Week - 7

1. ReactJS-HOL:

1.List the features of ES6:

ES6 brought significant improvements to JavaScript, introducing new syntax and features that made programming easier, more robust, and more maintainable. Major features include:

* let and const for block-scoped variable declarations.
* Arrow functions for concise function syntax.
* Classes using the class syntax.
* Template literals for string interpolation.
* Destructuring for extracting data from objects/arrays.
* Spread and Rest operators (...) for more flexible data handling.
* Promises for improved asynchronous code flow.
* Modules for code organization (import/export).
* Default parameters in functions.
* Enhanced object literals and more

2.Explain JavaScript let:

let declares a variable with block scope.

The variable is only accessible inside the block (e.g., {}) where it's declared.

No variable hoisting to the top of the block (there's a "Temporal Dead Zone" before declaration).

Cannot re-declare a let variable in the same scope—helps avoid accidental overwrites.

Syntax for let:

let x =10;

1. Identify the differences between var and let:

|  |  |  |
| --- | --- | --- |
| Aspect | var | let |
| Scope | Function or global | Block |
| Redecleration | Allowed | Not allowed |
| Hoisting | Yes, with initialization as undefined | Hoisted but not initialized |
| Accessibility in loop | Accessible outside loop block | Only inside loop block |
| Use case | Legacy code | Modern, preferred |

let is generally safer due to block scoping and no accidental redeclaration

1. Explain JavaScript const:

const is for block-scoped variables that cannot be re-assigned after initialization.

Must be initialized at the time of declaration.

Only the binding is constant; object properties/array items can still be changed.

Syntax:

const PI = 3.14;

If we try to re-assign a const variable, we get an error.

1. Explain ES6 class fundamentals:

Classes are blueprints for creating objects with shared properties and methods.

Syntax is more like other OOP languages.

Supports constructors and method definitions directly inside the class body.

1. Explain ES6 class inheritance:

Use extends to create a subclass that inherits from a parent class.

The super() function is used to call the parent class's constructor and methods.

Enables code reuse by inheriting properties and methods from base classes

1. Define ES6 arrow functions:

Arrow functions offer a shorter syntax for writing functions:

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

They do not have their own this, arguments, or super, and are best for non-method functions and callbacks.

Syntax is more concise, especially for single-line returns.

1. Identify set(), map():

Set:

A collection of unique values (no duplicates).

Supports methods like add(), delete(), has(), clear().

For example:

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

set.add(4);

Map:

A key-value pair collection.

Keys can be of any type.

Methods include set(key, value), get(key), has(key), delete(key).

For example:

let map = new Map();

map.set('a', 1);

map.set('b', 2);

map.get('a');

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() {

  // Sample players array (REQUIRED for flag === true)

  const players = [

    { name: "Jack", score: 70 },

    { name: "Michael", score: 70 },

    { name: "Smith", score: 90 },

    { name: "Rahul", score: 56 },

    { name: "Amit", score: 41 },

    { name: "John", score: 65 },

    { name: "James", score: 91 },

    { name: "Sohail", score: 55 },

    { name: "Dinesh", score: 67 },

    { name: "Manku", score: 55 },

    { name: "Rinku", score: 75 }

  ];

  const IndianTeam = ["Sachin", "Dhoni", "Virat", "Rahul", "Yuvraj", "Bumrah"];

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

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

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

  const flag = false;  // Toggle between true/false for different UI

  if (flag) {

    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>

        <h1>Indian Team</h1>

        <h2>Odd Players</h2>

        <OddPlayers players={IndianTeam} />

        <h2>Even Players</h2>

        <EvenPlayers players={IndianTeam} />

        <hr />

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

        <ListofIndianPlayers IndianPlayers={IndianPlayers} />

      </div>

    );

  }

}

export default App;

EvenPlayers.js:

import React from 'react';

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

  return (

    <ul>

      <li>Second: {second}</li>

      <li>Fourth: {fourth}</li>

      <li>Sixth: {sixth}</li>

    </ul>

  );

}

OddPlayers.js:

import React from 'react';

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

  return (

    <ul>

      <li>First: {first}</li>

      <li>Third: {third}</li>

      <li>Fifth: {fifth}</li>

    </ul>

  );

}

ListofIndianPlayers.js:

import React from 'react';

function ListofIndianPlayers({ IndianPlayers }) {

  return (

    <ul>

      {IndianPlayers.map((name, idx) => (

        <li key={idx}>{name}</li>

      ))}

    </ul>

  );

}

export default ListofIndianPlayers;

ListOfPlayers.js:

import React from 'react';

function ListofPlayers({ players }) {

  return (

    <ul>

      {players.map((item, idx) => (

        <li key={idx}>

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

        </li>

      ))}

    </ul>

  );

}

export default ListofPlayers;

ScoreBelow70.js:

import React from 'react';

function Scorebelow70({ players }) {

  // Filter using arrow function for ES6 style

  const below70 = players.filter(item => item.score < 70);

  return (

    <ul>

      {below70.map((item, idx) => (

        <li key={idx}>

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

        </li>

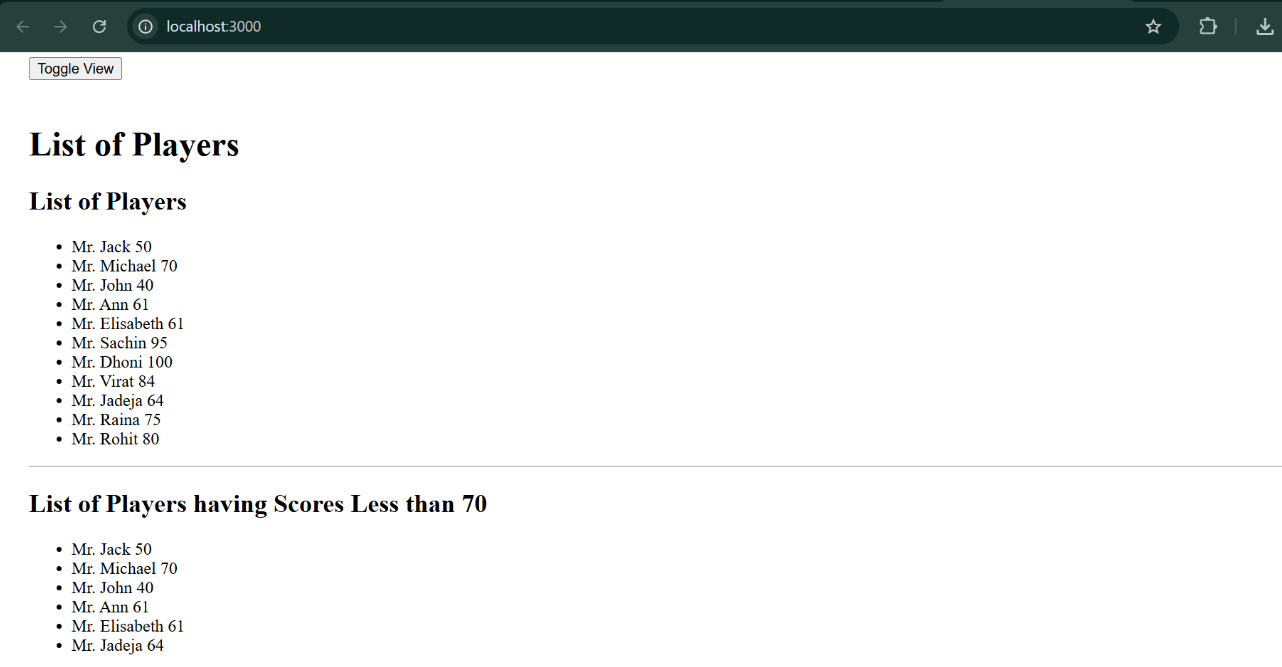
      ))}

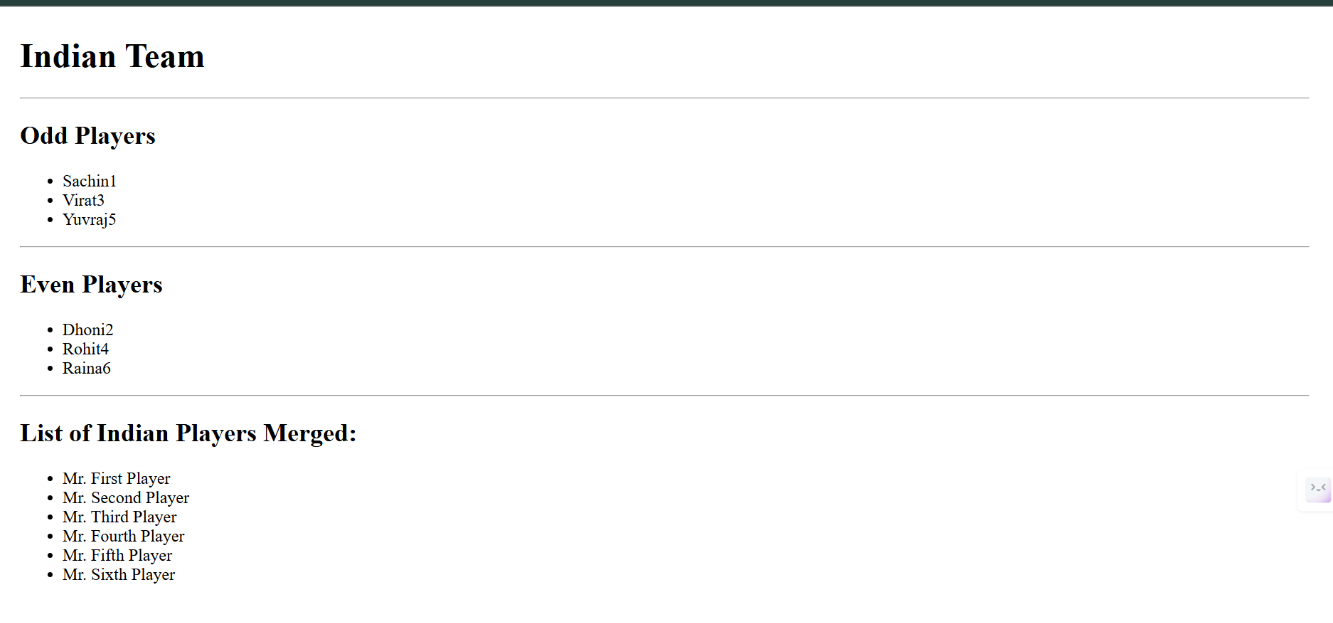
    </ul>

  );

}

export default Scorebelow70;





10.ReactJS-HOL:

· Define JSX

JSX (JavaScript XML) is a syntax extension for JavaScript that looks similar to HTML. It lets you write UI components in a declarative way, mixing JavaScript and markup. Browsers don’t understand JSX directly—your build tools (like Babel) compile JSX to standard JavaScript.

· Explain about ECMA Script

ECMAScript is the official standard that defines the core features of the JavaScript language. Versions of ECMAScript (like ES5, ES6/ES2015, etc.) introduce new language features, syntax, and improvements that are then implemented by JavaScript engines in browsers and Node.js.

Explain React.createElement()

React.createElement() is a React API that creates a virtual DOM node (a React element) from arguments you provide. Typically, you don’t write this by hand since JSX is compiled into these calls.

Explain how to create React nodes with JSX

To create a React node using JSX, write code that looks like HTML within your JavaScript:

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

Define how to render JSX to DOM

You render JSX into the DOM using the root render method:

import ReactDOM from 'react-dom/client';

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

root.render(<App />);

Explain how to use JavaScript expressions in JSX

You use curly braces {} to embed JavaScript expressions (variables, function calls, math, etc.) within JSX. Example:

const name = "Alice";

return <h2>Hello, {name}!</h2>;

Explain how to use inline CSS in JSX

You apply inline CSS by providing a style prop with a JavaScript object, where keys are camelCased CSS property names. Example:

return <div style={{ color: 'red', backgroundColor: 'yellow' }}>Styled!</div>;

Each style property is written as a string value in the object.

**APP.js**

import React from "react";

import "./App.css";

import imgDBS from "./images/dbs.jpg";

import imgRegus from "./images/regus.jpg";

import imgWeWork from "./images/wework.jpg";

function App() {

const element = "Office Space";

const officeList = [

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

{ Name: "Regus", Rent: 75000, Address: "Bangalore", img: imgRegus },

{ Name: "WeWork", Rent: 60000, Address: "Mumbai", img: imgWeWork }

];

return (

<div>

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

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

// Rent color logic

let colors = [];

if (office.Rent <= 60000) {

colors.push("textRed");

} else {

colors.push("textGreen");

}

return (

<div key={index}>

<img src={office.img} width="25%" height="25%" alt={office.Name} />

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

<h3 className={colors.join(" ")}>Rent: Rs. {office.Rent}</h3>

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

<hr />

</div>

);

})}

</div>

);

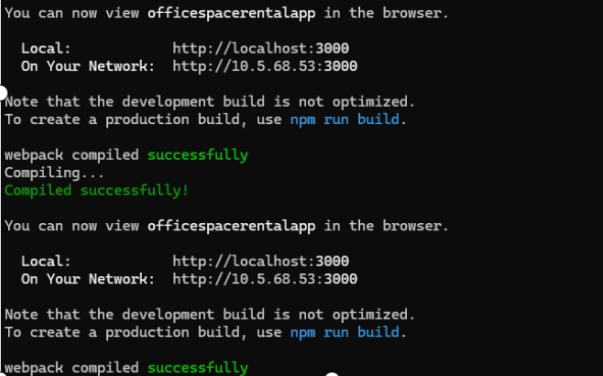
}

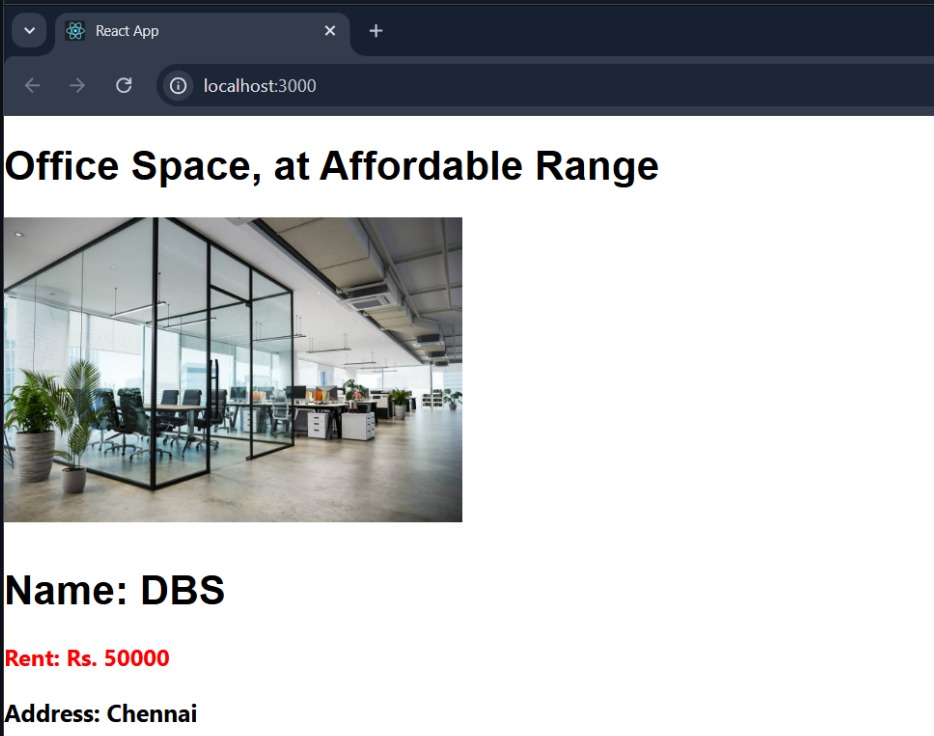
export default App;

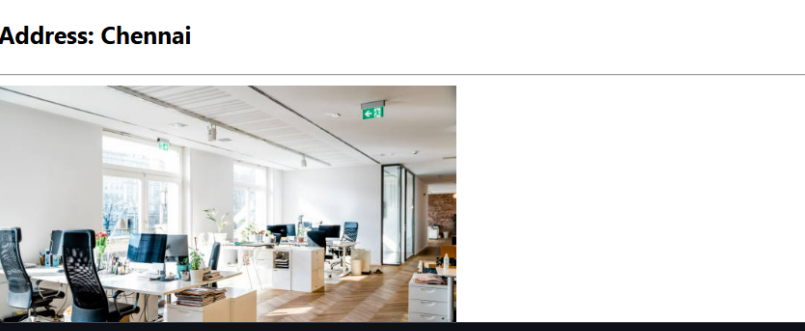
**App.test.js**

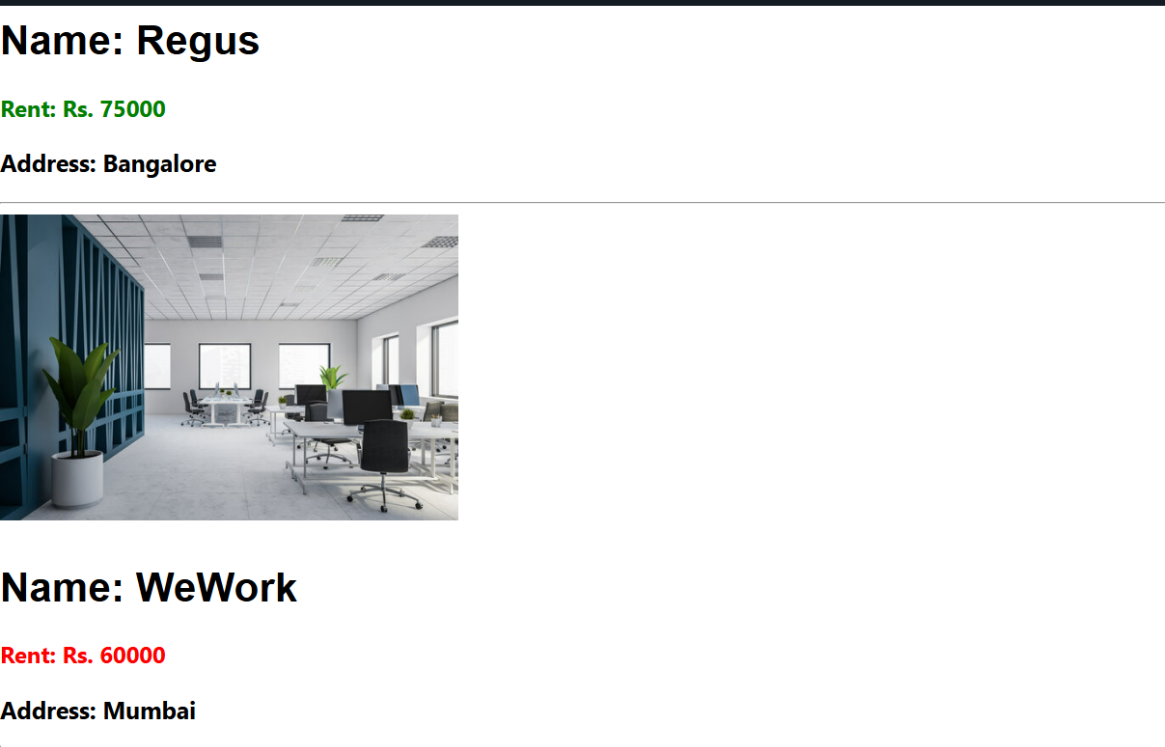


OUTPUT:









**11.**ReactJS-HOL:

**Create “Increment” button to increase the value of the counter and “Decrement” button to decrease the value of the counter. The “Increase” button should invoke multiple methods.**

* 1. **To increment the value**
  2. **Say Hello followed by a static message.**

1. **Create a button “Say Welcome” which invokes the function which takes “welcome” as an argument.**
2. **Create a button which invokes synthetic event “OnPress” which display “I was clicked”**

**Create a “CurrencyConvertor” component which will convert the Indian Rupees to Euro when the Convert button is clicked.**

**Handle the Click event of the button to invoke the handleSubmit event and handle the conversion of the euro to rupees.**

**1. Explain React events:**  
React events are objects that handle user interactions like clicks or form inputs in a React component.

**2. Explain about event handlers:**  
Event handlers are functions defined in React to respond to specific events like onClick or onChange.

**3. Define Synthetic event:**  
A SyntheticEvent is a cross-browser wrapper around the native browser event in React.

**4. Identify React event naming convention:**  
React event names use camelCase (e.g., onClick, onMouseOver) instead of lowercase.

**App.js**

import React, { useState } from "react";

import CurrencyConvertor from "./CurrencyConvertor";

export default function App() {

const [count, setCount] = useState(1);

const increment = () => {

setCount(count + 1);

};

const sayHello = () => {

alert("Hello! Member1");

};

const handleIncrementClick = () => {

increment();

sayHello();

};

const decrement = () => {

setCount(count - 1);

};

const sayWelcome = (msg) => {

alert(msg);

};

const handleClickEvent = () => {

alert("I was clicked");

};

return (

<div>

<p>$</p>

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

<br />

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

<br />

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

<br />

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

<br />

<CurrencyConvertor />

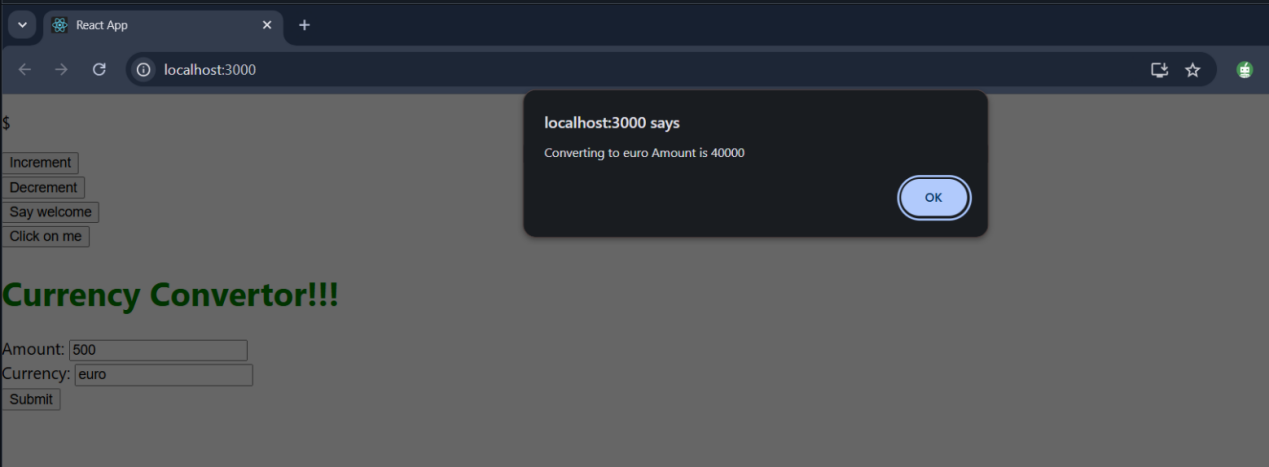
</div>

);

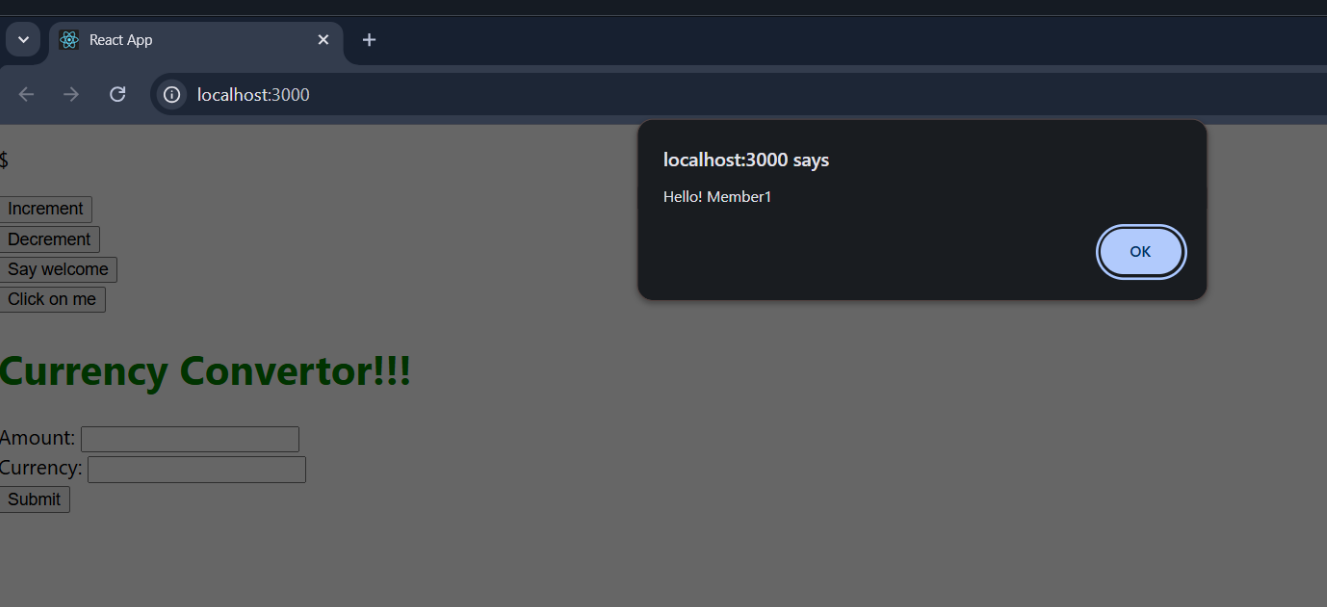
}

**OUTPUT**

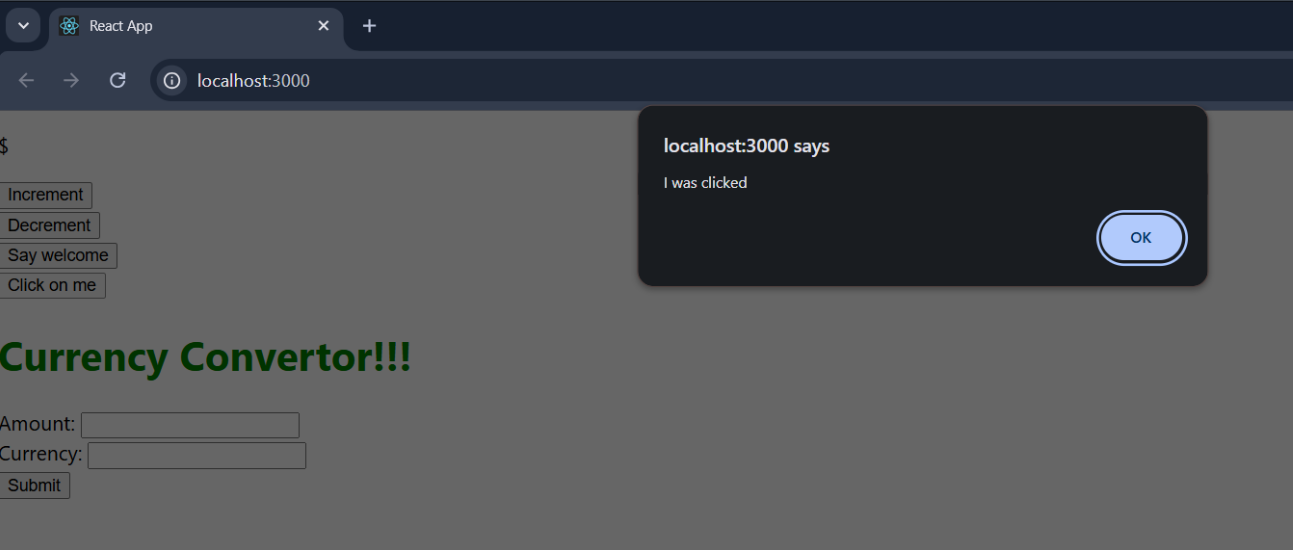
Curreny converter



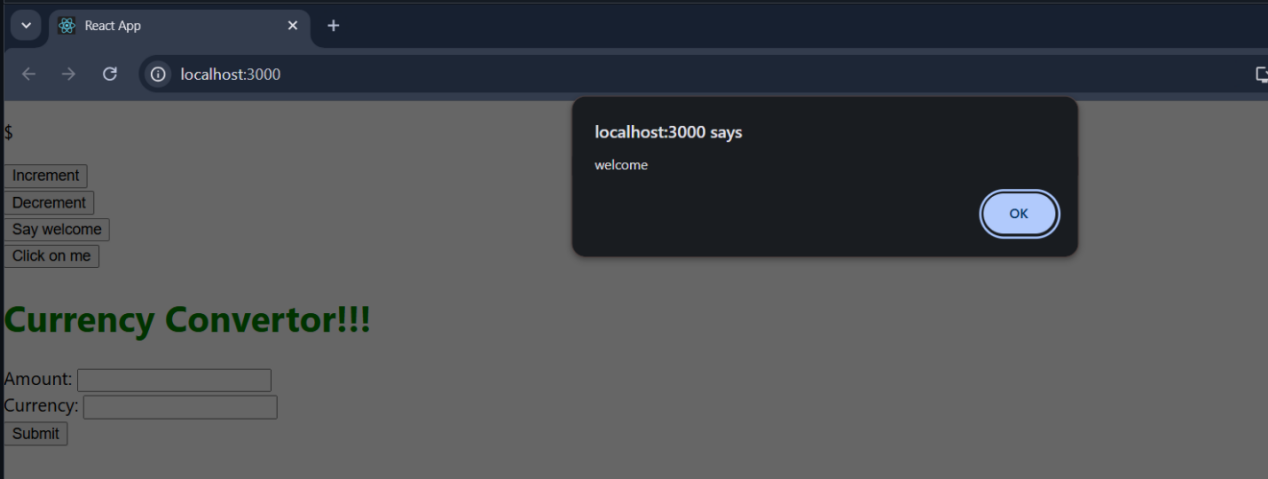
Increment-decrement



On-press



welcome



**12.**ReactJS-HOL:

**Create a React Application named “ticketbookingapp” where the guest user can browse the page where the flight details are displayed whereas the logged in user only can book tickets.**

**The Login and Logout buttons should accordingly display different pages. Once the user is logged in the User page should be displayed. When the user clicks on Logout, the Guest page should be displayed.**

**Ans:**

**Explain about conditional rendering in React:**  
Conditional rendering in React means showing components or elements based on certain conditions using JavaScript logic.

**Define element variables:**  
Element variables are used to store JSX elements in a variable to render them conditionally or dynamically.

**Explain how to prevent components from rendering:**  
Components can be prevented from rendering by returning null or using conditional statements.

**App.js**

import React, { useState } from "react";

import LoginButton from "./LoginButton";

import LogoutButton from "./LogoutButton";

import Greeting from "./Greeting";

export default 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={{ padding: "20px" }}>

<Greeting isLoggedIn={isLoggedIn} />

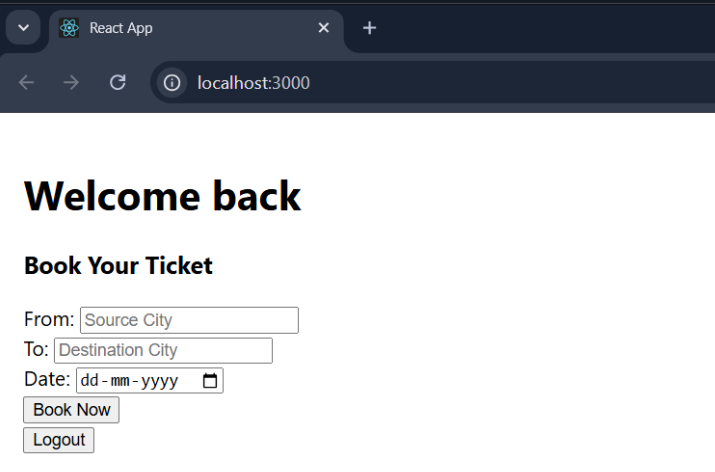
{button}

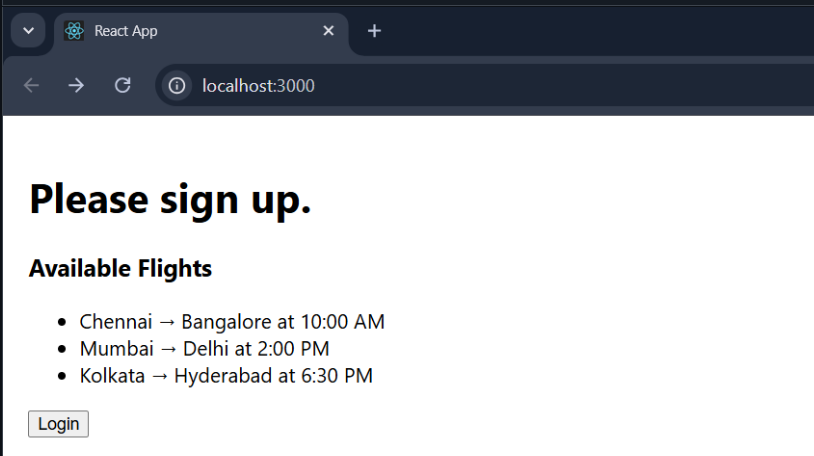
</div>

);

}

**OUTPUT**





**13.**

**Create a React App named “bloggerapp” in with 3 components.**

1. **Book Details**
2. **Blog Details**
3. **Course Details**

**Ans:**

**Explain various ways of conditional rendering:**  
Conditional rendering can be done using if-else, ternary operator, logical &&, or element variables.

**Explain how to render multiple components:**  
Multiple components can be rendered by including them inside a parent component or using fragments (<> </>).

**Define list component:**  
A list component in React renders a collection of similar elements using arrays and the map() function.

**Explain about keys in React applications:**  
Keys are unique identifiers used to help React efficiently update and manage list items.

**Explain how to extract components with keys:**  
You can extract list items into separate components and pass a unique key prop to each instance.

**Explain React Map,** map() **function:**  
The map() function in React is used to iterate over arrays and return a list of elements for rendering.

**App.js**

import React from "react";

import "./App.css";

import BookDetails from "./BookDetails";

import BlogDetails from "./BlogDetails";

import CourseDetails from "./CourseDetails";

export default function App() {

const books = [

{ 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 blogs = [

{

id: 1,

title: "React Learning",

author: "Stephen Biz",

content: "Welcome to learning React!"

},

{

id: 2,

title: "Installation",

author: "Schwezdneier",

content: "You can install React from npm."

}

];

const courses = [

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

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

];

return (

<div style={{ display: "flex", justifyContent: "space-around" }}>

<BookDetails books={books} />

<BlogDetails blogs={blogs} />

<CourseDetails courses={courses} />

</div>

);

}

**BlogDetails.js**

import React from "react";

export default function BlogDetails(props) {

// Conditional Rendering (Ternary Operator)

return (

<div className="v1">

<h1>Blog Details</h1>

{props.blogs.length > 0 ? (

props.blogs.map((blog) => (

<div key={blog.id}>

<h2>{blog.title}</h2>

<h4>{blog.author}</h4>

<p>{blog.content}</p>

</div>

))

) : (

<p>No blogs available.</p>

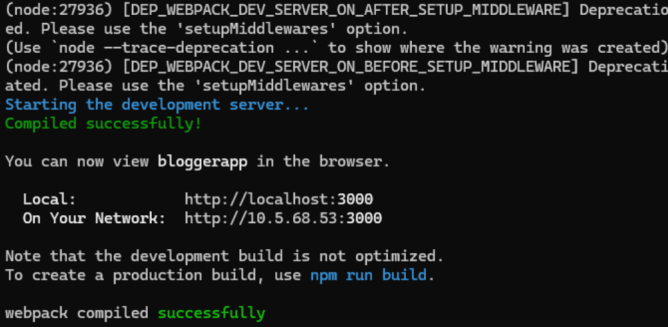
)}

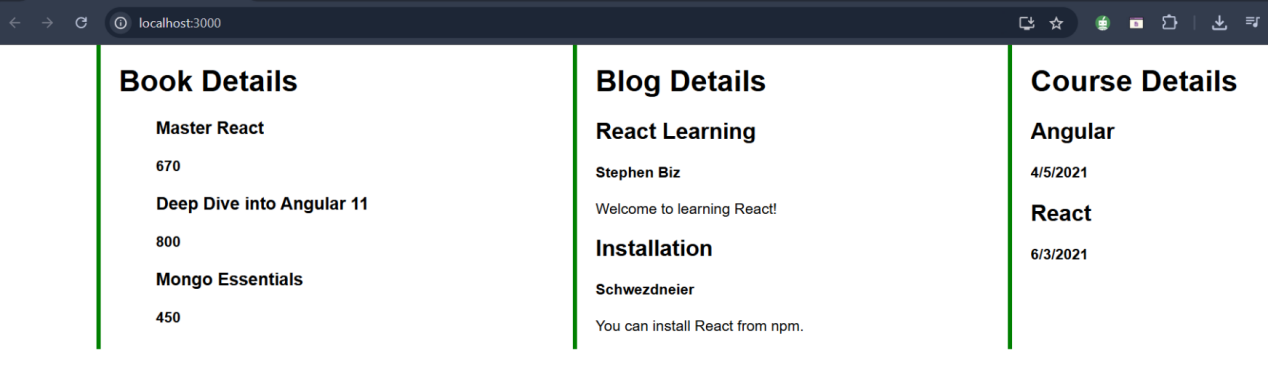
</div>

);

}

**OUTPUT**





**App.js**



**OUTPUT**

