**1. ReactJS-HOL**

**List the features of ES6**  
ES6 introduced several improvements in JavaScript such as:

* let and const for better variable declarations
* Arrow functions for cleaner syntax
* Destructuring to extract values from arrays and objects
* Template literals for easier string formatting
* Classes and inheritance
* Spread and rest operators
* Default function parameters
* Promises for asynchronous programming
* Map and Set for efficient data structures

**Explain JavaScript let**  
The let keyword is used to declare variables that are block-scoped. Unlike var, which is function-scoped, let only exists within the block it was defined in. It helps avoid issues with hoisting and redeclaration.

**Identify the differences between var and let**  
var is function-scoped and can be redeclared within the same function. It is hoisted and initialized as undefined. On the other hand, let is block-scoped, cannot be redeclared in the same scope, and while it is hoisted, it is not initialized.

**Explain JavaScript const**  
The const keyword is used to define constants. Once assigned, its value cannot be changed or reassigned. It is also block-scoped like let. If the value is an object or array, the contents can still be modified but the reference cannot be changed.

**Explain ES6 class fundamentals**  
ES6 introduced a class syntax to write object-oriented code. Classes can include constructors and methods, and the this keyword refers to the instance of the class. It provides a more structured and cleaner way to create reusable components.

**Explain ES6 class inheritance**  
In ES6, one class can inherit from another using the extends keyword. The child class can use the super() function to call the parent class's constructor. This makes code reusable and follows object-oriented principles.

**Define ES6 arrow functions**  
Arrow functions provide a shorter syntax to write functions. They do not have their own this binding and inherit it from the surrounding context. They are especially useful for callbacks and concise logic.

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

**Identify set() and map()**

* Set is a built-in object that stores unique values. If you add a duplicate, it automatically ignores it.  
  Example:  
  const uniqueValues = new Set([1, 2, 2, 3]);
* Map is a collection of key-value pairs where both keys and values can be of any data type.  
  Example:  
  const scores = new Map(); scores.set("Virat", 90);

**Commands:**

npm install -g create-react-app

npx create-react-app cricketapp

cd cricketapp

**Project Name: cricketapp**

**Code:**

**App.js**import React from "react";

import ListofPlayers from "./ListofPlayers";

import IndianPlayers from "./IndianPlayers";

function App() {

  const flag = true;

  return (

    <div className="App">

      {flag ? <ListofPlayers /> : <IndianPlayers />}

    </div>

  );

}

export default App;

**IndianPlayers.js**

import React from "react";

const IndianPlayers = () => {

  const oddPlayers = ["Sachin1", "Virat3", "Yuvraj5"];

  const evenPlayers = ["Dhoni2", "Rohit4", "Raina6"];

  const mergedPlayers = [

    "Mr. First Player",

    "Mr. Second Player",

    "Mr. Third Player",

    "Mr. Fourth Player",

    "Mr. Fifth Player",

    "Mr. Sixth Player"

  ];

  return (

    <div>

      <h2>Odd Players</h2>

      <ul>

        <li>First : {oddPlayers[0]}</li>

        <li>Third : {oddPlayers[1]}</li>

        <li>Fifth : {oddPlayers[2]}</li>

      </ul>

      <hr />

      <h2>Even Players</h2>

      <ul>

        <li>Second : {evenPlayers[0]}</li>

        <li>Fourth : {evenPlayers[1]}</li>

        <li>Sixth : {evenPlayers[2]}</li>

      </ul>

      <hr />

      <h2>List of Indian Players Merged:</h2>

      <ul>

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

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

        ))}

      </ul>

    </div>

  );

};

export default IndianPlayers;

**ListofPlayers.js**

import React from "react";

const ListofPlayers = () => {

  const players = [

    { name: "Mr. Jack", score: 50 },

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

    { name: "Mr. John", score: 40 },

    { name: "Mr. Amit", score: 61 },

    { name: "Mr. Elizabeth", 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 playersBelow70 = players.filter(player => player.score < 70);

  return (

    <div>

      <h2>List of Players</h2>

      <ul>

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

          <li key={index}>

            {player.name} {player.score}

          </li>

        ))}

      </ul>

      <hr />

      <h2>List of Players having Scores Less than 70</h2>

      <ul>

        {playersBelow70.map((player, index) => (

          <li key={index}>

            {player.name} {player.score}

          </li>

        ))}

      </ul>

    </div>

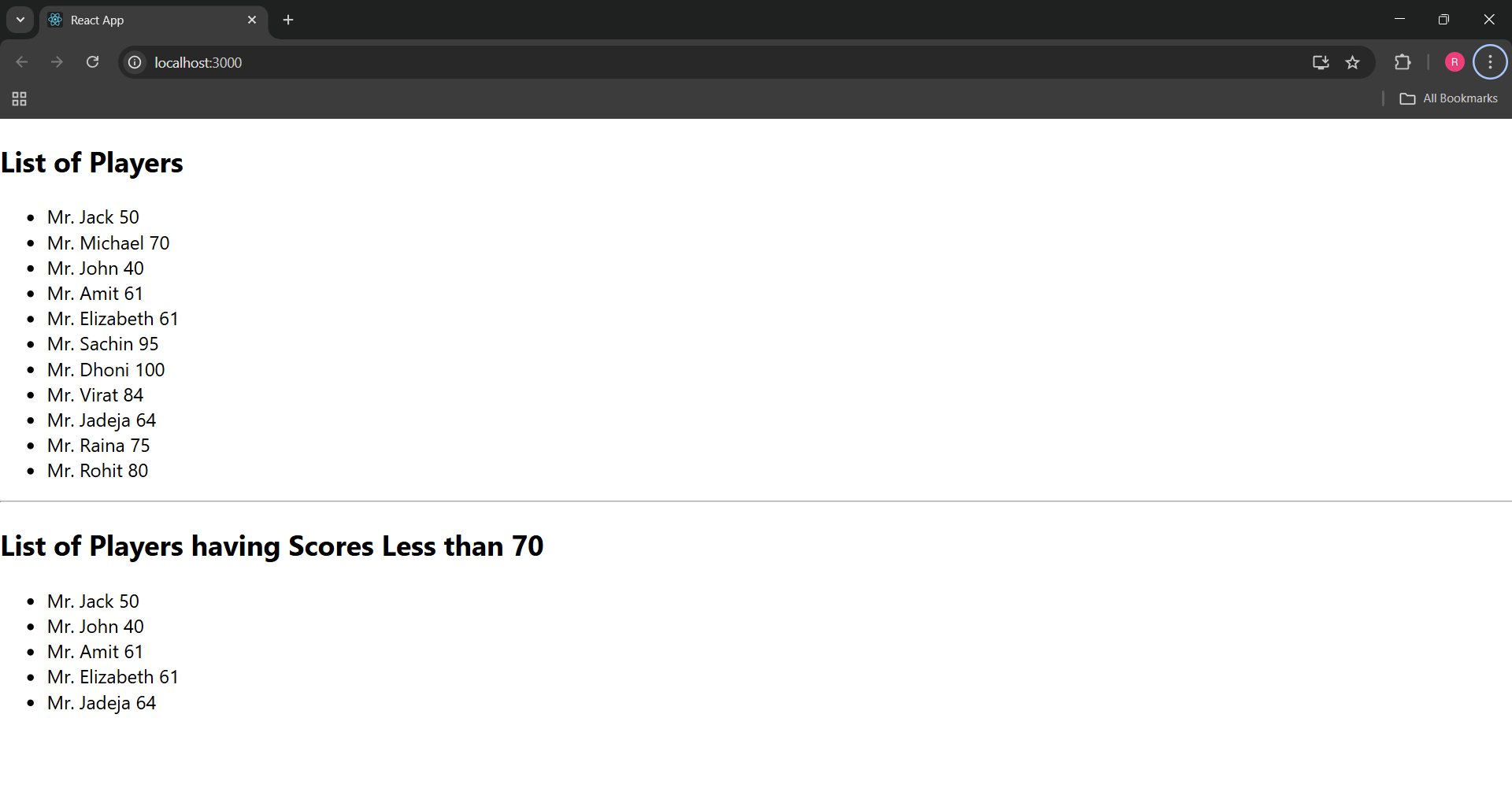
  );

};

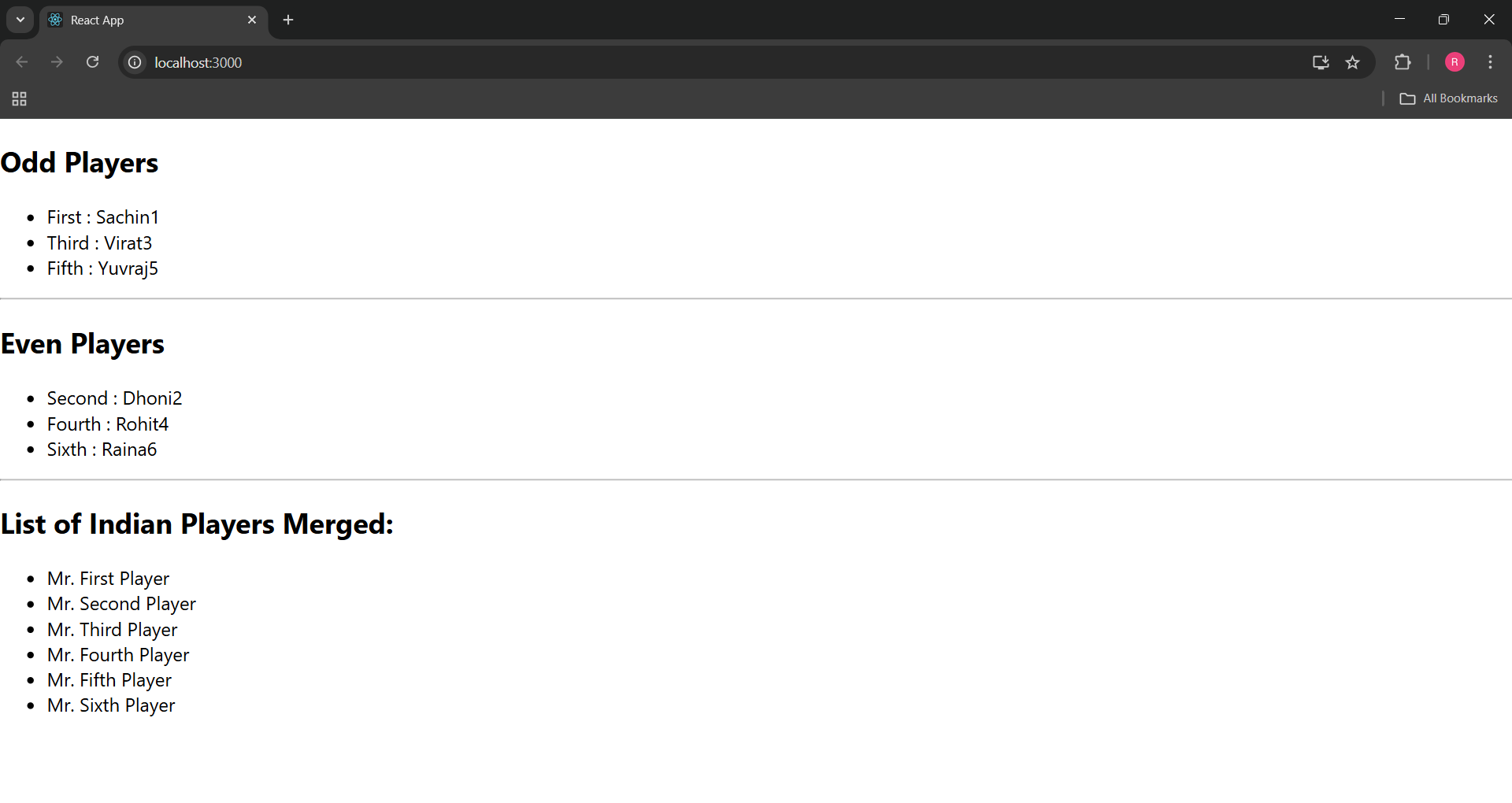
export default ListofPlayers;

**Output:**

**When flag = true**

****

**When flag = false**



**2. ReactJS-HOL**

**Define JSX**JSX stands for JavaScript XML. It is a syntax extension used in React that allows writing HTML-like code inside JavaScript. It makes the user interface code easier to write and understand by combining structure and logic in a single block.

**Explain ECMAScript**ECMAScript is the standardized version of JavaScript. It defines the rules, syntax, and features of the language. React and modern JavaScript use ECMAScript standards to ensure consistency and improve functionality across browsers and platforms.

**Explain React.createElement**This is a method provided by React to create virtual DOM elements without using JSX. JSX is internally converted to calls of this method. It takes the type of element, properties, and children as inputs and returns a representation of the element in memory**.**

**Explain how to create React nodes with JSX**React nodes can be created using JSX by writing HTML-like tags directly within JavaScript code. These nodes can represent headings, images, paragraphs, or any valid HTML elements and are later rendered into the DOM using React methods.

**Define how to render JSX to the DOM**JSX elements can be rendered into the DOM using React’s rendering method. This allows the dynamic elements created using JSX to be inserted into specific parts of the web page, usually into a root div.

**Explain how to use JavaScript expressions in JSX**JavaScript expressions such as variables, function calls, and simple calculations can be embedded within JSX. This is done using curly braces and allows dynamic content to be displayed based on logic.

**Explain how to use inline CSS in JSX**Inline styling in JSX is applied using a special style attribute that accepts a JavaScript object. The property names use camelCase instead of traditional CSS syntax. This method allows dynamic styling directly inside the component based on conditions.

**Commands:**

npx create-react-app officespacerentalapp

cd officespacerentalapp

**Project Name: officespacerentalapp**

**Code:**

**App.js**

import React from "react";

function App() {

const mainOffice = {

name: "DBS",

rent: 50000,

address: "Chennai",

image:

"https://images.unsplash.com/photo-1570129477492-45c003edd2be?auto=format&fit=crop&w=800&q=60"

};

const officeList = [

{

name: "Startup Arena",

rent: 45000,

address: "Bengaluru"

},

{

name: "CoSpace Hub",

rent: 68000,

address: "Pune"

},

{

name: "Prime Office",

rent: 72000,

address: "Hyderabad"

}

];

const rentStyle = (amount) => {

return {

color: amount < 60000 ? "red" : "green",

fontWeight: "bold"

};

};

return (

<div style={{ fontFamily: "Arial", padding: "30px", maxWidth: "600px" }}>

<h1 style={{ fontWeight: "bold" }}>

Office Space , at Affordable Range

</h1>

<img

src={mainOffice.image}

alt="Office"

style={{

width: "100%",

height: "300px",

objectFit: "cover",

borderRadius: "8px",

marginBottom: "20px"

}}

/>

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

<p style={rentStyle(mainOffice.rent)}>Rent: Rs. {mainOffice.rent}</p>

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

<hr />

<h2>Other Office Spaces</h2>

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

<div key={index} style={{ marginBottom: "20px" }}>

<h3>Name: {office.name}</h3>

<p style={rentStyle(office.rent)}>Rent: Rs. {office.rent}</p>

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

</div>

))}

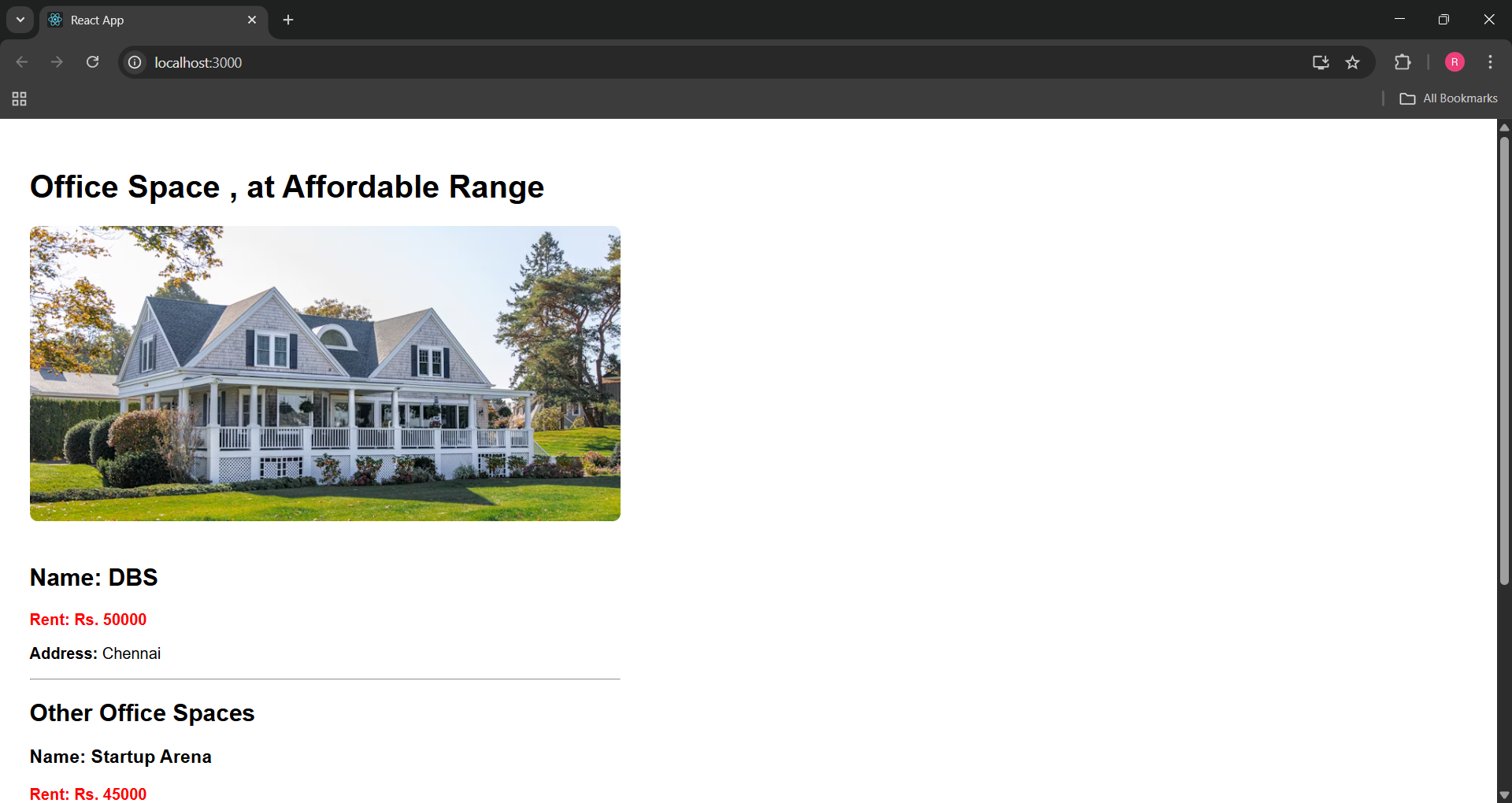
</div>

);

}

export default App;

**Output:**



**3. ReactJS-HOL**

**Explain React events**

React events are objects that represent interactions from the user, such as clicking a button, typing in an input field, or submitting a form. They are handled using functions called event handlers, just like in HTML, but written in camelCase in JSX.

**Explain about event handlers**

An event handler is a function that is triggered when a specific event occurs. In React, these handlers are passed as props to elements. For example, a button can have an onClick event that runs a function when clicked.

Example:

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

**Define Synthetic event**

A synthetic event is a wrapper around the native browser event provided by React. It works identically across all browsers, making React applications more consistent and easier to manage. It pools the event object for performance benefits.

**Identify React event naming convention**

In React, event names are written in **camelCase** rather than lowercase. For instance:

* HTML: onclick, onchange
* React: onClick, onChange

Also, in React, event handler values are **JS functions**.

**Commands:**

npx create-react-app eventexamplesapp

cd eventexamplesapp

**Project Name: eventexamplesapp**

**App.js**

import React, { useState } from "react";

import CurrencyConvertor from "./CurrencyConvertor";

function App() {

  const [count, setCount] = useState(0);

  const increment = () => {

    setCount(prev => prev + 1);

  };

  const decrement = () => {

    setCount(prev => prev - 1);

  };

  const sayWelcome = () => {

    alert("welcome");

  };

  const handleClick = () => {

    alert("You clicked on the button");

  };

  return (

    <div style={{ padding: "30px", fontFamily: "Arial", textAlign: "left" }}>

      <h2>{count}</h2>

      <button onClick={increment} style={{ marginBottom: "5px" }}>Increment</button><br />

      <button onClick={decrement} style={{ marginBottom: "5px" }}>Decrement</button><br />

      <button onClick={sayWelcome} style={{ marginBottom: "5px" }}>Say welcome</button><br />

      <button onClick={handleClick} style={{ marginBottom: "20px" }}>Click on me</button>

      <CurrencyConvertor />

    </div>

  );

}

export default App;

**CurrencyConvertor.js**

import React, { useState } from "react";

function CurrencyConvertor() {

  const [amount, setAmount] = useState("");

  const [currency, setCurrency] = useState("");

  const handleSubmit = () => {

    alert(`Converted ${amount} to ${currency}`);

  };

  return (

    <div>

      <h2 style={{ color: "green", fontWeight: "bold" }}>Currency Convertor!!!</h2>

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

        <label>Amount:&nbsp;</label>

        <input

          type="text"

          value={amount}

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

        />

      </div>

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

        <label>Currency:&nbsp;</label>

        <textarea

          value={currency}

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

        />

      </div>

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

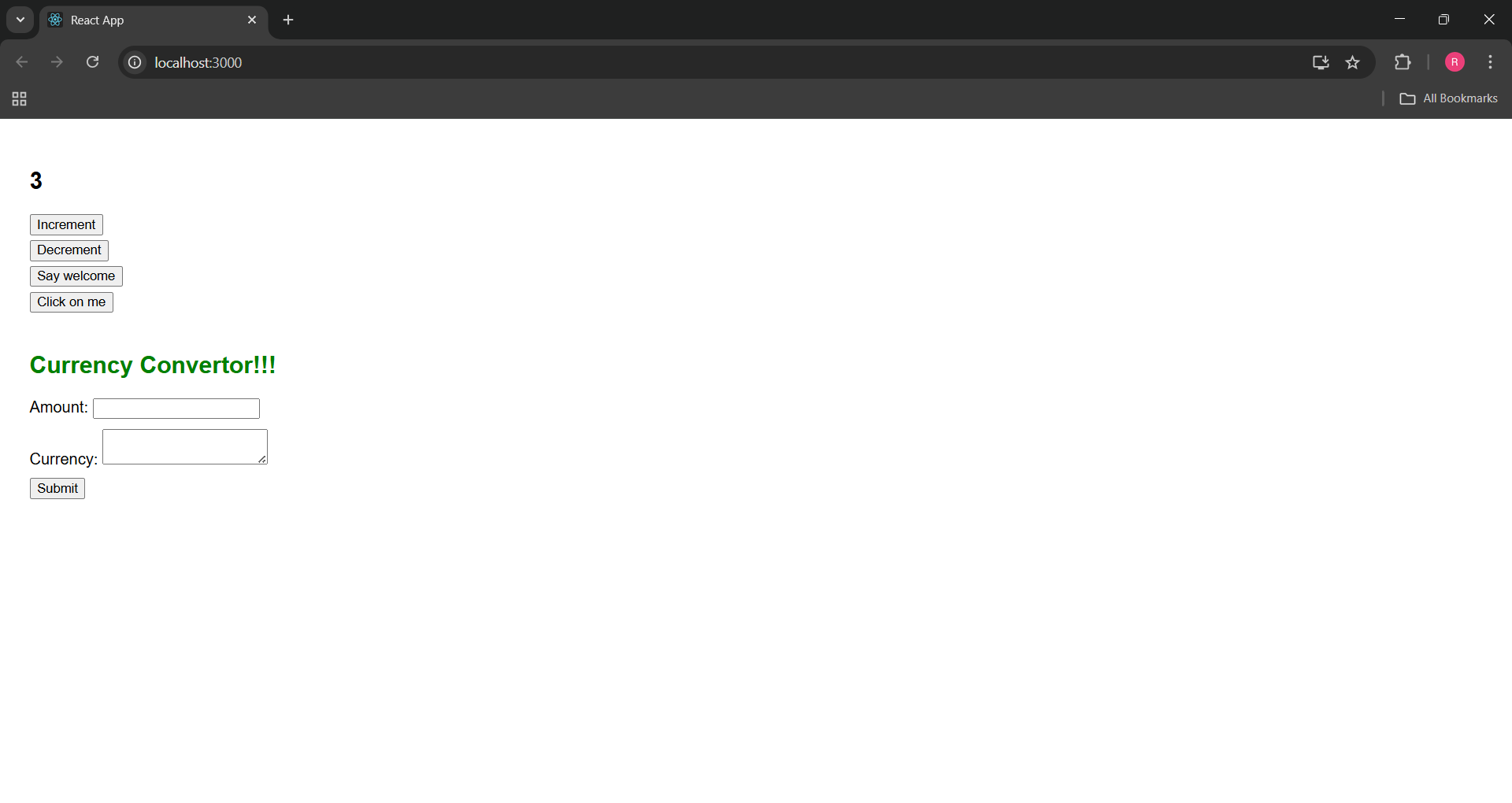
    </div>

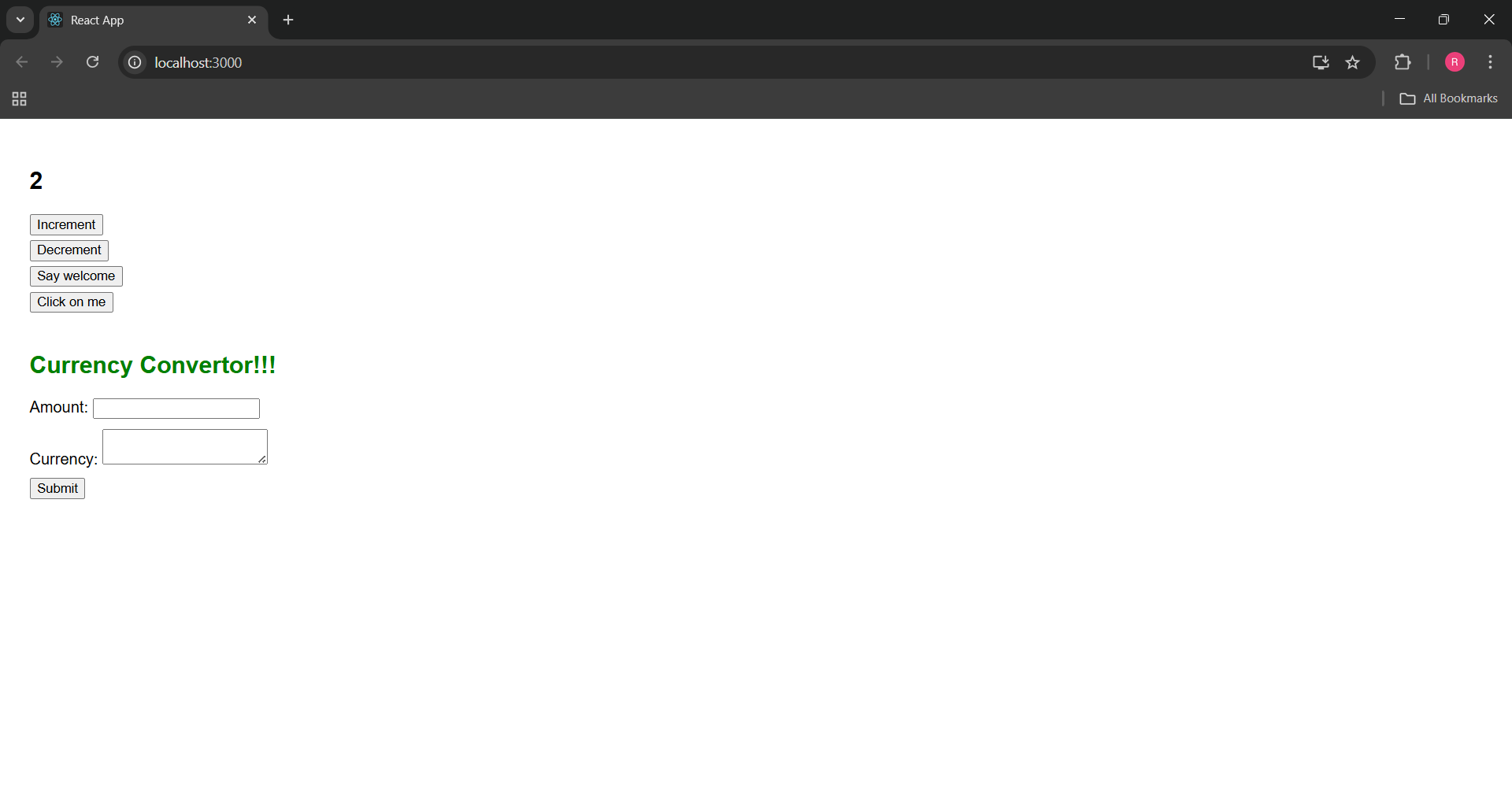
  );

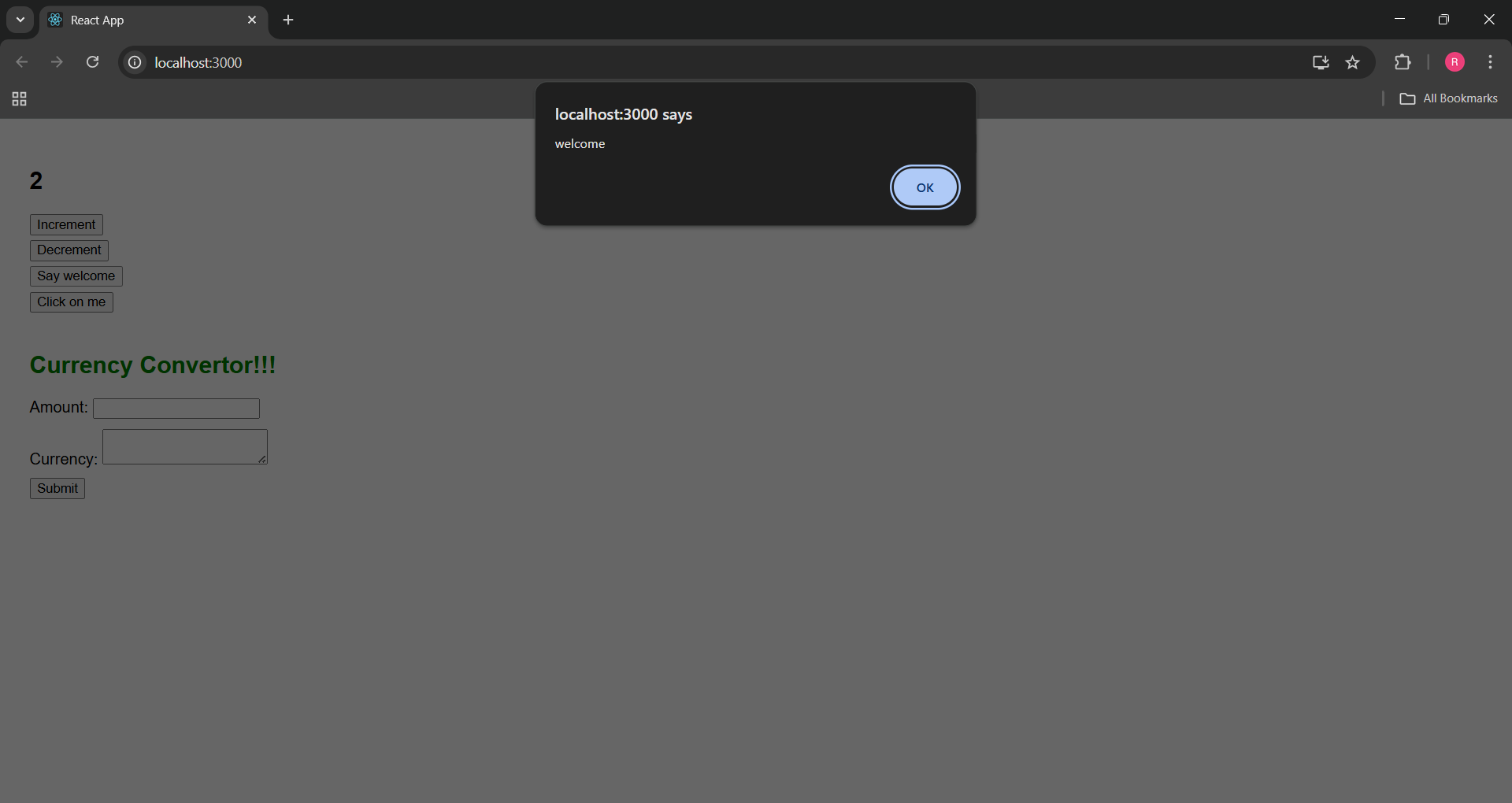
}

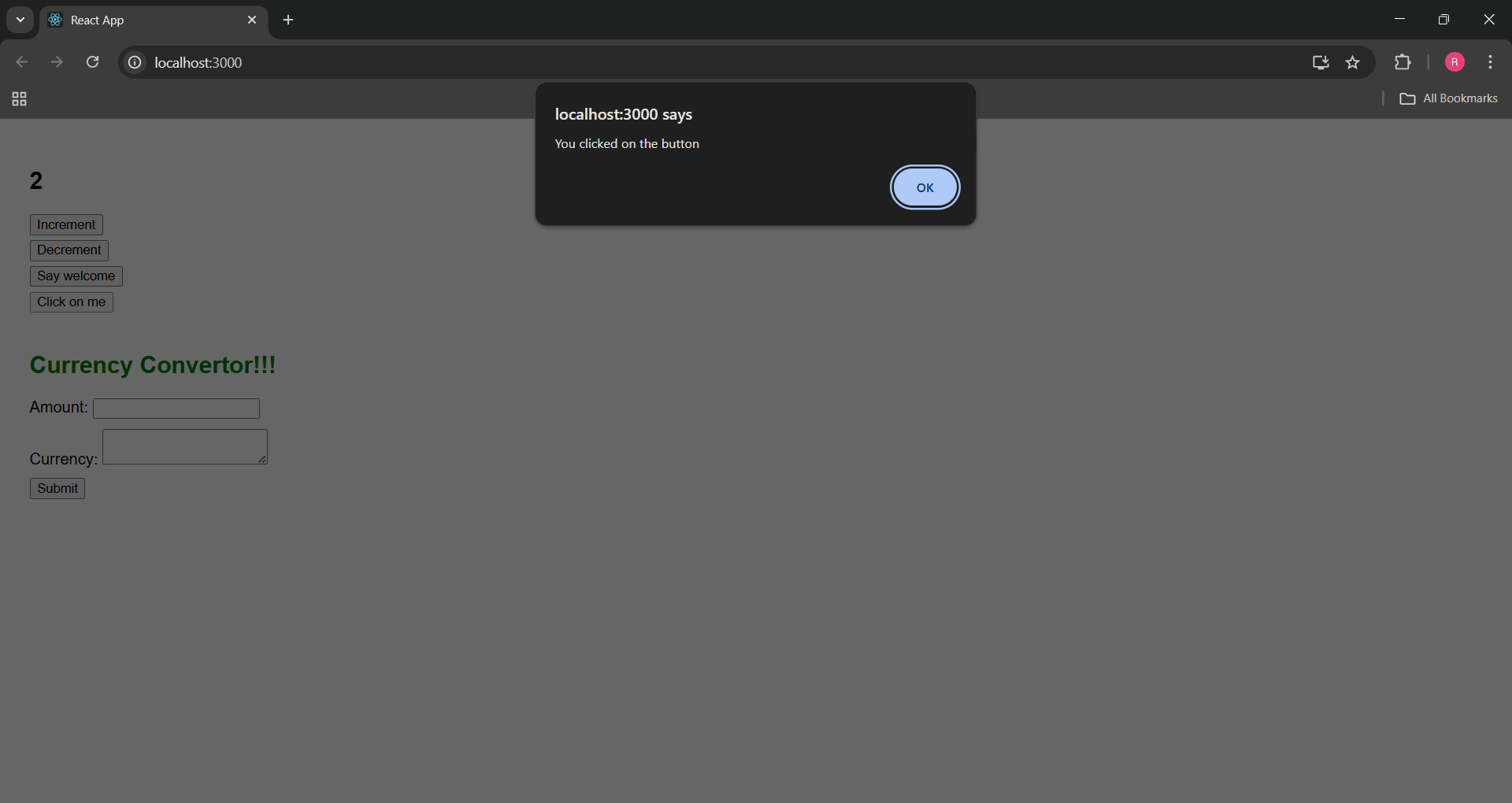
export default CurrencyConvertor;

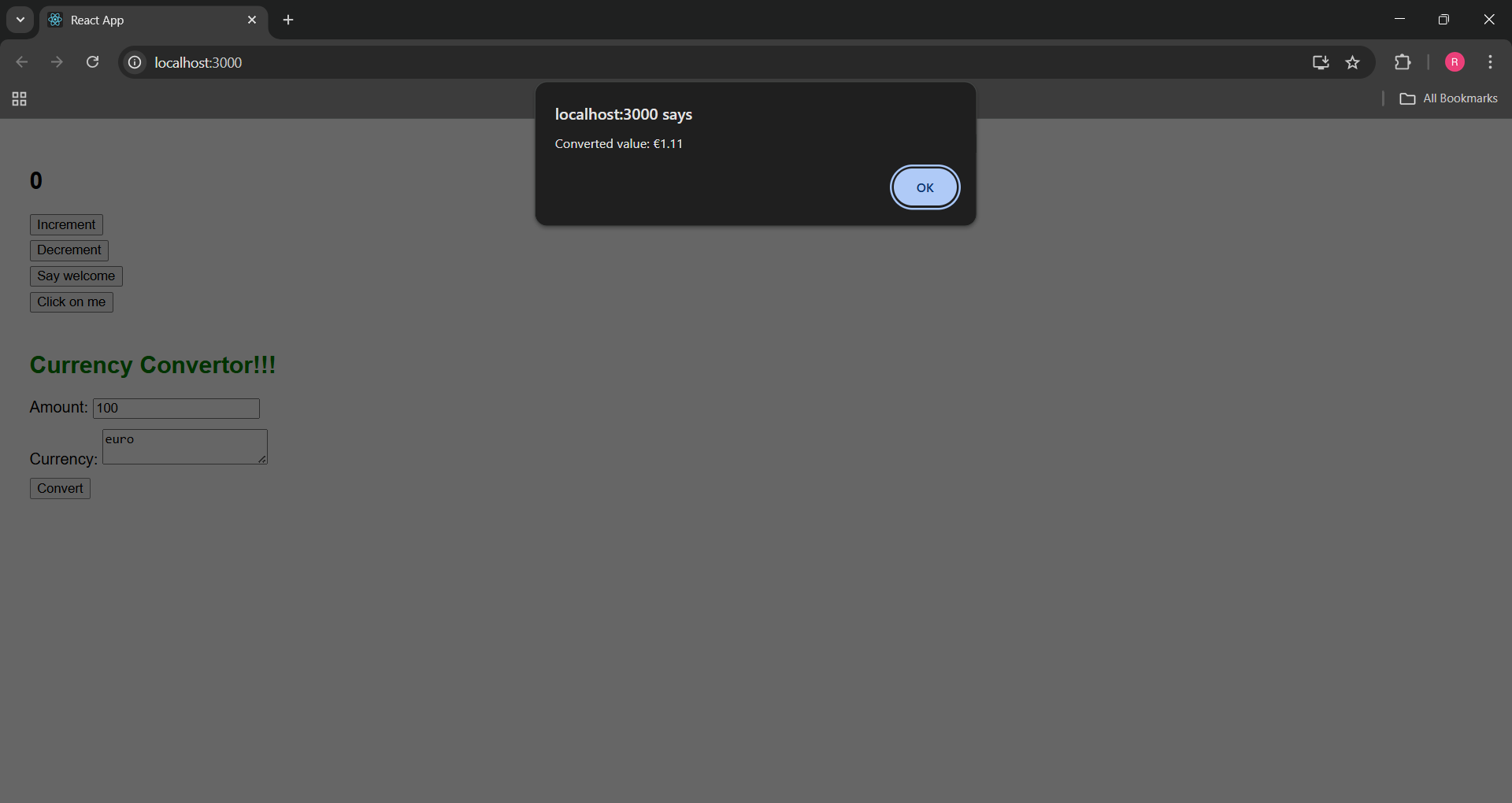
**Output:**











**4. ReactJS-HOL**

**Explain about conditional rendering in React**

Conditional rendering in React means showing different UI elements based on certain conditions like user login, permissions, or data availability. It's similar to if-else statements in regular JavaScript but used inside JSX.

**Define Element Variables**

Element variables are used to store JSX elements in a variable, which can then be conditionally assigned and rendered in the return statement. This improves readability and logic control in the render flow.

**Explain How to Prevent Component from Rendering**

We can prevent a component from rendering by using a condition like

{condition && <Component />}

Or by returning null from the component's render logic. React will skip rendering that part of the UI.

**Commands:**

npx create-react-app eventexamplesapp

cd eventexamplesapp

**Project Name: eventexamplesapp**

**App.js**

import React, { useState } from "react";

import GuestPage from "./GuestPage";

import UserPage from "./UserPage";

function App() {

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

  const handleLogin = () => {

    setIsLoggedIn(true);

    alert("Logged in successfully!");

  };

  const handleLogout = () => {

    setIsLoggedIn(false);

    alert("Logged out successfully!");

  };

  let page;

  let button;

  if (isLoggedIn) {

    page = <UserPage />;

    button = <button onClick={handleLogout}>Logout</button>;

  } else {

    page = <GuestPage />;

    button = <button onClick={handleLogin}>Login</button>;

  }

  return (

    <div style={{ padding: "30px", fontFamily: "Arial", textAlign: "left" }}>

      <h1>Flight Ticket Booking</h1>

      {button}

      {page}

    </div>

  );

}

export default App;

**GuestPage.js**

import React from "react";

function GuestPage() {

  return (

    <div>

      <h2>Guest Page</h2>

      <p>Please log in to book your tickets and view flight details.</p>

    </div>

  );

}

export default GuestPage;

**UserPage.js**

import React from "react";

function UserPage() {

  return (

    <div>

      <h2>User Page</h2>

      <p>Welcome back! You can now book your tickets.</p>

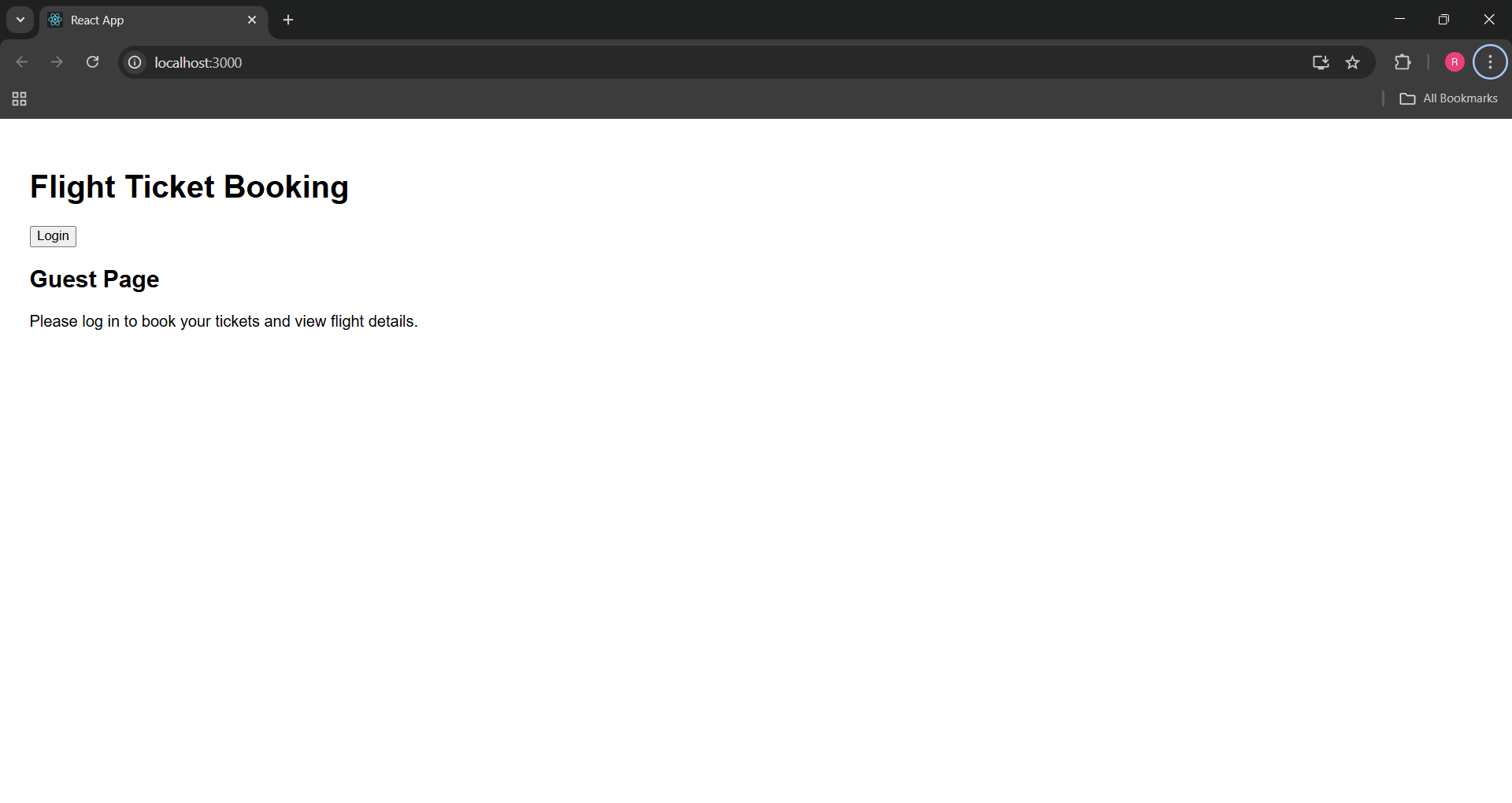
    </div>

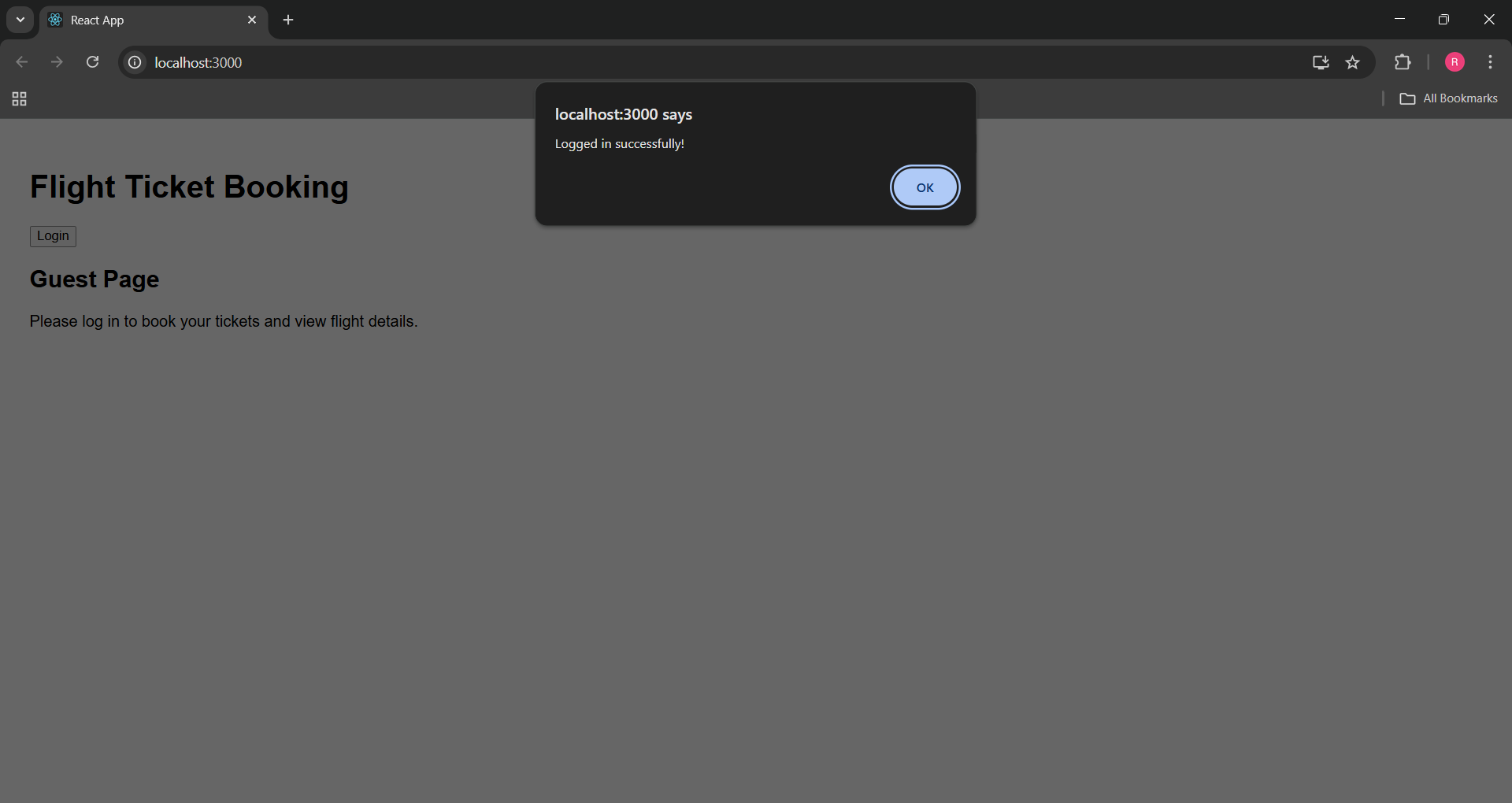
  );

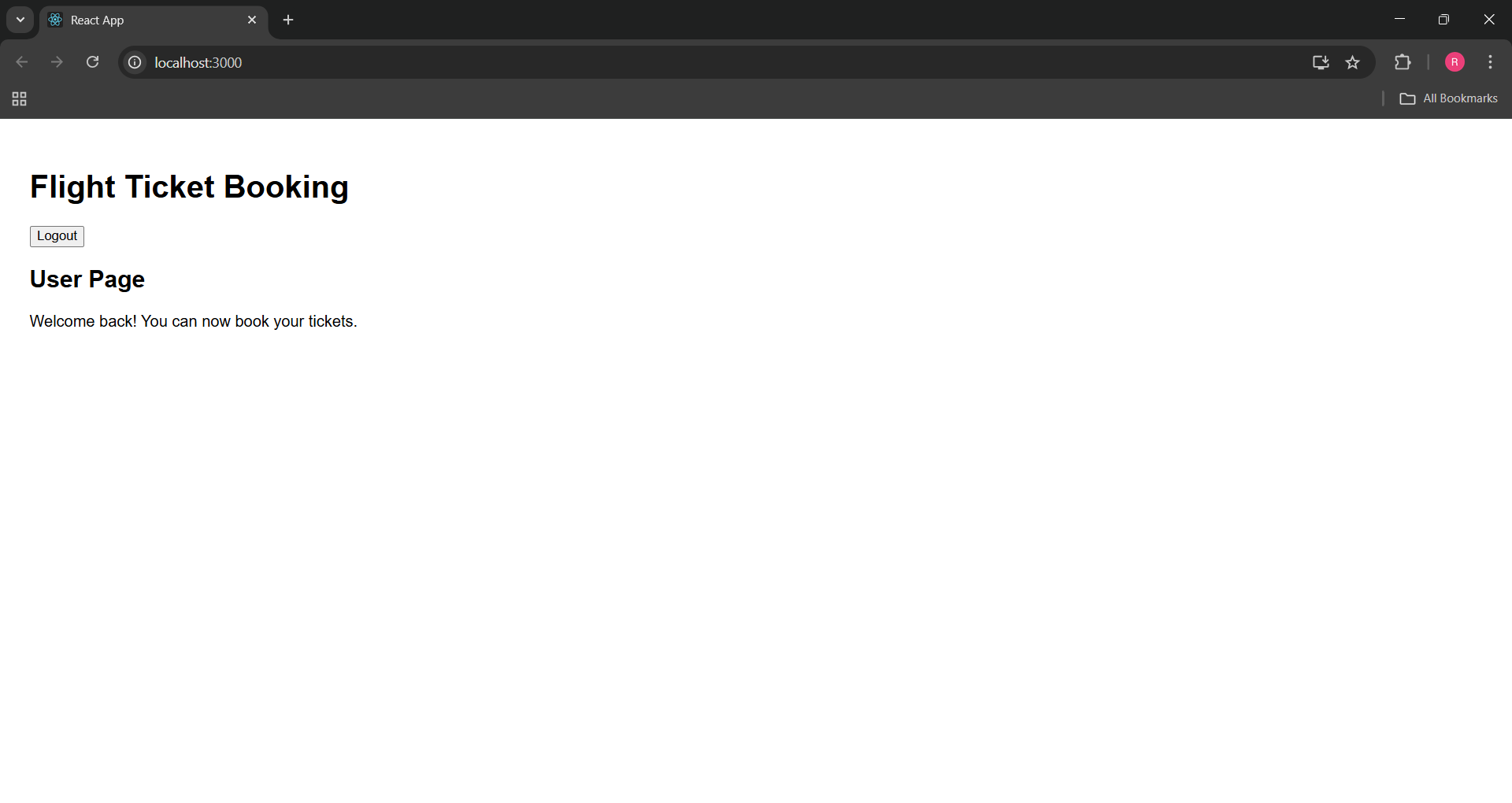
}

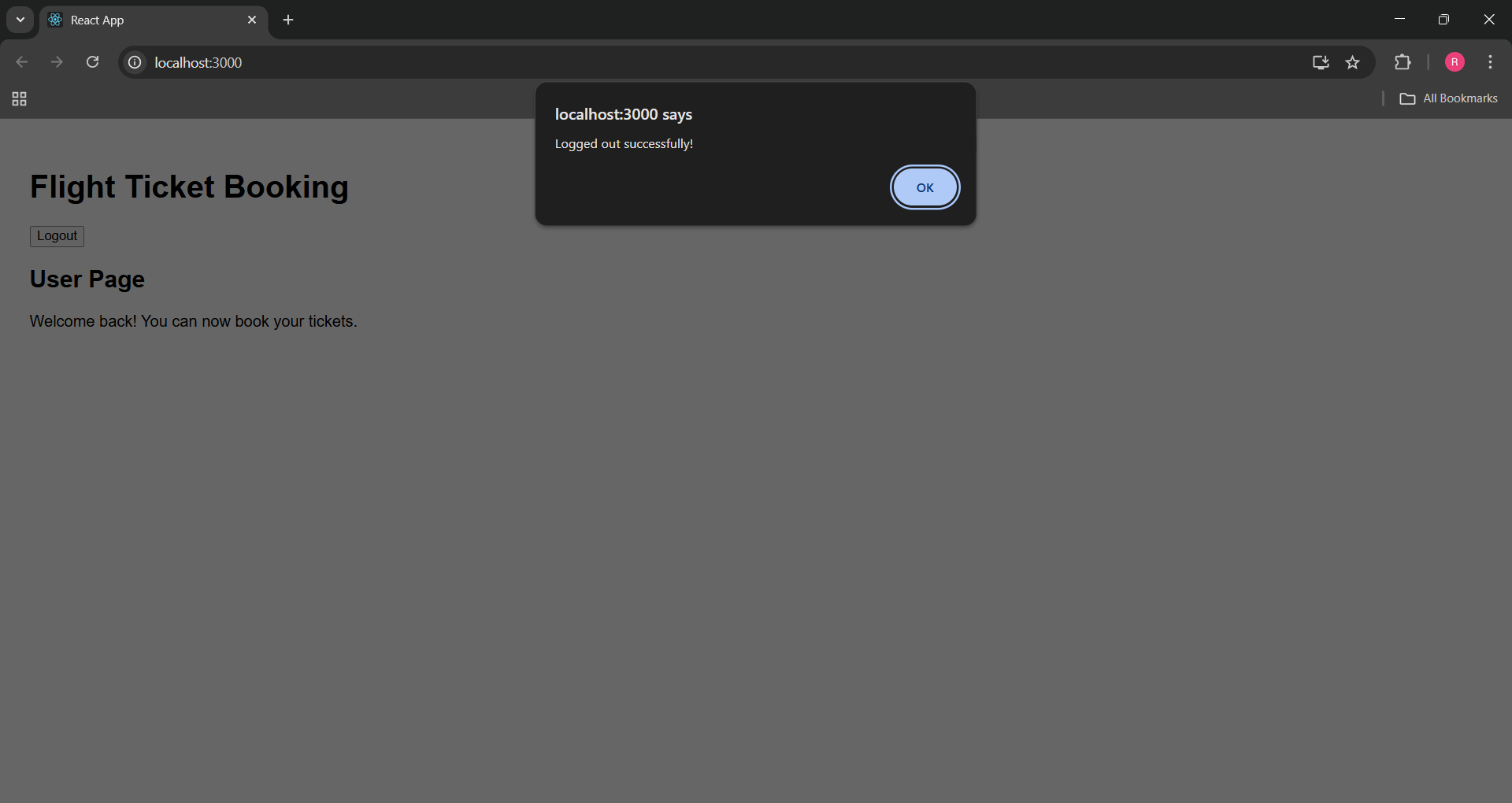
export default UserPage;

**Output:**

****







**5. ReactJS-HOL**

**Explain various ways of conditional rendering**

Conditional rendering in React allows different content to be displayed based on certain conditions. It can be implemented using standard JavaScript control structures like if-else statements, the ternary operator, and logical AND operators. These methods are used inside the component's render logic or directly within JSX to control which elements are shown depending on state or props.

**Explain how to render multiple components**

Multiple components can be rendered in React by including them within the return statement of a parent component. All the components should be properly imported and arranged within the JSX structure. This allows each component to render its own portion of the UI, contributing to a modular and organized application layout.

**Define list component**

A list component in React is used to display a collection of similar items. It allows developers to handle dynamic content efficiently by rendering multiple elements based on data stored in arrays. This helps create structured and reusable displays of data such as posts, users, or products.

**Explain about keys in React applications**

Keys are unique identifiers assigned to elements in a list in React. They help React determine which items have changed, been added, or removed. This optimizes the rendering process by allowing React to update only the necessary parts of the DOM, improving application performance and stability.

**Explain how to extract components with keys**

When working with dynamic lists in React, components can be extracted from within the list mapping logic to improve code readability and maintainability. During this process, keys must be passed as props to ensure each instance of the component remains uniquely identifiable and React can handle re-rendering correctly.

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

In React, the map() function is used to iterate over arrays and return a set of JSX elements. This approach is commonly used for dynamically rendering lists of data. It enables the creation of flexible UI components that respond to changes in the underlying data structure.

**Commands:**

npx create-react-app bloggerapp

cd bloggerapp

**Project Name: bloggerapp**

**App.js**

import React from "react";

import CourseDetails from "./CourseDetails";

import BookDetails from "./BookDetails";

import BlogDetails from "./BlogDetails";

import "./App.css";

function App() {

  const courseList = [

    { id: 1, cname: "Angular", date: "4/5/2021" },

    { id: 2, cname: "React", date: "6/3/2021" },

  ];

  const bookList = [

    { id: 1, bname: "Master React", price: 670 },

    { id: 2, bname: "Deep Dive into Angular 11", price: 800 },

    { id: 3, bname: "Mongo Essentials", price: 450 },

  ];

  const blogList = [

    {

      id: 1,

      title: "React Learning",

      author: "Stephen Biz",

      content: "Welcome to learning React!",

    },

    {

      id: 2,

      title: "Installation",

      author: "Schewzdenier",

      content: "You can install React from npm.",

    },

  ];

  return (

    <div className="container">

      <CourseDetails courses={courseList} />

      <BookDetails books={bookList} />

      <BlogDetails blogs={blogList} />

    </div>

  );

}

export default App;

**App.css**

.App {

  text-align: center;

}

.App-logo {

  height: 40vmin;

  pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

  .App-logo {

    animation: App-logo-spin infinite 20s linear;

  }

}

.App-header {

  background-color: #282c34;

  min-height: 100vh;

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  font-size: calc(10px + 2vmin);

  color: white;

}

.App-link {

  color: #61dafb;

}

@keyframes App-logo-spin {

  from {

    transform: rotate(0deg);

  }

  to {

    transform: rotate(360deg);

  }

}.container {

  display: flex;

  justify-content: space-evenly;

  padding: 20px;

  font-family: Arial, sans-serif;

}

.box {

  border-left: 3px solid green;

  padding: 0 20px;

  width: 30%;

}

**BookDetails.js**

import React from "react";

function BookDetails(props) {

  return (

    <div className="box">

      <h2>Book Details</h2>

      {props.books.map((book) => (

        <div key={book.id}>

          <h3>{book.bname}</h3>

          <h4>{book.price}</h4>

        </div>

      ))}

    </div>

  );

}

export default BookDetails;

**CourseDetails.js**

import React from "react";

function CourseDetails(props) {

  return (

    <div className="box">

      <h2>Course Details</h2>

      {props.courses.map((course) => (

        <div key={course.id}>

          <h3>{course.cname}</h3>

          <p>{course.date}</p>

        </div>

      ))}

    </div>

  );

}

export default CourseDetails;

**BlogDetails.js**

import React from "react";

function BlogDetails(props) {

  return (

    <div className="box">

      <h2>Blog Details</h2>

      {props.blogs.map((blog) => (

        <div key={blog.id}>

          <h3>{blog.title}</h3>

          <strong>{blog.author}</strong>

          <p>{blog.content}</p>

        </div>

      ))}

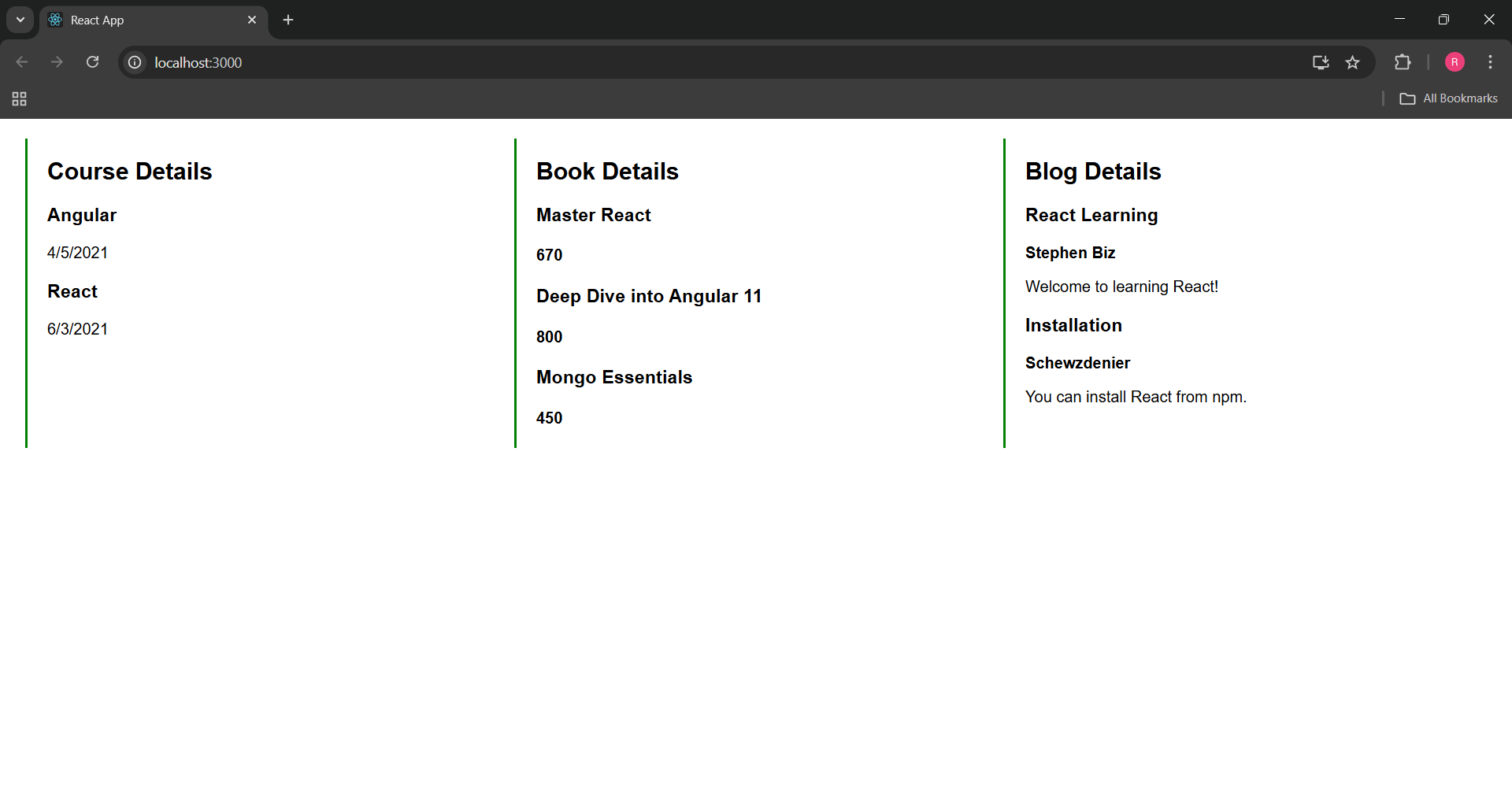
    </div>

  );

}

export default BlogDetails;

**Output:**

****