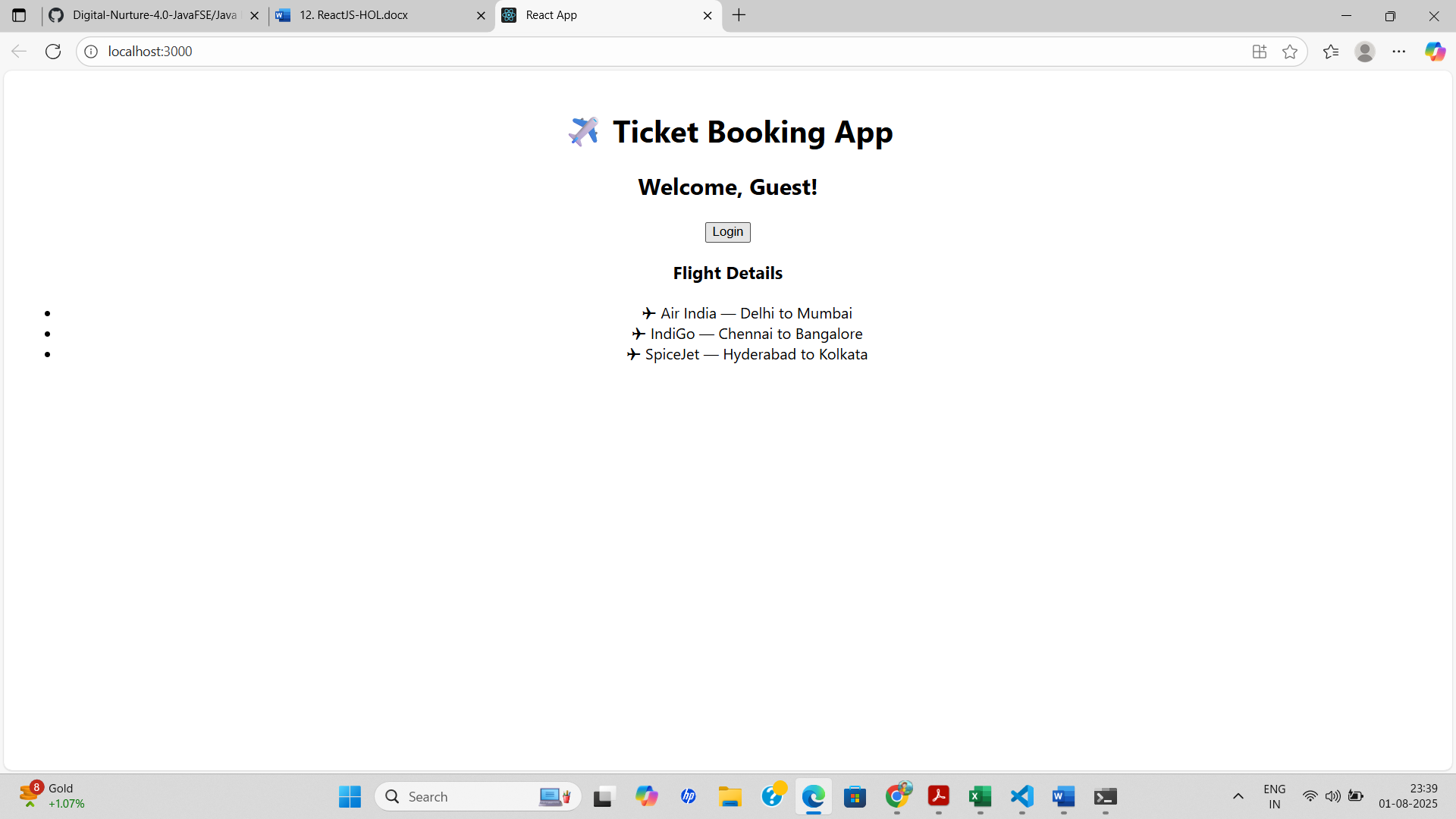
export default App;

**Output**

Login button → switches to user page with "Book Ticket"



Logout button → returns to guest view



**Exercise - 13**

**1. Various ways of conditional rendering:**

\* Ternary operator: {isLoggedIn ? <Dashboard /> : <Login />}

\* Logical AND (`&&`) operator: {isAdmin && <AdminPanel />}

\* `if-else` statements inside functions.

\* Element variables to store JSX conditionally.

\* Early return: Using `return null` to skip rendering.

**2. Render multiple components:**

\* Return multiple components using:

\* React Fragments:

<>

<Header />

<Content />

<Footer />

</>

\* Arrays of elements with keys.

\* Group components inside a parent container.

**3. Define list component:**

\* A component designed to render a collection of items (like an array) using loops like `map()`.

\* It displays repeated UI for each item, e.g., list of products, posts, or users.

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

\* Keys are unique identifiers assigned to list items in React.

\* They help React identify which items have changed, been added, or removed.

\* Must be stable, predictable, and unique (like IDs or array indexes).

**5. To extract components with keys:**

\* When mapping lists, you can move the JSX for each item into a child component and pass a `key` prop:

items.map(item => <ListItem key={item.id} value={item} />)

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

\* `map()` is a JavaScript array method used in React to create elements dynamically:

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

\* In React, `map()` is often used inside JSX to render lists from data arrays.

**Explanation**

\* The app uses conditional rendering to display one section at a time based on user choice.

\* A state variable tracks which component (`Books`, `Blogs`, or `Courses`) should be displayed.

\* Inside the child component, `if` statements determine which JSX block to render.

\* Buttons in the main component allow the user to switch views by updating the state.

\* This approach improves the UI by rendering only the selected component, optimizing performance.

**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>

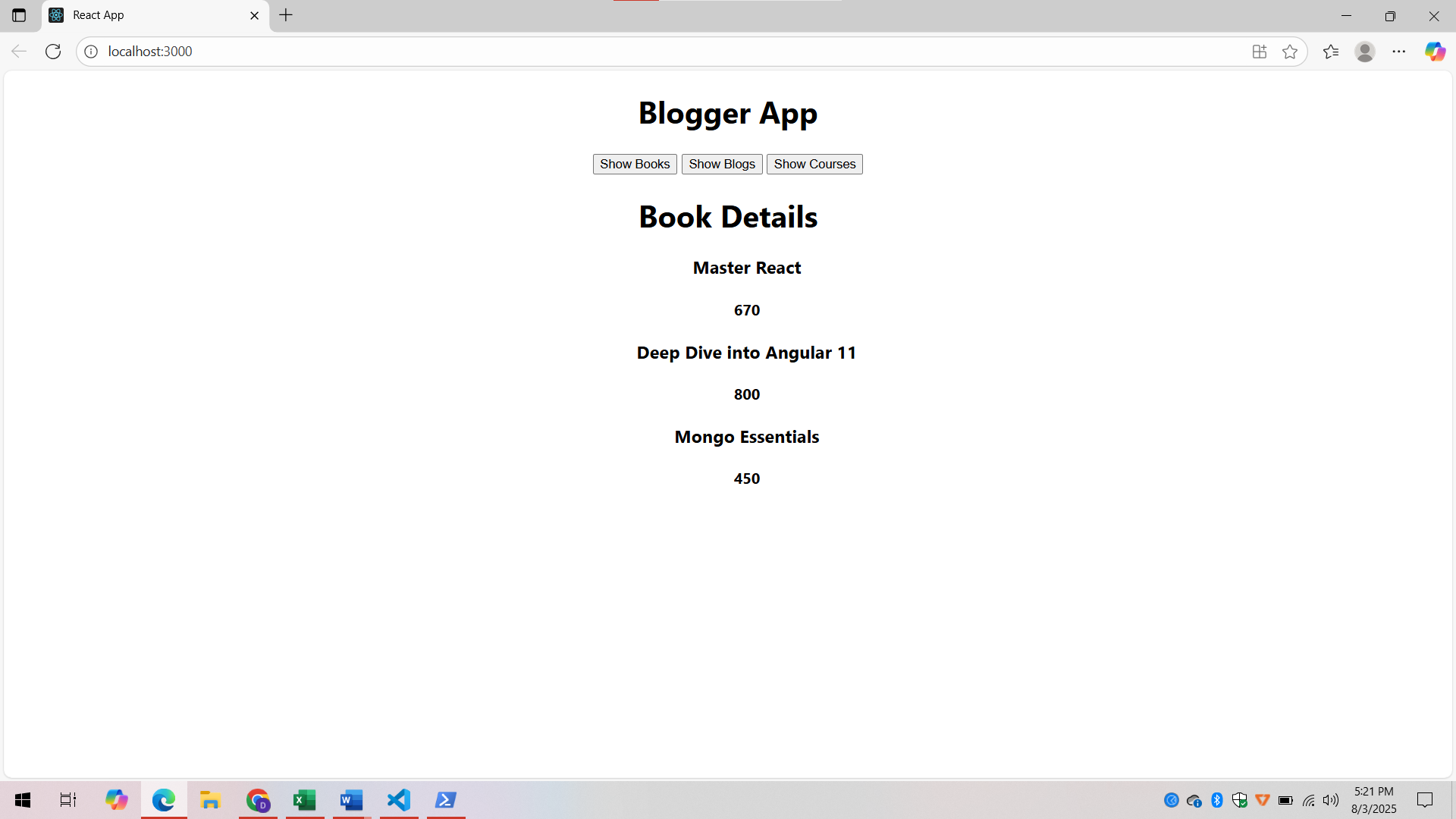
<Details books={books} blogs={blogs} courses={courses} view={view} />

</div> );}

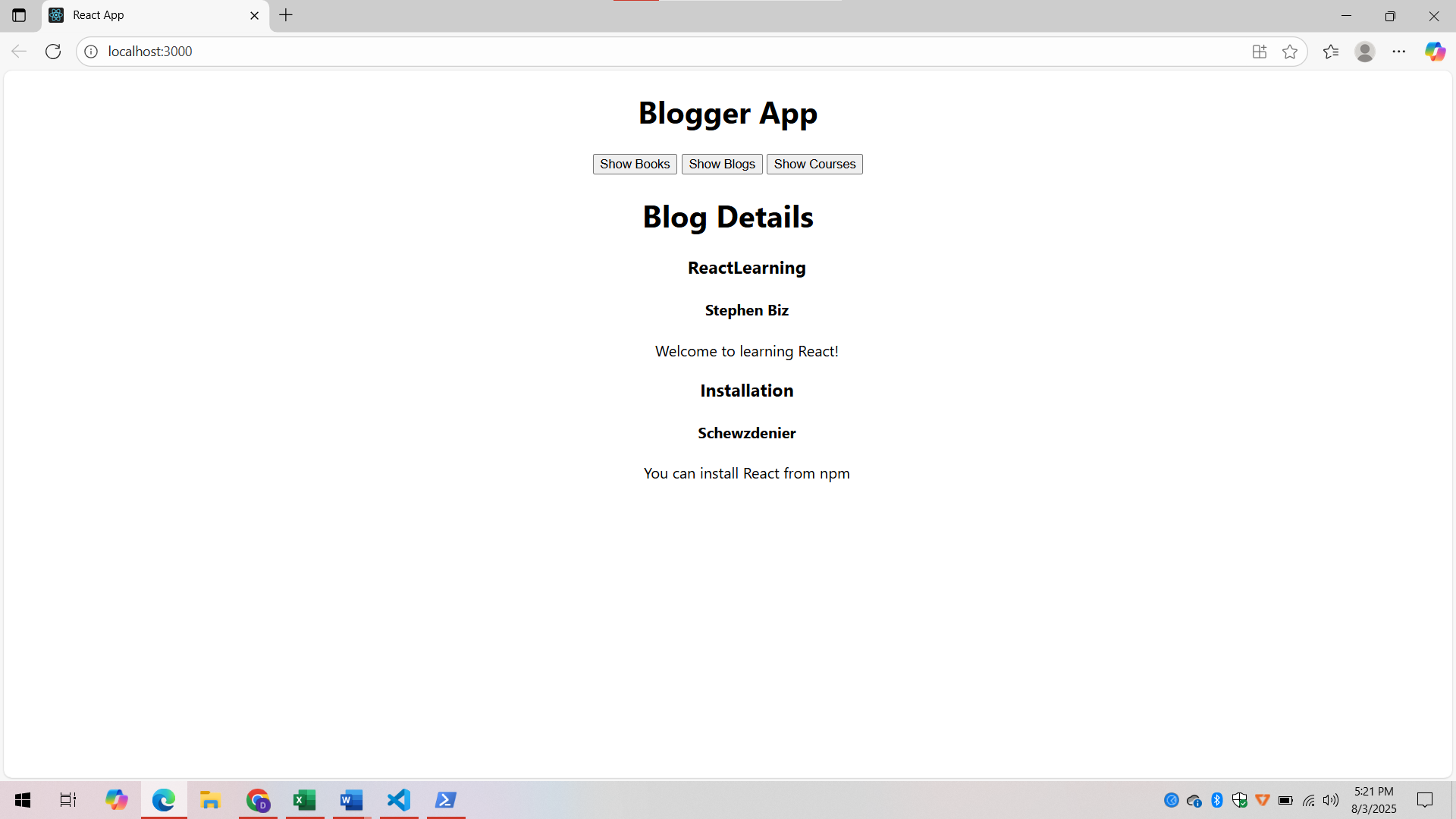
export default App;

**Output**

When “Show Books” is clicked,



When “Show Blogs” is clicked,



When “Show Courses” is clicked,

