**Digital Nurture 4.0 Deep Skilling - Java FSE**  
**WEEK –7 Hands-on Exercises**  
**Module 10 - React**

**9. ReactJS-HOL**

**1. List the features of ES6**

* Block-scoped variables: let and const
* Arrow functions: Concise function syntax (() => {})
* Classes: Native class syntax for object-oriented code
* Template literals: String interpolation with backticks and ${}
* Destructuring assignment: Easy extraction from arrays/objects
* Default, rest, and spread operators:
  + Default function parameters
  + Rest (...args) to collect
  + Spread (...array) to expand
* Modules: import and export syntax for modular JS
* Promises: Simplified asynchronous programming
* Enhanced object literals: Shorter syntax/methods
* for...of loops: Iterating over collections
* Map and Set data structures
* Symbol type for unique object property keys

**2. Explain JavaScript let**

* let is used to declare variables that are **block-scoped** (only accessible within the block they are declared in, e.g., inside {}).
* Variables declared with let **can be updated**, but **cannot be redeclared** in the same scope.
* Helps prevent bugs from unintentional variable hoisting and scope issues.

**Example:**

let x = 10;

if (true) {

let x = 20;

// This x is different from the x above

}

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

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Redeclaration | Can be redeclared in same scope | Cannot redeclare in same scope |
| Hoisting | Hoisted and initialized as undefined | Hoisted but not initialized |
| Reassignment | Allowed | Allowed |

**Example:**

if(true) {

var a = 1; // available outside the block

let b = 2; // only available inside the block

}

console.log(a); // 1

console.log(b); // error: b is not defined

**4. Explain JavaScript const**

* const is used to declare block-scoped constants.
* Must be initialized at declaration.
* Cannot be reassigned. However, if the variable holds an object or array, the object’s properties or array’s elements can still be modified.

**Example:**

const PI = 3.14;

PI = 3.1416; // Error

const arr = [1,2,3];

arr.push(4); // Allowed

**5. Explain ES6 class fundamentals**

* ES6 introduced a native class syntax for creating objects and handling inheritance, making OOP in JS more intuitive.
* Classes include:
  + **Constructor:** Special method for creating and initializing objects.
  + **Methods:** Functions attached to the class.

**Example:**

class Person {

constructor(name) {

this.name = name;

}

greet() {

return `Hello, ${this.name}!`;

}

}

**6. Explain ES6 class inheritance**

* ES6 allows classes to **inherit** from other classes using the extends keyword.
* The super keyword calls the parent class’s constructor or methods.

**Example:**

class Person {

constructor(name) { this.name = name; }

}

class Student extends Person {

constructor(name, roll) {

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

this.roll = roll;

}

}

**7. Define ES6 arrow functions**

* A shorter syntax for writing functions.
* **Does not bind its own this** (inherits from the parent scope).
* Cannot be used as constructors.

**Example:**

// Traditional

function add(a, b) {

return a + b;

}

// Arrow

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

**8. Identify set(), map()**

**Set**: Collection of unique values (no duplicates).

Methods: .add(), .has(), .delete()

* + Example:

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

mySet.add(4); // Set {1, 2, 3, 4}

**Map**: Collection of key-value pairs, where keys can be any type.

* Methods: .set(key, value), .get(key), .has(key)
* Example:

const myMap = new Map();

myMap.set('a', 1);

myMap.get('a'); // 1

CODE:

import React, { useState } from "react";

function App() {

  const [showListOfPlayers, setShowListOfPlayers] = useState(true);

  // Sample players array as per the image

  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 },

  ];

  // Filter players with score less than 70

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

  return (

    <div>

      <button onClick={() => setShowListOfPlayers(!showListOfPlayers)}>

        Switch View

      </button>

      {showListOfPlayers && (

        <>

          <h1>List of Players</h1>

          <ul>

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

              <li key={idx}>

                {player.name} {player.score}

              </li>

            ))}

          </ul>

          <hr />

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

          <ul>

            {lowScorers.map((player, idx) => (

              <li key={idx}>

                {player.name} {player.score}

              </li>

            ))}

          </ul>

        </>

      )}

      {/\* Here you can show the alternate view (when flag is false) \*/}

      {/\* {!showListOfPlayers && (

        <AlternateOutput />

      )} \*/}

    </div>

  );

}

export default App;

**src/App.js:**

// src/App.js

import React from 'react';

import ListOfPlayers from './ListOfPlayers';

import IndianPlayers from './IndianPlayers';

function App() {

  const flag = true; // change this to false to test conditional rendering

  return (

    <div className="App">

      <h1>Welcome to Cricket App</h1>

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

    </div>

  );

}

export default App;

**src/App.js:**

import React, { useState } from "react";

function App() {

const [showListOfPlayers, setShowListOfPlayers] = useState(false);

const indianPlayers = [

"Sachin1", // 0 (First)

"Dhoni2", // 1 (Second)

"Virat3", // 2 (Third)

"Rohit4", // 3 (Fourth)

"Yuvaraj5", // 4 (Fifth)

"Raina6", // 5 (Sixth)

];

// Odd (index 0,2,4): 0, 2, 4

const oddPlayers = [

{ position: "First", name: indianPlayers[0] },

{ position: "Third", name: indianPlayers[2] },

{ position: "Fifth", name: indianPlayers[4] },

];

// Even (index 1,3,5): 1, 3, 5

const evenPlayers = [

{ position: "Second", name: indianPlayers[1] },

{ position: "Fourth", name: indianPlayers[3] },

{ position: "Sixth", name: indianPlayers[5] },

];

// Merged list from two arrays

const T20players = [

"Mr. First Player",

"Mr. Third Player",

"Mr. Fifth Player",

];

const RanjiPlayers = [

"Mr. Second Player",

"Mr. Fourth Player",

"Mr. Sixth Player",

];

const mergedPlayers = [...T20players, ...RanjiPlayers];

return (

<div>

<button onClick={() => setShowListOfPlayers(!showListOfPlayers)}>

Switch View

</button>

{!showListOfPlayers && (

<>

<h1>Odd Players</h1>

<ul>

{oddPlayers.map((player, idx) => (

<li key={idx}>

{player.position} : {player.name}

</li>

))}

</ul>

<hr />

<h1>Even Players</h1>

<ul>

{evenPlayers.map((player, idx) => (

<li key={idx}>

{player.position} : {player.name}

</li>

))}

</ul>

<hr />

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

<ul>

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

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

))}

</ul>

</>

)}

{/\* The flag==true view goes here \*/}

</div>

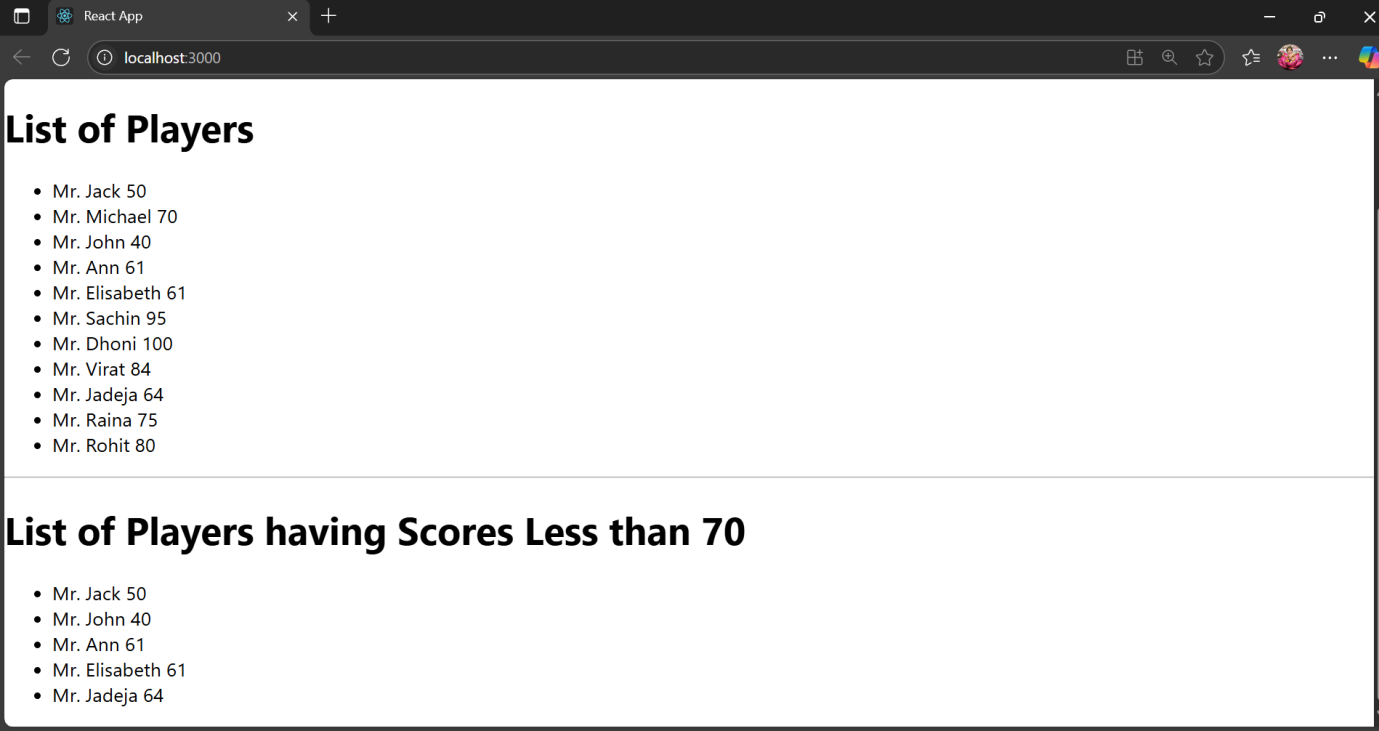
);

}

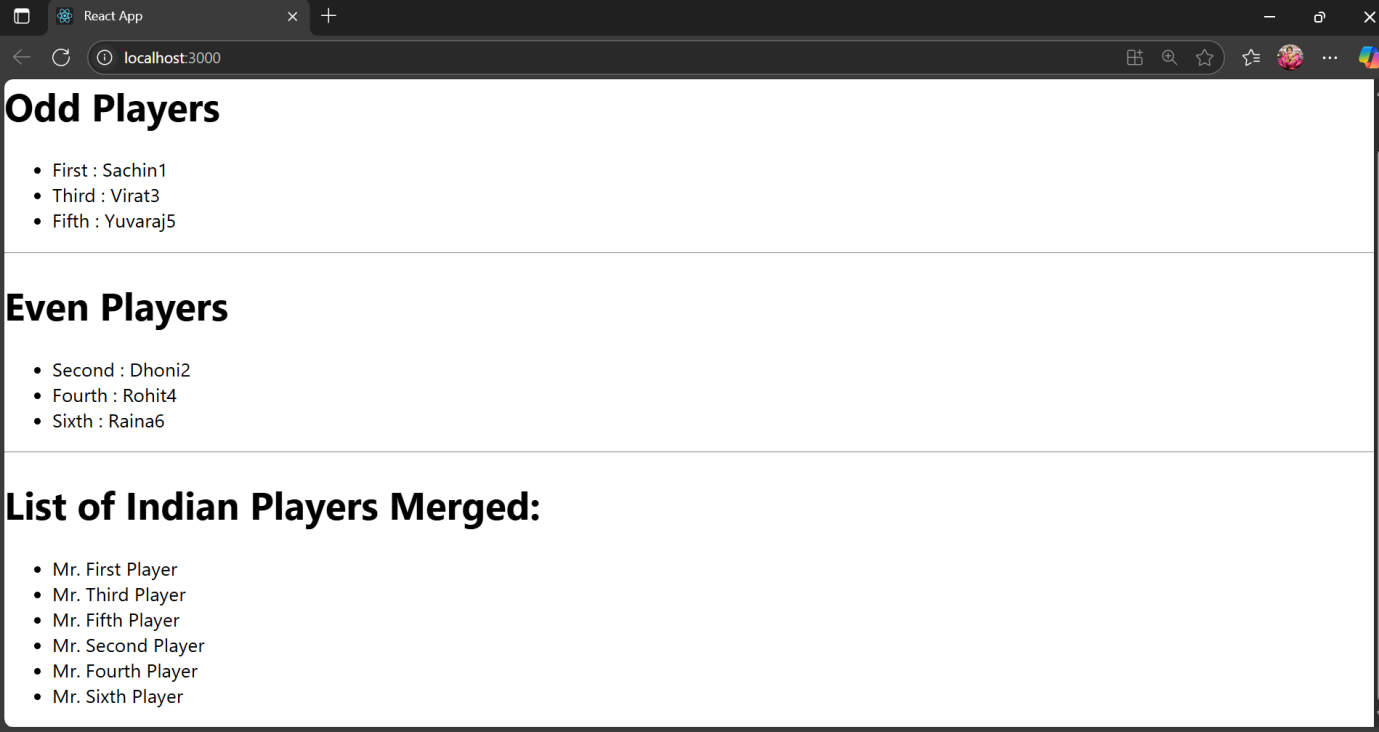
export default App;

**OUTPUT:**

When flag=true



When flag=false



1. **ReactJS-HOL**

**Objectives**

**1. Define JSX**

JSX (JavaScript XML) is a syntax extension for JavaScript used with React.  
It allows you to write HTML-like code inside JavaScript, which gets transformed into React elements.

**Example:**

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

JSX makes your UI code more readable and concise.

**2. Explain ECMA Script**

**ECMAScript (ES)** is the **standardized version of JavaScript**.

* **ES6 (ECMAScript 2015)** introduced major features like:
  + let and const
  + Arrow functions
  + Classes
  + Template literals
  + Destructuring
  + Modules (import/export)
  + Promises

React is built using ES6+ syntax for cleaner and modern code.

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

Before JSX, React elements were created using the React.createElement() function.

**Syntax:**

React.createElement(type, props, children)

**Example:**

const element = React.createElement('h1', { className: 'title' }, 'Hello React');

**This is equivalent to JSX:**

const element = <h1 className="title">Hello React</h1>;

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

In JSX, a **React node** can be created just like writing HTML inside JS:

**Example:**

const heading = <h1>Welcome to Office Space</h1>;

const image = <img src="office.jpg" alt="Office" />;

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

In React, you render JSX into the DOM using ReactDOM.render() (in older versions) or using the root.render() method in React 18+.

**Example (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 />);

JSX in <App /> gets rendered inside the div with id="root" in public/index.html.

**6. Using JavaScript Expressions in JSX**

You can embed any **valid JavaScript expression** inside JSX using {}.

**Example:**

const user = 'Priya';

const rent = 55000;

return (

<div>

<h1>Welcome, {user}</h1>

<p>Rent: Rs. {rent}</p>

<p>Status: {rent > 60000 ? "Expensive" : "Affordable"}</p>

</div>

);

**7. Using Inline CSS in JSX**

Use the style prop with a JavaScript object (camelCase style names):

**Example:**

<p style={{ color: 'red', fontWeight: 'bold' }}>Rent: Rs. 50000</p>

**CODE:**

**Src/App.js:**

import React from "react";

function App() {

  return (

    <div style={{ marginLeft: "80px", marginTop: "60px" }}>

      <h1>

        Office Space , at Affordable Range

      </h1>

      <img

        src="https://images.unsplash.com/photo-1573164574572-cb89e39749b4?auto=format&fit=crop&w=600&q=60"

        alt="office"

        style={{ width: "250px", height: "200px", objectFit: "cover", display: "block", marginTop: "25px" }}

      />

      <div style={{ marginTop: "30px" }}>

        <h2>

          Name: <span style={{ fontWeight: "bold" }}>DBS</span>

        </h2>

        <p style={{ color: "red", fontWeight: "bold", fontSize: "20px", margin: '0' }}>

          Rent: Rs. 50000

        </p>

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

          Address: <span style={{ color: "black" }}>Chennai</span>

        </p>

      </div>

    </div>

  );

}

export default App;

**src/App.css:**

/\* src/App.css \*/

.App {

  text-align: center;

  font-family: Arial, sans-serif;

  padding: 20px;

}

h1 {

  color: navy;

}

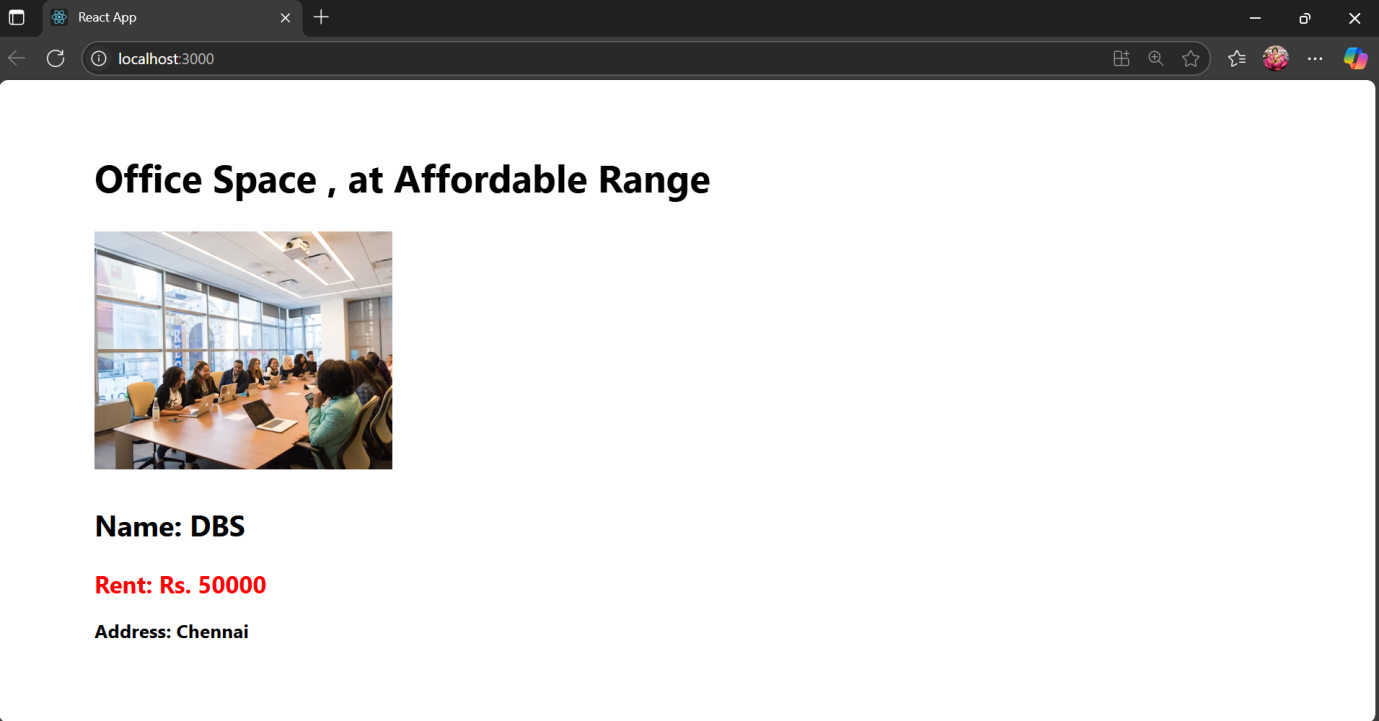
img {

  border-radius: 8px;

  margin-bottom: 20px;

}

**OUTPUT:**



**11. ReactJS-HOL**

**Objectives**

**1. Explain React Events**

React events are how you handle user interactions in a React application — like clicks, typing, hovering, submitting forms, etc.

React uses its own cross-browser wrapper for native DOM events — called the Synthetic Event System**.**

**Example:**

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

**2. Explain Event Handlers**

**Event handlers** are functions that respond when a specific event occurs.

They can be:

* Inline
* Arrow functions
* Defined as separate named functions

**Example**: Event handler in React

function handleClick() {

alert("Button Clicked!");

}

function App() {

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

}

**3. Define Synthetic Event**

A **Synthetic Event** is a **wrapper around the browser’s native event** system provided by React.

* It normalizes events across all browsers.
* You don’t need to worry about cross-browser differences (like IE vs Chrome).

**Example:**

function handleChange(event) {

console.log(event.target.value); // event is a SyntheticEvent

}

**4. React Event Naming Conventions**

React follows **camelCase** for event names, unlike traditional HTML.

| **HTML** | **React** |
| --- | --- |
| onclick | onClick |
| onchange | onChange |
| onmouseover | onMouseOver |
| onsubmit | onSubmit |

**CODE:**

Src/App.js:

import React, { useState } from "react";

function App() {

  // Counter state

  const [counter, setCounter] = useState(5);

  // Currency converter states

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

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

  // Increment button - triggers two functions (alert + increase counter)

  const handleIncrement = () => {

    alert("Hello! Member1");

    setCounter(counter + 1);

  };

  // Decrement button

  const handleDecrement = () => {

    setCounter(counter - 1);

  };

  // Say welcome button with argument

  const handleWelcome = () => {

    alert("welcome");

  };

  // Universal click event

  const handleClick = () => {

    alert("I was clicked");

  };

  // Currency converter submit

  const handleSubmit = (e) => {

    e.preventDefault();

    // Example: if converting to Euro and amount is entered

    if (currency.trim().toLowerCase() === "euro" && amount) {

      // Let's say 1 unit = 80 INR (so 80 INR = 1 Euro, as per your screenshot logic: 80 x 80 = 6400)

      const amt = parseFloat(amount);

      if (!isNaN(amt)) {

        alert(`Converting to Euro Amount is ${amt \* 80}`);

      }

    }

  };

  return (

    <div style={{ margin: "30px" }}>

      {/\* Counter display \*/}

      <div style={{ fontSize: "20px", marginBottom: "10px" }}>{counter}</div>

      {/\* Buttons for Increment/Decrement/Welcome/Click \*/}

      <button onClick={handleIncrement}>Increment</button>

      <br />

      <button onClick={handleDecrement}>Decrement</button>

      <br />

      <button onClick={handleWelcome}>Say welcome</button>

      <br />

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

      {/\* Heading \*/}

      <h1 style={{

        color:"green",

        marginTop: "40px",

        fontFamily: "Arial",

        fontWeight: "bold"

      }}>Currency Convertor!!!</h1>

      {/\* Currency Converter Form \*/}

      <form onSubmit={handleSubmit}>

        <table>

          <tbody>

            <tr>

              <td style={{ fontSize: "18px" }}>Amount:</td>

              <td>

                <input

                  type="text"

                  value={amount}

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

                  style={{ margin: "10px", height: "22px", fontSize: "16px" }}

                />

              </td>

            </tr>

            <tr>

              <td style={{ fontSize: "18px" }}>Currency:</td>

              <td>

                <input

                  type="text"

                  value={currency}

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

                  style={{ margin: "10px", height: "30px", fontSize: "16px" }}

                />

              </td>

            </tr>

            <tr>

              <td colSpan={2}>

                <button

                  type="submit"

                  style={{

                    marginLeft: "70px",

                    width: "60px",

                    height: "28px"

                  }}

                >Submit</button>

              </td>

            </tr>

          </tbody>

        </table>

      </form>

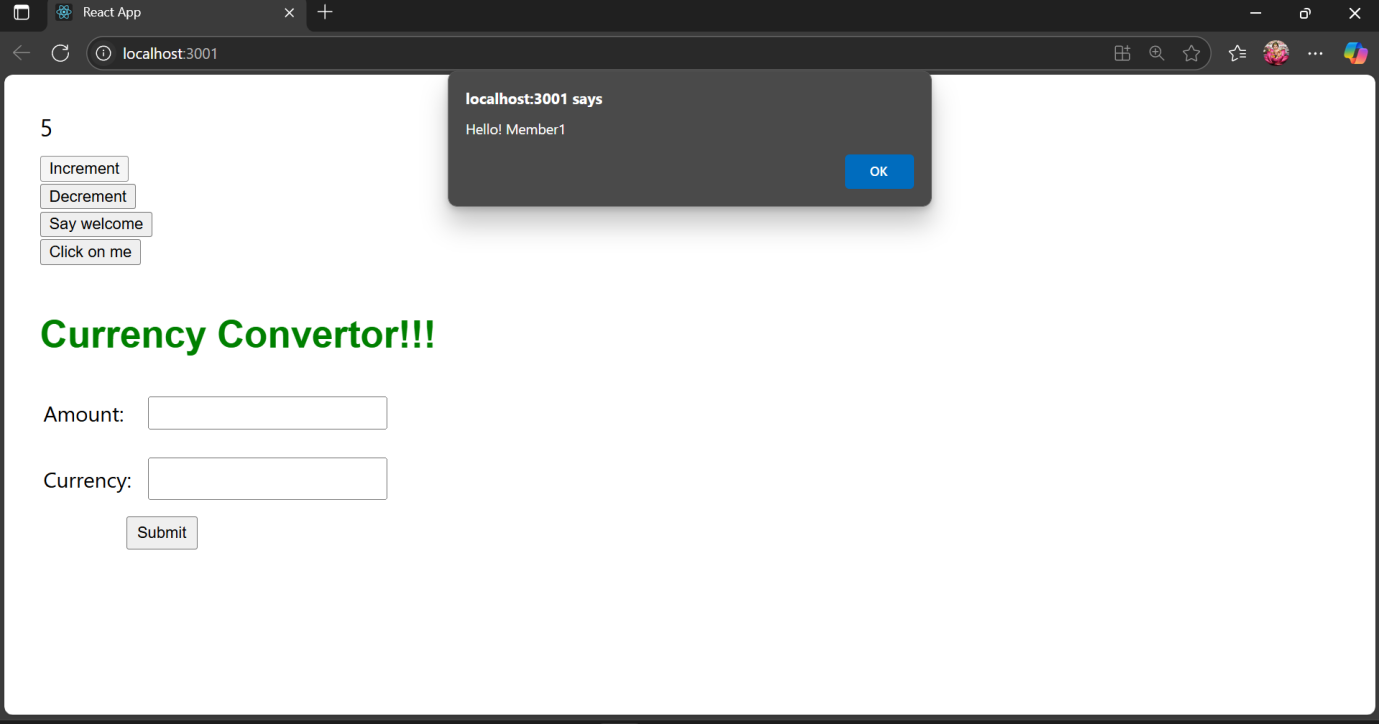
    </div>

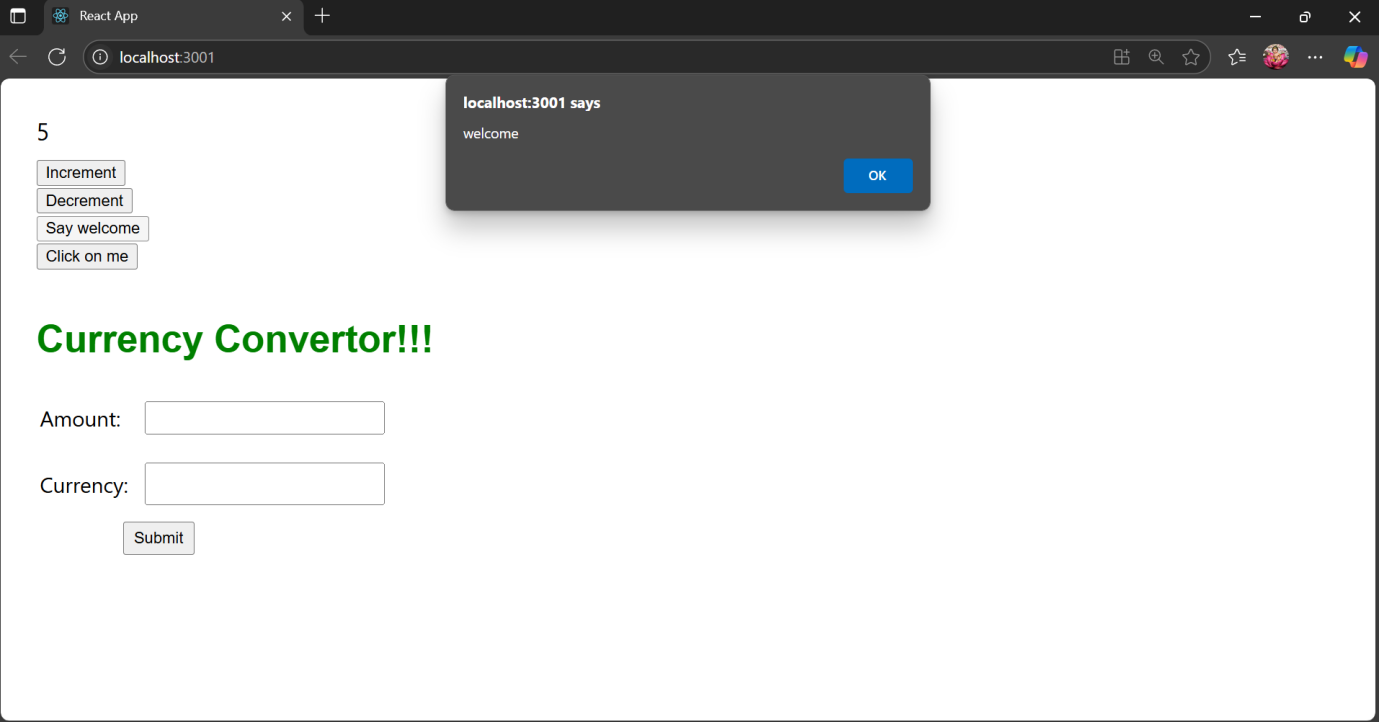
  );

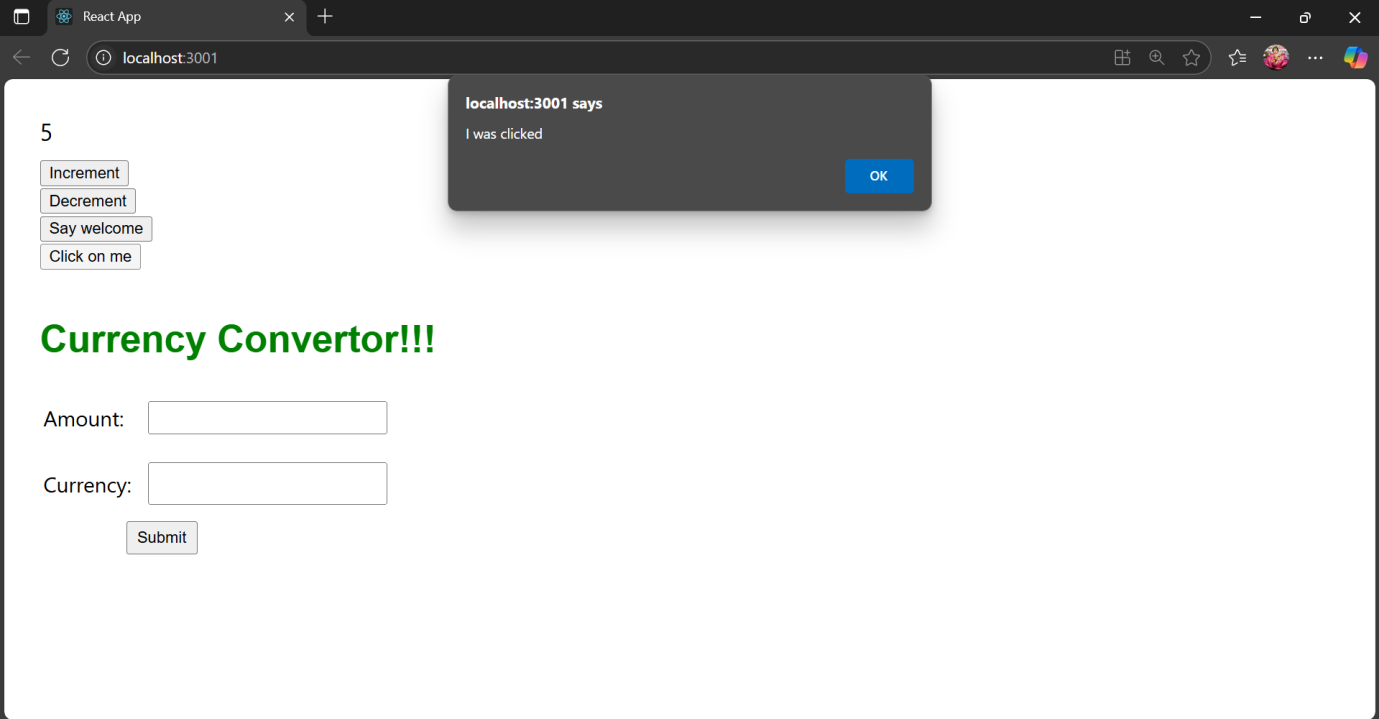
}

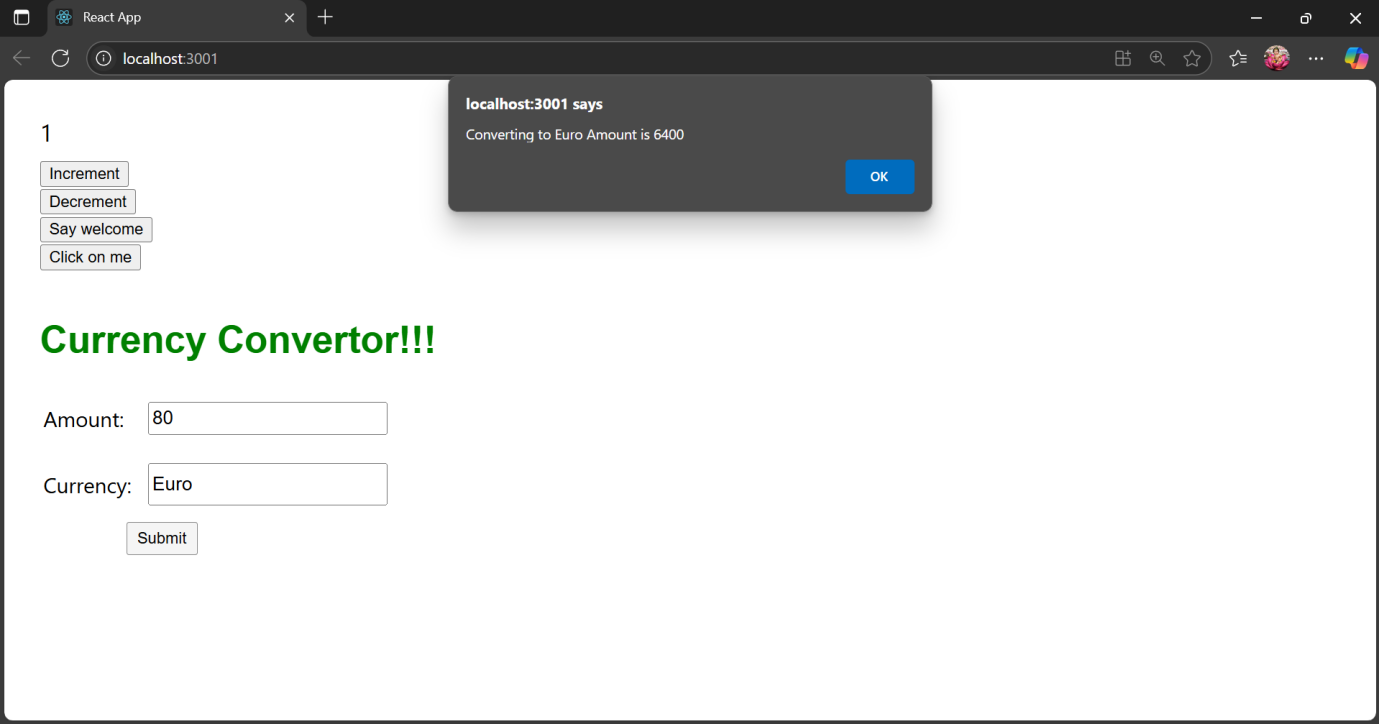
export default App;

**OUTPUT:**

****

****

****

****

**12.ReactJS-HOL**

**Objectives**

**1. Explain Conditional Rendering in React**

Conditional rendering means showing or hiding certain parts of the UI based on conditions (like if, else, or true/false flags).

React uses JavaScript expressions (like ternary or logical operators) inside JSX to control what gets rendered.

**Example 1:** Using if-else

function Greeting(props) {

if (props.isLoggedIn) {

return <h1>Welcome Back!</h1>;

}

return <h1>Please Log In</h1>;

}

**Example 2:** Using ternary operator

<h1>{isLoggedIn ? 'Welcome Back!' : 'Please Log In'}</h1>

**Example 3:** Using && (logical AND)

{isLoggedIn && <p>You have new notifications</p>}

**2. Define Element Variables**

**Element variables** let you store JSX elements in variables so you can conditionally assign and render them later.

**Example:**

function UserStatus(props) {

let message;

if (props.isOnline) {

message = <p>User is Online</p>;

} else {

message = <p>User is Offline</p>;

}

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

}

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

**Option 1: Return null from the component**

If a component returns null, **nothing will be rendered**.

function WarningBanner(props) {

if (!props.showWarning) {

return null; // Nothing gets rendered

}

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

}

**Option 2: Use conditional rendering in parent**

{showBanner && <WarningBanner />}

The WarningBanner component will not even be created unless showBanner is true.

**Option 3: Use CSS to hide content (not ideal)**

<div style={{ display: show ? 'block' : 'none' }}>Content</div>

**CODE:**

**src/App.js:**

import React, { useState } from "react";

function App() {

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

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

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

  return (

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

      {isLoggedIn ? (

        <>

          <h1>Welcome back</h1>

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

        </>

      ) : (

        <>

          <h1>Please sign up.</h1>

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

        </>

      )}

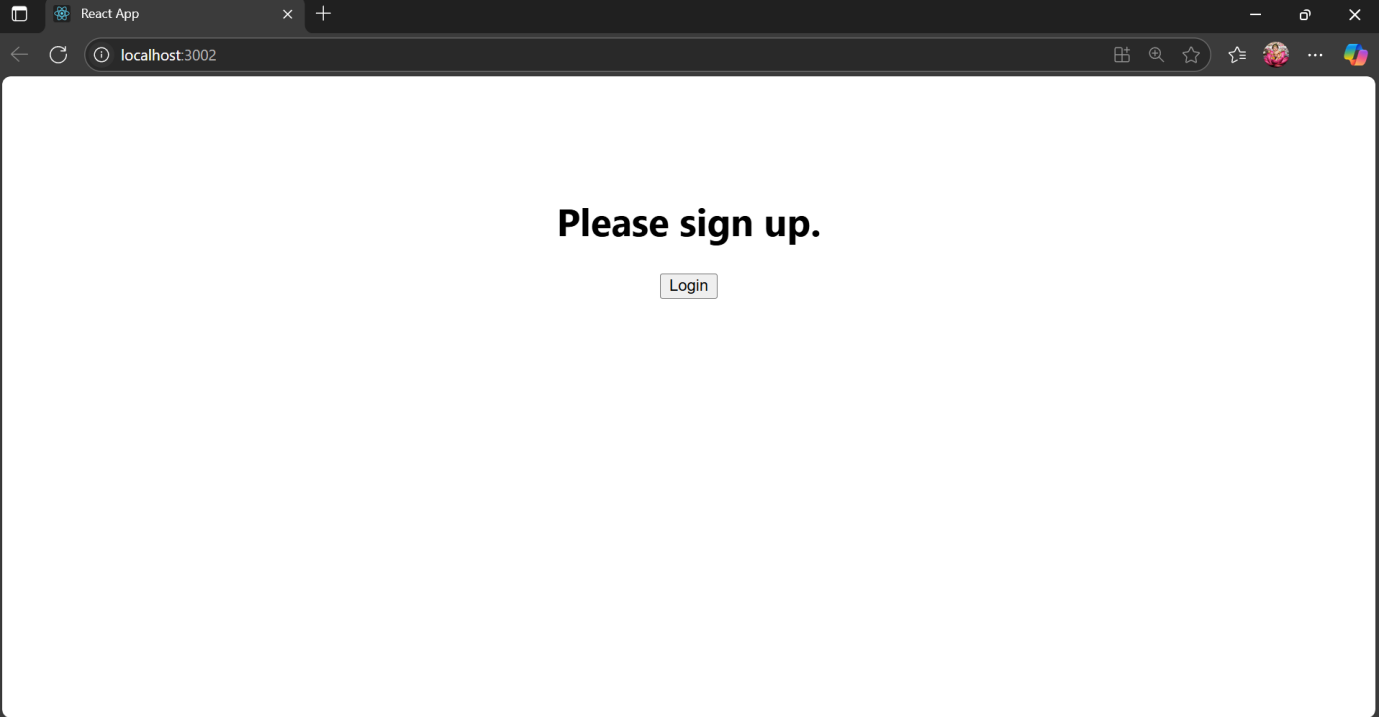
    </div>

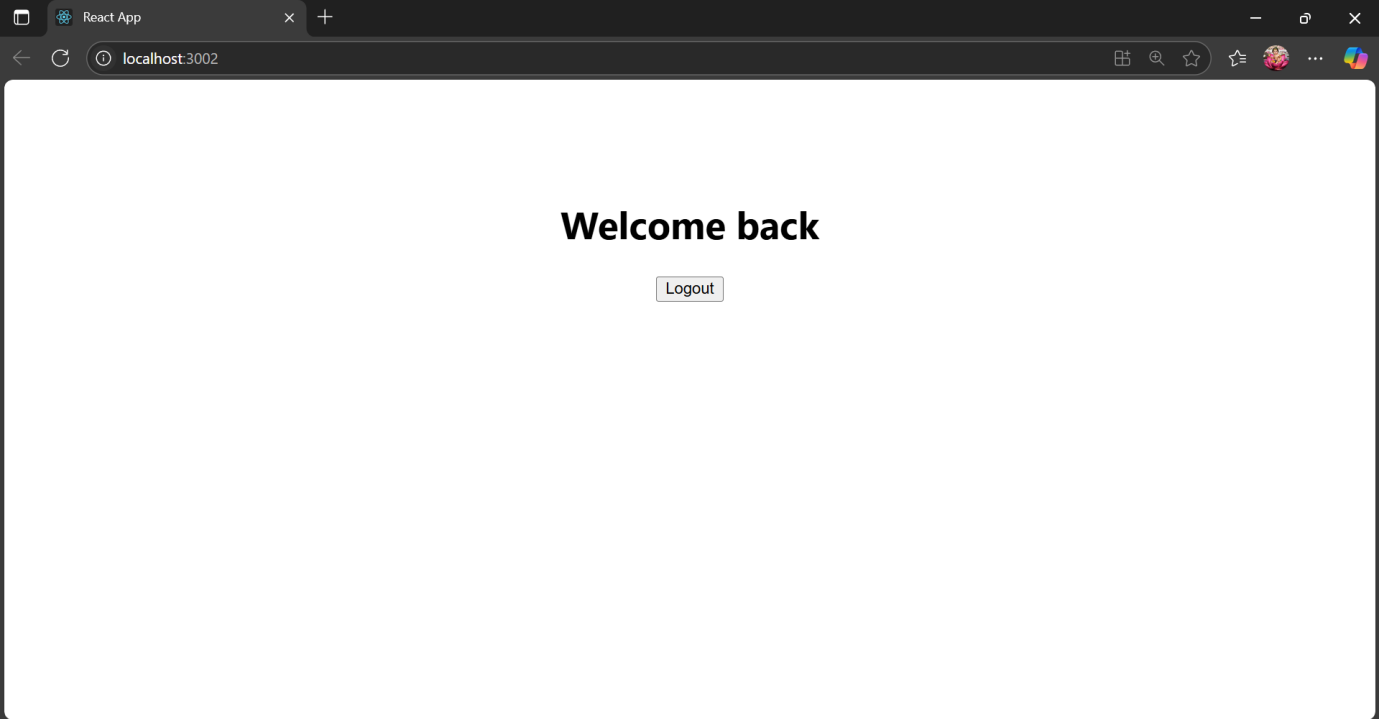
  );

}

export default App;

**OUTPUT:**

****

****

**13. ReactJS-HOL**

**Objectivies**

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

React allows different ways to render UI conditionally:

**🔹 a. Using if-else statements**

if (isLoggedIn) {

return <Dashboard />;

} else {

return <Login />;

}

🔹 b. **Ternary operator**

{isLoggedIn ? <Dashboard /> : <Login />}

🔹 c. **Logical && operator**

{hasNotification && <Notification />}

🔹 d. **Element variables**

let content;

if (isAdmin) {

content = <AdminPanel />;

} else {

content = <UserPanel />;

}

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

**2. How to Render Multiple Components**

You can render multiple components inside a parent <div> or <React.Fragment>:

**Example:**

function App() {

return (

<div>

<Header />

<Sidebar />

<Content />

<Footer />

</div>

);

}

**3. Define List Component**

A list component renders a dynamic list of items, usually using map().

**Example:**

function OfficeList() {

const offices = ['Chennai', 'Bangalore', 'Delhi'];

return (

<ul>

{offices.map((city, index) => <li key={index}>{city}</li>)}

</ul>

);

}

**4. Explain Keys in React Applications**

**Keys** help React **identify which items have changed, added, or removed** in a list.  
They should be **unique and stable**.

**🔹 Why are keys important?**

* Helps React optimize re-rendering
* Prevents unwanted reordering or duplication

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

* When rendering a list, you often want to extract a separate component. Pass the key **to the outer component**, not inside it.

function Office({ name }) {

return <li>{name}</li>;

}

function OfficeList() {

const offices = ['Chennai', 'Bangalore', 'Delhi'];

return (

<ul>

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

<Office key={index} name={city} />

))}

</ul>

);

}

**6. Explain React map() / map() Function**

The map() function in React is used to **render a list of elements** dynamically from an array.

**Syntax:**

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

return <Element key={index} />;

});

**Example:**

const names = ['Anu', 'Ravi', 'Kiran'];

const nameList = names.map((name, i) => <li key={i}>{name}</li>);

**CODE:**

**src/App.js:**

import React from "react";

function App() {

  return (

    <div style={{

      minHeight: "100vh",

      display: "flex",

      justifyContent: "center",

      alignItems: "flex-start",

      gap: 40,

      fontFamily: "Arial, sans-serif"

    }}>

      {/\* Column 1: Course Details \*/}

      <div style={{minWidth: 250, padding: "10px"}}>

        <h1 style={{fontSize: "2rem"}}>Course Details</h1>

        <div style={{margin: "30px 0 10px 0"}}>

          <span style={{fontWeight: "bold", fontSize: "1.4rem"}}>Angular</span><br/>

          <span style={{fontSize: "1rem"}}>4/5/2021</span>

        </div>

        <div>

          <span style={{fontWeight: "bold", fontSize: "1.4rem"}}>React</span><br/>

          <span style={{fontSize: "1rem"}}>6/3/20201</span>

        </div>

      </div>

      {/\* Vertical line \*/}

      <div style={{borderLeft:"4px solid green", height:"330px"}} />

      {/\* Column 2: Book Details \*/}

      <div style={{minWidth: 250, padding: "10px"}}>

        <h1 style={{fontSize: "2rem"}}>Book Details</h1>

        <div style={{margin: "20px 0 0 0", fontWeight: "bold"}}>Master React</div>

        <div>670</div>

        <div style={{margin: "20px 0 0 0", fontWeight: "bold"}}>Deep Dive into Angular 11</div>

        <div>800</div>

        <div style={{margin: "20px 0 0 0", fontWeight: "bold"}}>Mongo Essentials</div>

        <div>450</div>

      </div>

      {/\* Vertical line \*/}

      <div style={{borderLeft:"4px solid green", height:"330px"}} />

      {/\* Column 3: Blog Details \*/}

      <div style={{minWidth: 250, padding: "10px"}}>

        <h1 style={{fontSize: "2rem"}}>Blog Details</h1>

        <div style={{margin:"24px 0 0 0"}}>

          <span style={{fontWeight:"bold", fontSize:"1.4rem"}}>React Learning</span>

          <br />

          <span style={{fontWeight:"bold", fontSize:"1rem"}}>Stephen Biz</span>

          <br />

          <span>Welcome to learning React!</span>

        </div>

        <div style={{margin:"24px 0 0 0"}}>

          <span style={{fontWeight:"bold", fontSize:"1.3rem"}}>Installation</span>

          <br />

          <span style={{fontWeight:"bold", fontSize:"1rem"}}>Schwezdenier</span>

          <br />

          <span>You can install React from npm.</span>

        </div>

      </div>

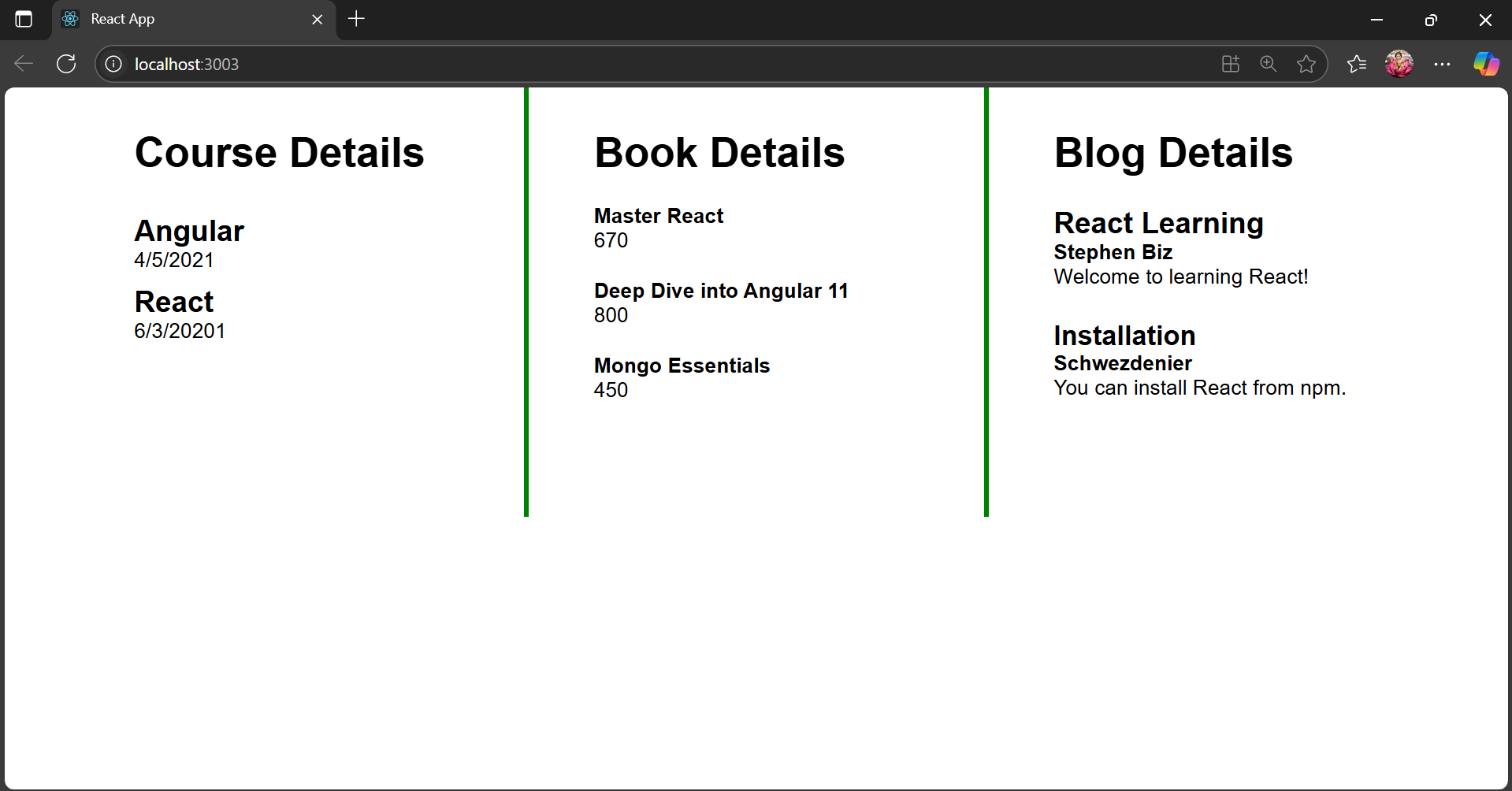
    </div>

  );

}

export default App;

**OUTPUT:**

****