**ReactJs-HOL-9**

**1.List the features of ES6**

* Block-scoped declarations using let and const.
* Arrow functions for shorter function syntax.
* Template literals for easier string concatenation and embedding expressions.
* Default parameters in functions.
* Rest and spread operators (...) for handling multiple arguments or array operations.
* Classes and inheritance for object-oriented programming.
* Promises for better asynchronous handling.
* Enhanced object literals for cleaner syntax.

**2. Explain JavaScript let**

* Introduced in ES6 for block-scoped variable declarations.
* Unlike var, let variables are not hoisted to the top of their scope.
* Cannot be redeclared in the same block.
* Useful for maintaining variables within loops or conditional blocks.

let x = 10;

if (true) {

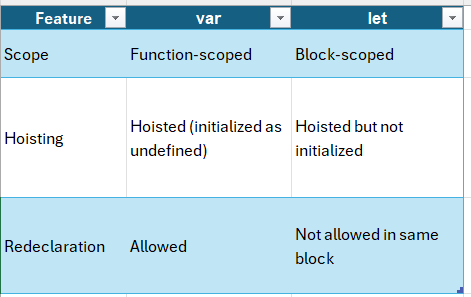
let x = 20;

console.log(x);

}

console.log(x);

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



**4. Explain JavaScript const**

* Declares variables that **cannot be reassigned**.
* Block-scoped like let.
* Must be **initialized** during declaration.
* For objects and arrays, the reference cannot change, but values inside can be modified.

const PI = 3.14;

**5. Explain ES6 class fundamentals**

* Provides a cleaner, more concise syntax for creating objects and handling inheritance.
* Uses class keyword.
* Supports constructor methods and class methods.

Example:

class Person {

constructor(name) {

this.name = name;

}

greet() {

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

}

}

**6. Explain ES6 class inheritance**

* Achieved using extends keyword.
* super() is used to call the parent class constructor**.**

**Example:**

class Employee extends Person {

constructor(name, role) {

super(name);

this.role = role;

}

}

**7. Define ES6 arrow functions**

* Shorter syntax for writing functions.
* Does not bind its own this.
* Useful in callbacks and functional programming.

Example:

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

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

**Set**:

* Stores unique values of any type.
* Methods: add(), delete(), has(), clear().

Example:

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

s.add(4);

console.log(s);

**Map**:

* Stores key-value pairs where keys can be any type.
* Methods: set(), get(), delete(), has(), clear().

Example:

let m = new Map();

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

console.log(m.get('name'));

**CricketApp:**

**src/components/ListofPlayers.js:**

import React from 'react';

export default function ListofPlayers({ players }) {

return (

<div>

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

<div key={index}>

<li>

Mr. {item.name} <span>{item.score}</span>

</li>

</div>

))}

</div>

);

}

export function Scorebelow70({ players }) {

const players70 = [];

players.map(item => {

if (item.score <= 70) {

players70.push(item);

}

return null;

});

return (

<div>

{players70.map((item, index) => (

<li key={index}>{item.name} - {item.score}</li>

))}

</div>

);

}

**src/components/IndianPlayers.js**

import React from 'react';

export function OddPlayers([first, , third, , fifth]) {

return (

<div>

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

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

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

</div>

);

}

export function EvenPlayers([, second, , fourth, , sixth]) {

return (

<div>

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

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

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

</div>

);

}

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

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

export const IndianPlayers = [...T20Players, ...RanjiTrophyPlayers];

export function ListofIndianPlayers({ IndianPlayers }) {

return (

<div>

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

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

))}

</div>

);

}

**App.js:**

import React from 'react';

import ListofPlayers, { Scorebelow70 } from './components/ListofPlayers';

import { OddPlayers, EvenPlayers, IndianPlayers, ListofIndianPlayers } from './components/IndianPlayers';

function App() {

const flag = true;

const players = [

{ name: 'Player1', score: 90 },

{ name: 'Player2', score: 65 },

{ name: 'Player3', score: 75 },

{ name: 'Player4', score: 55 },

{ name: 'Player5', score: 100 },

{ name: 'Player6', score: 80 },

{ name: 'Player7', score: 60 },

{ name: 'Player8', score: 95 },

{ name: 'Player9', score: 40 },

{ name: 'Player10', score: 88 },

{ name: 'Player11', score: 66 }

];

const IndianTeam = ['Player1', 'Player2', 'Player3', 'Player4', 'Player5', 'Player6'];

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>

);

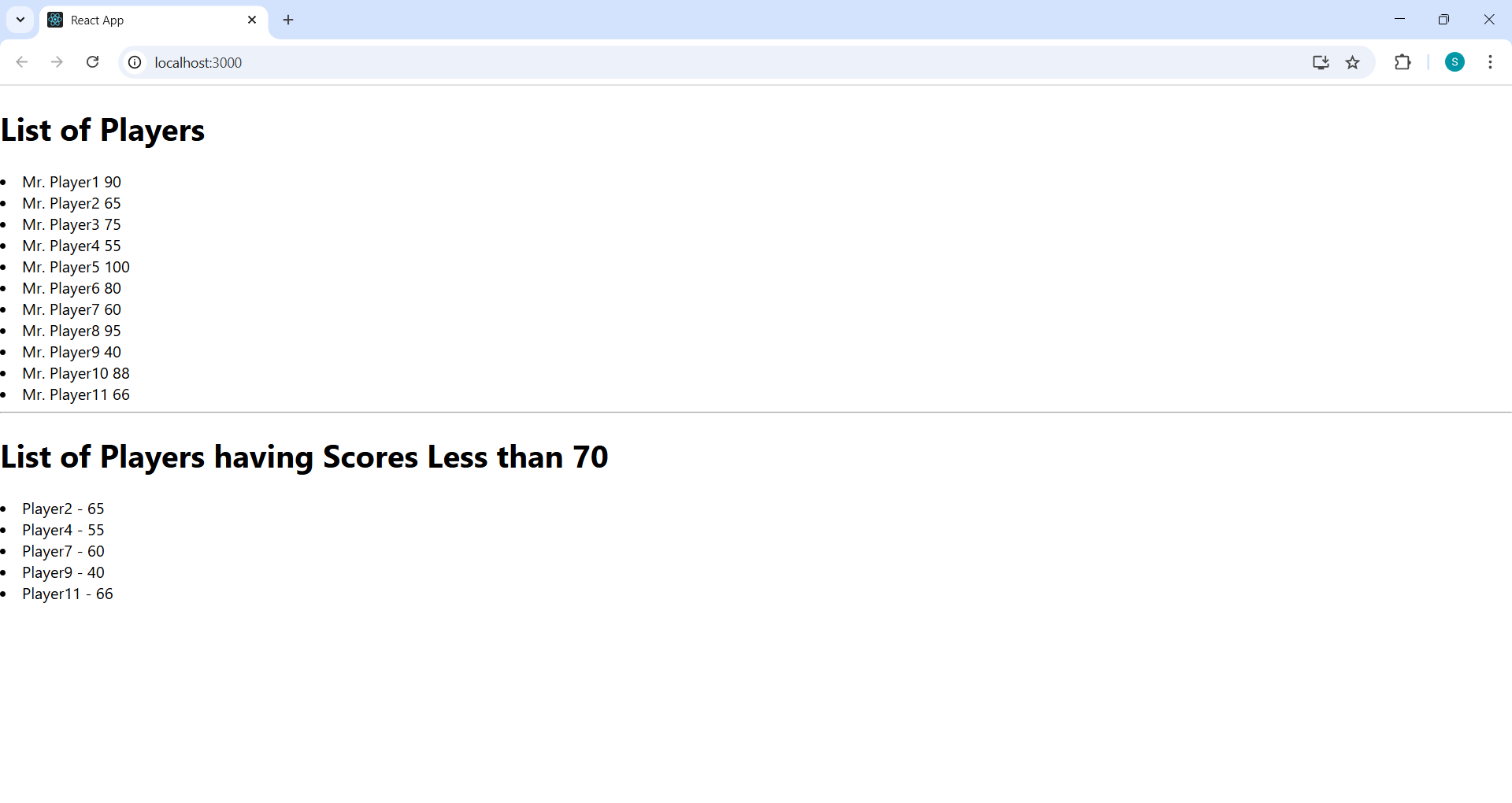
}

}

export default App;

**OUTPUT:**

**flag==true**

****

**flag==false**

**A screenshot of a computer

AI-generated content may be incorrect.**

**ReactJs-HOL-10**

**1.Define JSX**

* **JSX (JavaScript XML)** is a syntax extension for JavaScript used in React.
* It allows developers to write HTML-like code within JavaScript, which is then transformed into React elements.
* JSX improves readability and makes UI code easier to write and understand.

Example:

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

2. **Explain about ECMA Script**

* **ECMAScript (ES)** is the standard specification for scripting languages like JavaScript.
* It defines rules, syntax, and features for the language.
* ES6 (ECMAScript 2015) introduced modern JavaScript features like let, const, arrow functions, classes, promises, modules, etc.

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

* React.createElement() is a core React function that creates a virtual DOM element.
* It takes three arguments:
  1. The **type** of element (e.g., 'div', 'h1').
  2. The **props** (attributes or data).
  3. The **children** (nested elements or text).

Example:

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

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

* Instead of React.createElement(), JSX is commonly used for creating nodes.
* JSX is compiled into React.createElement() calls by Babel.

Example:

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

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

* Use ReactDOM.createRoot() and root.render() to render JSX into an HTML element.

Example:

import React from 'react';

import ReactDOM from 'react-dom/client';

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

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

root.render(element);

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

* Curly braces {} allow you to embed any JavaScript expression inside JSX.

Example:

const name = 'Alice';

const element = <h1>Hello, {name}!</h1>;

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

* Inline CSS in JSX is written as an object using **camelCase** for property names.
* The style attribute takes a JavaScript object.

Example:

const element = <h1 style={{ color: 'blue', fontSize: '24px' }}>Styled Text</h1>;

**Office Space rental App**

**App.js:**

import React from 'react';

import './App.css';

function App() {

const heading = "Office Space";

const officeList = [

{ Name: "DBS", Rent: 50000, Address: "Chennai", Image: "https://psnltd.com/wp-content/uploads/2022/12/iStock-1352854006.jpg" },

{ Name: "Regus", Rent: 65000, Address: "Bangalore", Image: "https://www.sohopodomorocity.com/wp-content/uploads/2016/09/4.jpg" },

{ Name: "CoWrks", Rent: 45000, Address: "Hyderabad", Image: "https://mir-s3-cdn-cf.behance.net/project\_modules/max\_1200/16b92355737099.599170d4c13d1.jpg" }

];

return (

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

<h1>{heading}, at Affordable Range</h1>

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

let rentColor = item.Rent <= 60000 ? 'red' : 'green';

return (

<div key={index} style={{ margin: '20px', border: '1px solid #ccc', padding: '10px', display: 'inline-block' }}>

<img src={item.Image} width="250" height="150" alt="Office Space" />

<h2>Name: {item.Name}</h2>

<h3 style={{ color: rentColor }}>Rent: Rs. {item.Rent}</h3>

<h4>Address: {item.Address}</h4>

</div>

);

})}

</div>

);

}

export default App;

**App.css:**

body {

  font-family: Arial, sans-serif;

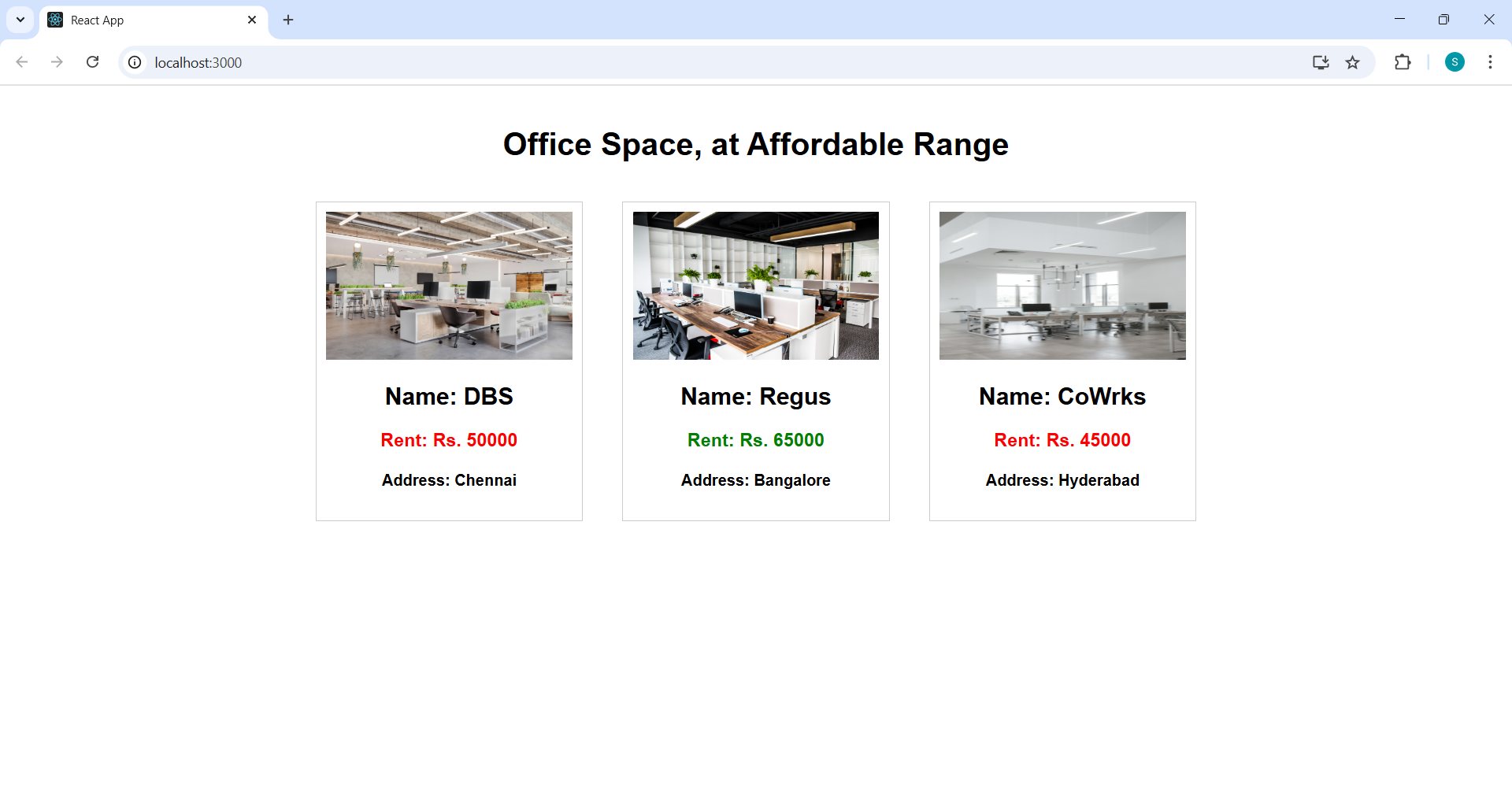
}

h1 {

  margin-bottom: 20px;

}

**OUTPUT:**



**ReactJs-HOL-11**

**1.** **Explain React events**

In React, events are actions or occurrences that happen in the browser, such as a user clicking a button, typing in an input field, or hovering over an element. React provides a way to handle these events using a syntax similar to HTML, but with some differences. These events are wrapped inside React’s SyntheticEvent system, ensuring cross-browser compatibility.

**2. Explain about event handlers**

Event handlers are functions in React that respond to specific events. They define what should happen when a particular event occurs on a React element. For example, when a button is clicked, an event handler function can be triggered to perform an action like updating the state or submitting a form.

**Example:**

function handleClick() {

alert("Button was clicked!");

}

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

Here, handleClick is the event handler that runs when the button is clicked.

**3. Define Synthetic event**

A SyntheticEvent is a cross-browser wrapper around the native browser event in React. It is part of React's event delegation system and provides a consistent API for handling events across different browsers.

React creates this wrapper to normalize the event object and improve performance. The synthetic event behaves identically to native events, but it works the same way in all browsers.

**4. Identify React event naming convention**

React uses **camelCase** for naming event handlers, unlike HTML which uses lowercase.

| **HTML Event Attribute** |
| --- |
| onclick |
| onchange |
| **React Event Handler** |
| onClick |
| onChange |

**Event Examples App**

**App.js:**

import React, { useState } from 'react';

import CurrencyConvertor from './CurrencyConvertor';

function App() {

  const [count, setCount] = useState(5);

  const increment = () => {

    setCount(prev => prev + 1);

    sayHello();

    showMessage();

  };

  const decrement = () => {

    setCount(prev => prev - 1);

  };

  const sayHello = () => {

    console.log("Hello");

  };

  const showMessage = () => {

    console.log("Static message shown");

  };

  const sayWelcome = (message) => {

    alert(message);

  };

  const handleClick = (e) => {

    alert("I was clicked");

  };

  return (

    <div style={{ padding: '20px', fontFamily: 'Arial' }}>

      <div>{count}</div>

      <button onClick={increment}>Increment</button><br />

      <button onClick={decrement}>Decrement</button><br />

      <button onClick={() => sayWelcome("welcome")}>Say welcome</button><br />

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

      <br /><br />

      <CurrencyConvertor />

    </div>

  );

}

export default App;

**CurrencyConverter.js:**

import React, { useState } from 'react';

function CurrencyConvertor() {

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

  const [converted, setConverted] = useState('');

  const handleSubmit = () => {

    const inr = parseFloat(amount) \* 80;

    alert(`Converting to  Euro Amount is ${inr}`);

    setConverted('Euro');

  };

  return (

    <div>

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

      <label>

        Amount:<br />

        <input

          type="number"

          value={amount}

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

        />

      </label>

      <br /><br />

      <label>

        Currency:<br />

        <textarea value={converted} readOnly />

      </label>

      <br /><br />

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

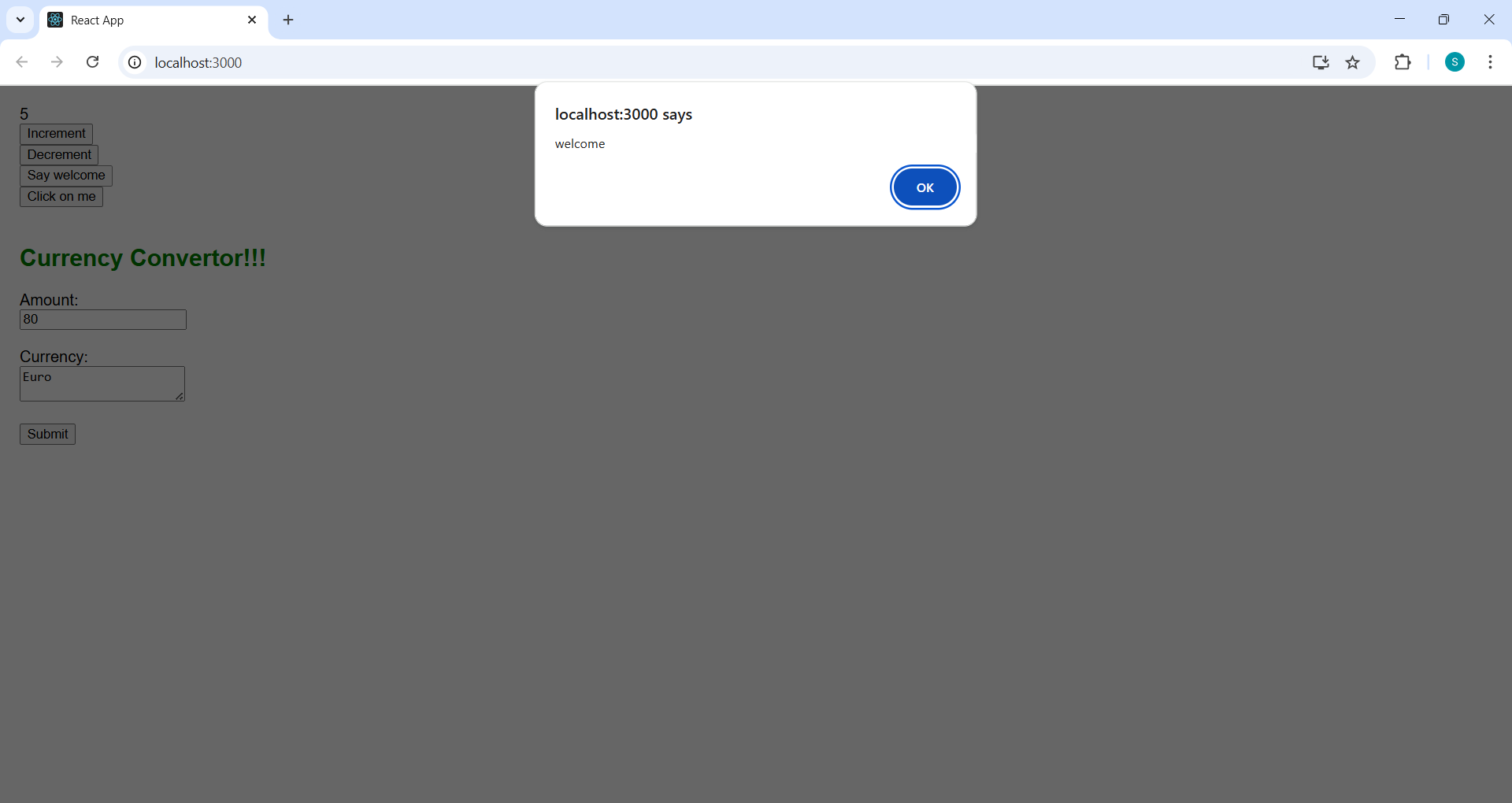
    </div>

  );

}

export default CurrencyConvertor;

**OUTPUT:**



**A screenshot of a computer

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

**ReactJs-HOL-12**

**1.Explain about conditional rendering in React**

Conditional rendering in React allows components to render different UI elements based on certain conditions. It works similarly to how conditions work in JavaScript. This is useful when we want to display or hide elements dynamically, such as showing a login button when a user is not authenticated or a logout button when they are.

**Common Methods for Conditional Rendering:**

**1.Using if-else statements**:

if (isLoggedIn) {

return <LogoutButton />;

} else {

return <LoginButton />;

}

2. **Using ternary operator**:

{isLoggedIn ? <LogoutButton /> : <LoginButton />}

3. **Using logical && operator**:

{isLoggedIn && <WelcomeMessage />}

**2. Define element variables**

Element variables are used in React to store elements in a variable and then include them in the JSX conditionally or dynamically. This helps to simplify code and improve readability. let button;

if (isLoggedIn) {

button = <LogoutButton />;

} else {

button = <LoginButton />;

}

return (

<div>

<h1>Welcome!</h1>

{button}

</div>

);

Here, button is an element variable that holds either the LoginButton or LogoutButton component depending on the condition.

**3. Explain how to prevent components from rendering**

To prevent a component from rendering, you can use conditional rendering and return null from the component. Returning null tells React to render nothing for that component.

function WarningBanner(props) {

if (!props.warn) {

return null;

}

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

}

In this example, if the warn prop is false, the component returns null and nothing will be rendered on the screen.

**Ticket booking App**

**App.js:**

import React, { useState } from 'react';

import LoginButton from './components/LoginButton';

import LogoutButton from './components/LogoutButton';

import Greeting from './components/Greeting';

import GuestPage from './components/GuestPage';

import UserPage from './components/UserPage';

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' }}>

      <Greeting isLoggedIn={isLoggedIn} />

      {button}

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

    </div>

  );

}

export default App;

**LoginButton.js:**

import React from 'react';

function LoginButton(props) {

return (

<button onClick={props.onClick}>

Login

</button>

);

}

export default LoginButton;

**LogoutButton.js:**

import React from 'react';

function LogoutButton(props) {

return (

<button onClick={props.onClick}>

Logout

</button>

);

}

export default LogoutButton;

**Greeting.js:**

import React from 'react';

function UserGreeting() {

return <h1>Welcome back</h1>;

}

function GuestGreeting() {

return <h1>Please sign up.</h1>;

}

function Greeting(props) {

const isLoggedIn = props.isLoggedIn;

if (isLoggedIn) {

return <UserGreeting />;

}

return <GuestGreeting />;

}

export default Greeting;

**UserPage.js:**

import React from 'react';

function UserPage() {

return (

<div>

<h2>Book Your Tickets</h2>

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

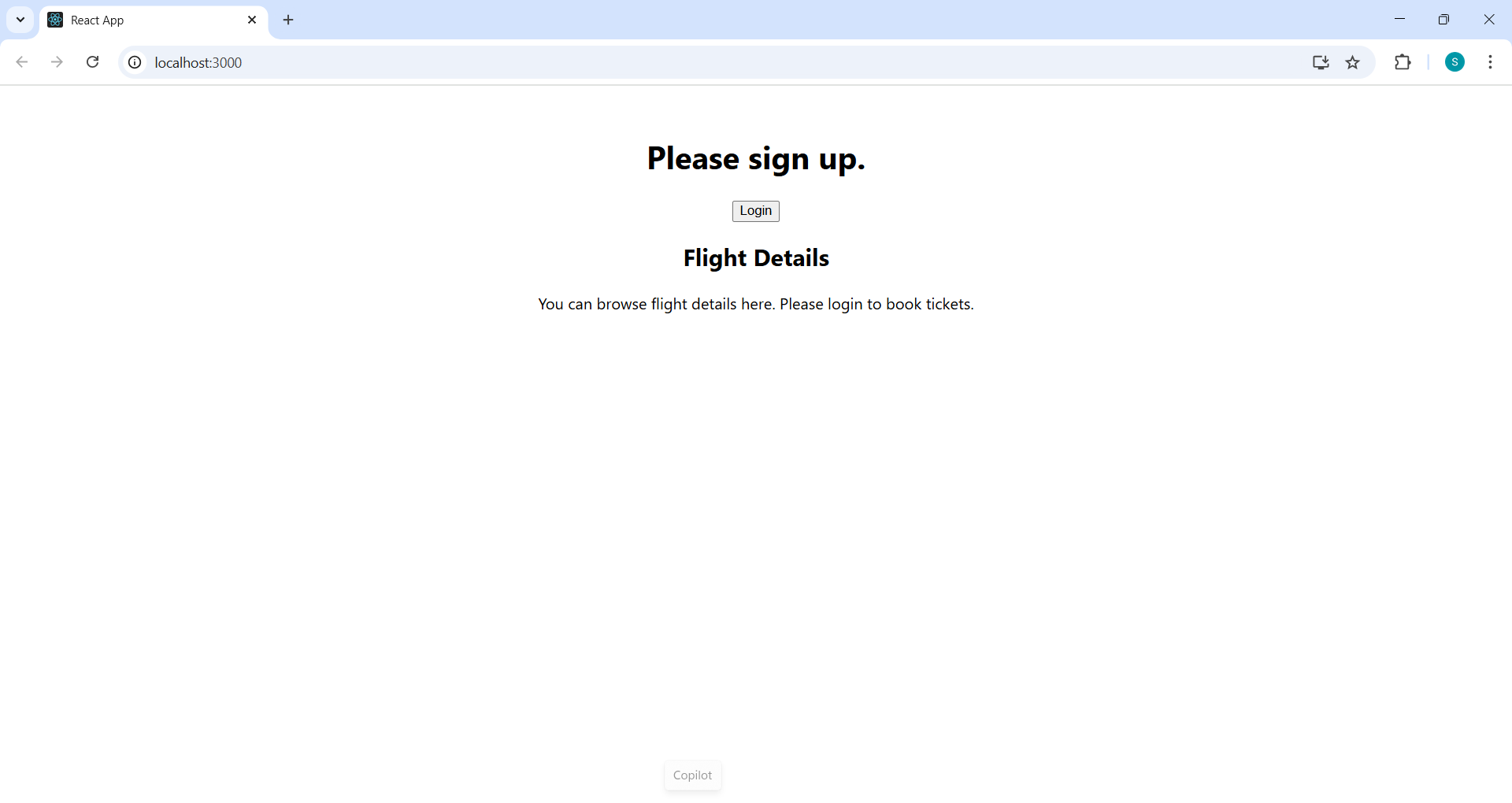
{/\* Additional booking form can be added here \*/}

</div>

);

}

export default UserPage;



A screenshot of a computer

AI-generated content may be incorrect.

**ReactJs-HOL-13**

**1.Rendering Multiple Components in React**

You can render multiple components by placing them together inside a parent element like a div, <> (React fragment), or any container component.

function App() {

return (

<>

<Header />

<MainContent />

<Footer />

</>

);

}

**2. Define list component**

A List component is used to render a collection of data items using JSX. You typically use the map() function to loop through the data and generate multiple elements.

function NameList() {

const names = ['Alice', 'Bob', 'Charlie'];

return (

<ul>

{names.map(name => <li key={name}>{name}</li>)}

</ul>

);

}

**3. Explain about keys in React applications**

Keys are special string attributes you need to include when creating lists of elements in React. They help React identify which items have changed, been added, or removed.

**🔹 Importance:**

* Improve performance
* Prevent rendering issues
* Enable efficient diffing in the virtual DOM

const numbers = [1, 2, 3];

const listItems = numbers.map(number =>

<li key={number.toString()}>{number}</li>

);

**4. Explain how to extract components with keys**

When rendering a list, you may extract each item into a separate component. The key must still be passed to the element in the list, not inside the extracted component.

**Example:**

function ListItem(props) {

return <li>{props.value}</li>;

}

function NumberList(props) {

const numbers = props.numbers;

return (

<ul>

{numbers.map((number) =>

<ListItem key={number.toString()} value={number} />

)}

</ul>

);

}

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

The map() function is a JavaScript method used in React to transform an array into a list of elements.

* Easily convert data arrays into JSX elements
* Dynamically render content
* Helps maintain clean and readable code

**Example:**

const fruits = ['Apple', 'Banana', 'Mango'];

const fruitList = fruits.map((fruit, index) =>

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

);

return <ul>{fruitList}</ul>;

**Blogger App:**

**App.js:**

import React, { useState } from 'react';

import './App.css';

import { books, blogs, courses } from './data';

import BookDetails from './components/BookDetails';

import BlogDetails from './components/BlogDetails';

import CourseDetails from './components/CourseDetails';

function App() {

  const [showBooks, setShowBooks] = useState(true);

  const [showBlogs, setShowBlogs] = useState(true);

  const [showCourses, setShowCourses] = useState(true);

  const bookdet = showBooks ? <BookDetails books={books} /> : <p>No books available.</p>;

  const content = showBlogs && <BlogDetails blogs={blogs} />;

  const coursedet = (() => {

    if (showCourses) {

      return <CourseDetails courses={courses} />;

    } else {

      return <p>Course details are hidden.</p>;

    }

  })();

  return (

    <div className="container">

      <div className="section separator">

        <h1>Course Details</h1>

        {coursedet}

      </div>

      <div className="section separator">

        <h1>Book Details</h1>

        {bookdet}

      </div>

      <div className="section">

        <h1>Blog Details</h1>

        {content}

      </div>

    </div>

  );

}

export default App;

**App.css:**

.container {

  display: flex;

  justify-content: space-around;

  padding: 50px;

  font-family: Arial, sans-serif;

}

.section {

  width: 30%;

  padding: 10px;

}

.separator {

  border-right: 4px solid green;

}

**BookDetails.js:**

import React from 'react';

function BookDetails(props) {

  return (

    <ul>

      {props.books.map((book) => (

        <div key={book.id}>

          <h3>{book.bname}</h3>

          <h4>{book.price}</h4>

        </div>

      ))}

    </ul>

  );

}

export default BookDetails;

**BlogDetails.js:**

import React from 'react';

function BlogDetails({ blogs }) {

  return (

    <div>

      {blogs.map((blog, index) => (

        <div key={index}>

          <h3>{blog.title}</h3>

          <h5>{blog.author}</h5>

          <p>{blog.content}</p>

        </div>

      ))}

    </div>

  );

}

export default BlogDetails;

**CourseDetails.js:**

import React from 'react';

function CourseDetails({ courses }) {

  return (

    <div>

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

        <div key={index}>

          <h3>{course.name}</h3>

          <p>{course.date}</p>

        </div>

      ))}

    </div>

  );

}

export default CourseDetails;

**OUTPUT:**

