SUPERSET ID - 6364309
Name- Aryabrat Mishra

REACTJS HOL - 9 (LAB1)

THEORIES FOR LAB:

Here's a comprehensive explanation for each of your points related to ES6 (ECMAScript 2015) features:

Features of ES6 (ECMAScript 2015)

1. let and const keywords for block-scoped variables.
2. Arrow functions for concise syntax.
3. Classes and class inheritance (OOP support).
4. Template literals for string interpolation.
5. Default parameters in functions.
6. Destructuring assignment for arrays and objects.
7. Spread (...) and rest (...) operators.
8. Modules (import/export syntax).
9. Promises for asynchronous programming.
10. Map and Set objects.
11. for...of loop.
12. Symbol type.
13. Enhanced object literals (shorthand syntax, computed property names).
14. Iterators and generators.
15. New built-in methods (e.g., Array.from(), Object.assign()).

JavaScript let

1. Introduced in ES6 to declare block-scoped variables.
2. Unlike var, let respects the boundaries of { }.
3. Not hoisted in the same way as var (temporal dead zone applies).
4. Can be reassigned but not redeclared in the same scope.

Example:

```
let x = 10;
if (true) {
  let x = 20;
  console.log(x); // 20
}
console.log(x); // 10
```

✅ Difference between var and let

Feature	var	let
Scope	Function-scoped	Block-scoped ({})

Feature	var	let
Redeclaration	Allowed	Not allowed in same scope
Hoisting	Hoisted (undefined)	Hoisted (TDZ error if used before declaration)
Use case	Legacy code	Preferred in modern JS

JavaScript const

1. Used to declare block-scoped constants.
2. Cannot be reassigned, but the content of objects/arrays declared with const can be modified.
3. Must be initialized at the time of declaration.

Example:

```
const PI = 3.14;
// PI = 3.14159; // Error
const arr = [1, 2];
arr.push(3); // Works
```

ES6 Class Fundamentals

A cleaner and more intuitive syntax for creating objects and inheritance.
Equivalent to constructor functions under the hood.

Example:

```
class Person {
  constructor(name) {
    this.name = name;
  }

  greet() {
    console.log(`Hello, I'm ${this.name}`);
  }
}

const p = new Person("Alice");
p.greet(); // Hello, I'm Alice
```

ES6 Class Inheritance

Classes can extend other classes using extends keyword.
super() is used to call the parent constructor.

Example:

```
class Animal {
  constructor(name) {
    this.name = name;
  }

  speak() {
    console.log(`${this.name} makes a noise`);
  }
}

class Dog extends Animal {
  speak() {
    console.log(`${this.name} barks`);
  }
}

const d = new Dog("Rex");
d.speak(); // Rex barks
```

ES6 Arrow Functions

Concise function syntax: () => {}.
Do not bind their own this, making them useful in callbacks.
Cannot be used as constructors.

Example:
```
const add = (a, b) => a + b;
console.log(add(2, 3)); // 5
```
With no parameters:
```
const greet = () => console.log("Hello");
```

Set and Map in ES6
Set:
Collection of unique values.
Can store any type of value (primitives or object references).

Example:
```
const mySet = new Set();
mySet.add(1);
mySet.add(2);
mySet.add(1); // ignored
console.log(mySet); // Set { 1, 2 }
```

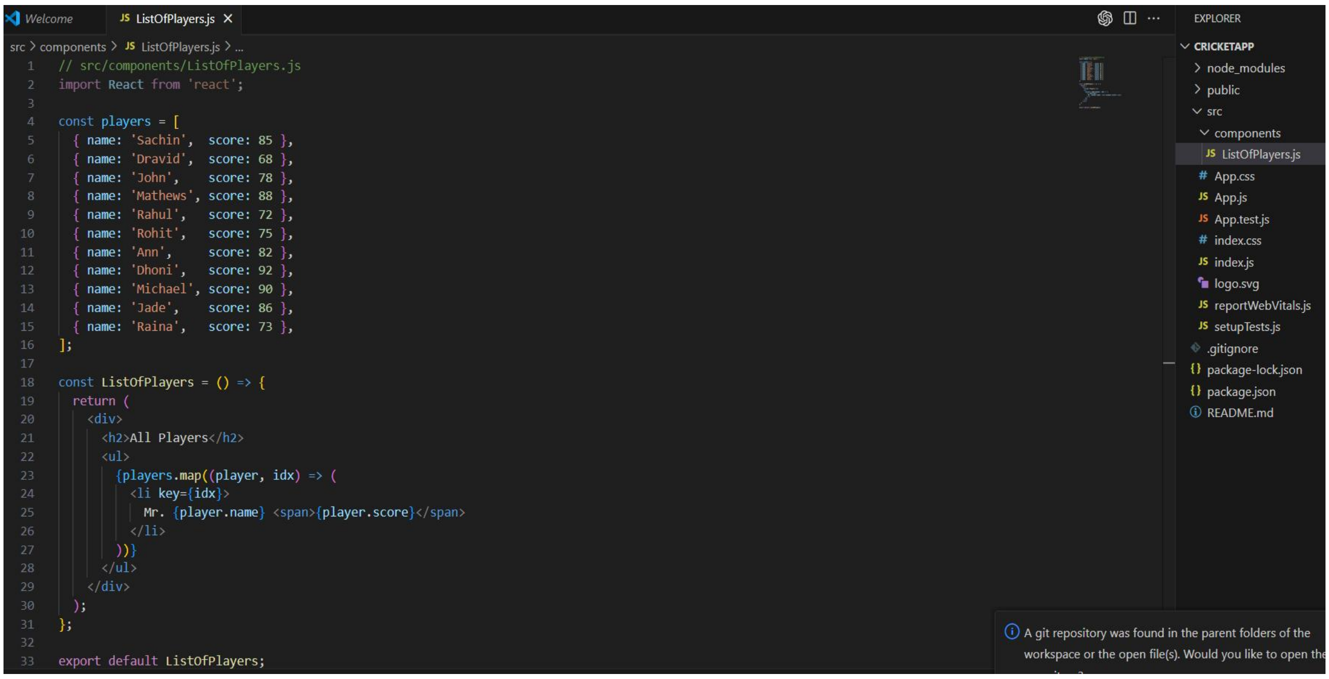
Map:
Collection of key-value pairs.
Keys can be of any type (unlike plain objects where keys are strings or symbols).

```
Example:
const myMap = new Map();
myMap.set('a', 1);
myMap.set(1, 'one');
console.log(myMap.get(1)); // "one"
```

MY WORK LAB1

Create a React Application named “cricketapp” with the following components:

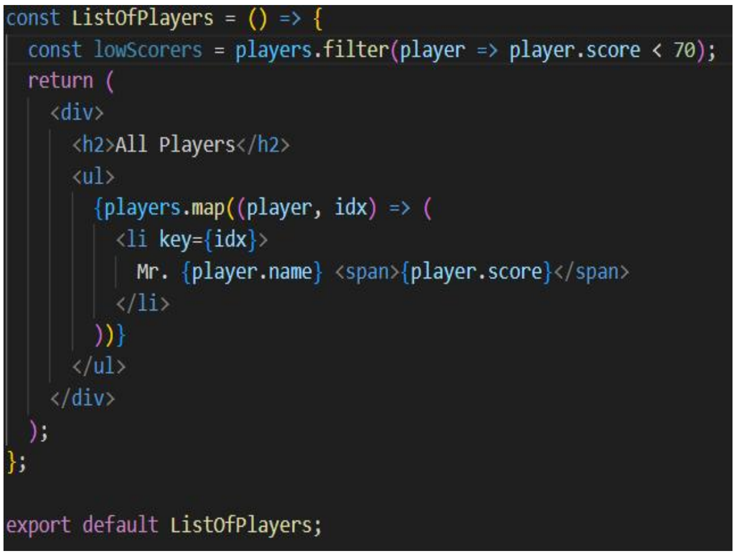
1. ListofPlayers
   - Declare an array with 11 players and store details of their names and scores using the map feature of ES6



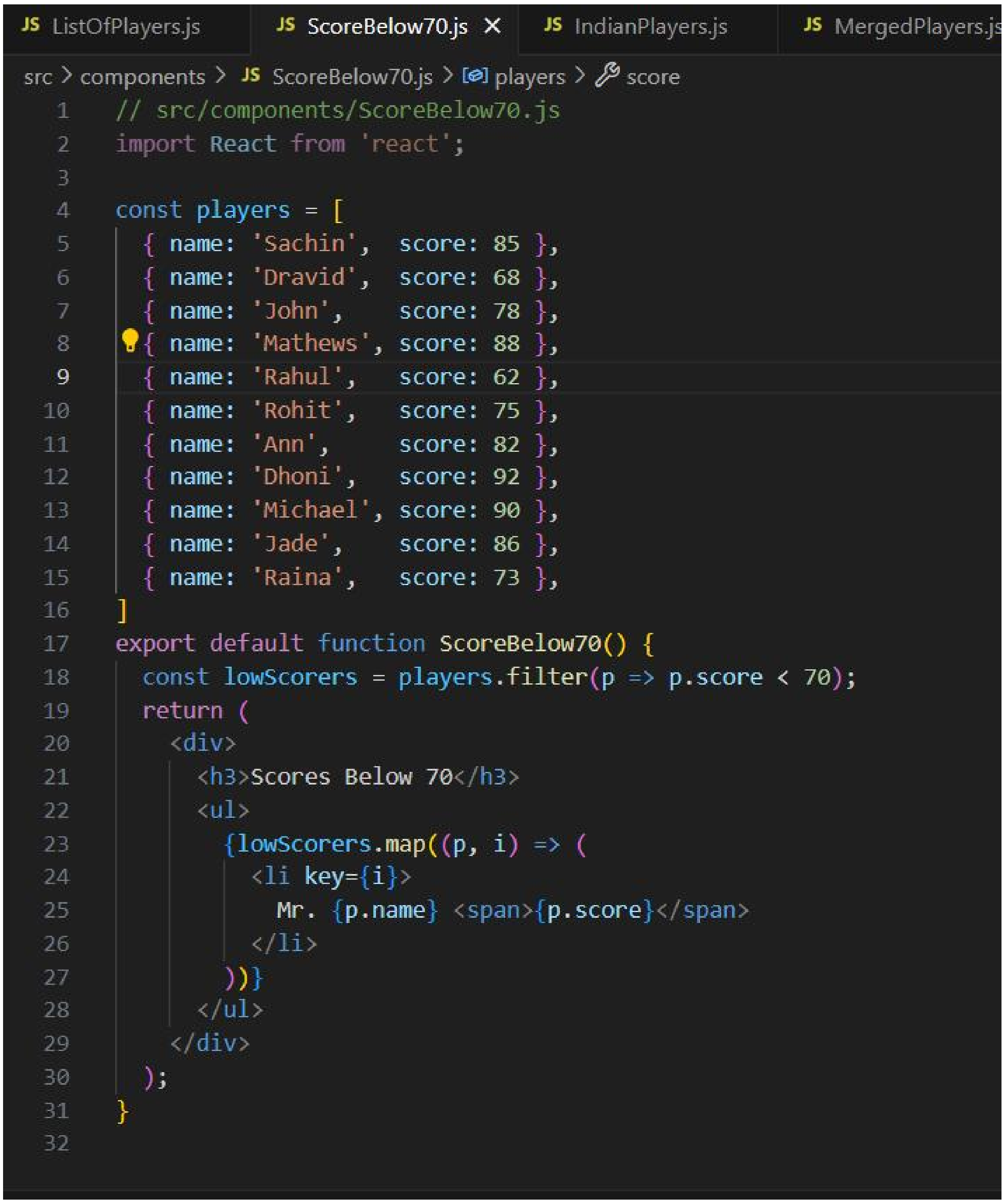
```
// src/components/ListOfPlayers.js
import React from 'react';

const players = [
  { name: 'Sachin',  score: 85 },
  { name: 'Dravid',  score: 68 },
  { name: 'John',    score: 78 },
  { name: 'Mathews', score: 88 },
  { name: 'Rahul',   score: 72 },
  { name: 'Rohit',   score: 75 },
  { name: 'Ann',     score: 82 },
  { name: 'Dhoni',   score: 92 },
  { name: 'Michael', score: 90 },
  { name: 'Jade',    score: 86 },
  { name: 'Raina',   score: 73 },
];

const ListOfPlayers = () => {
  return (
    <div>
      <h2>All Players</h2>
      <ul>
        {players.map((player, idx) => (
          <li key={idx}>
            Mr. {player.name} <span>{player.score}</span>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default ListOfPlayers;
```

A git repository was found in the parent folders of the workspace or the open file(s). Would you like to open the

   - Filter the players with scores below 70 using arrow functions of ES6.



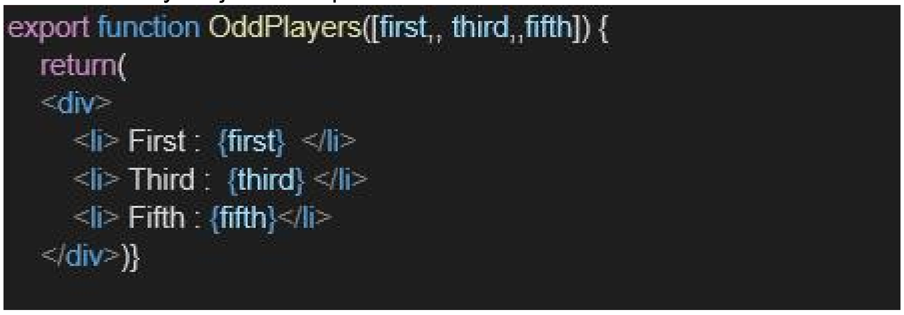
```
const ListOfPlayers = () => {
  const lowScorers = players.filter(player => player.score < 70);
  return (
    <div>
      <h2>All Players</h2>
      <ul>
        {players.map((player, idx) => (
          <li key={idx}>
            Mr. {player.name} <span>{player.score}</span>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default ListOfPlayers;
```



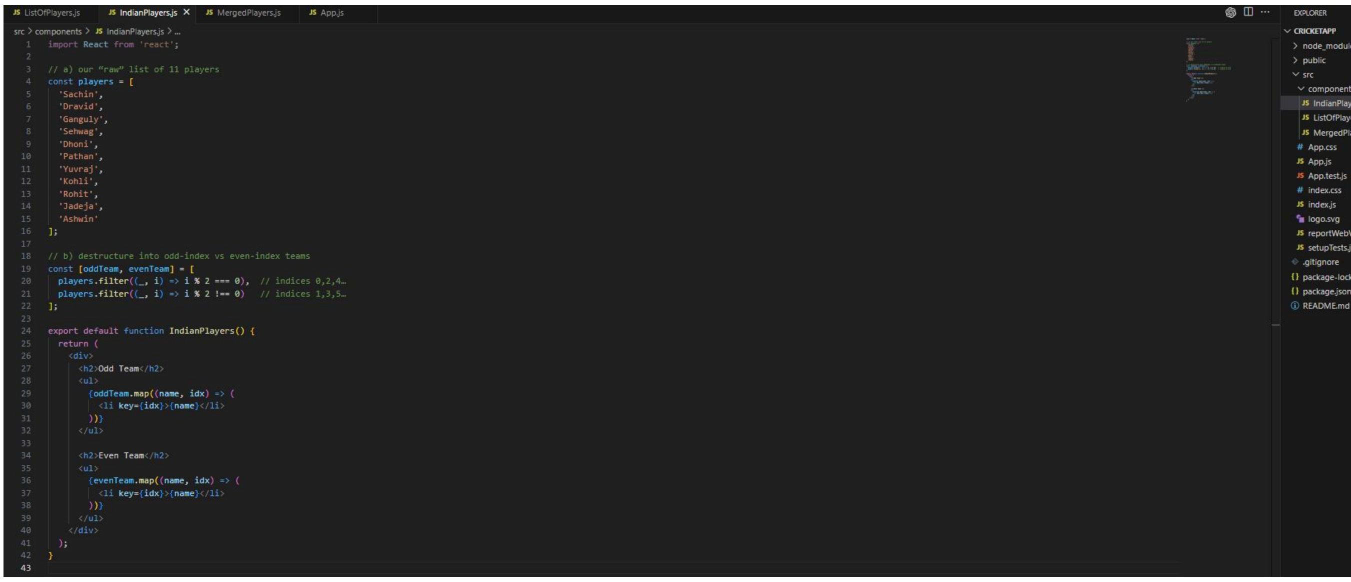
```
// src/components/ScoreBelow70.js
import React from 'react';

const players = [
  { name: 'Sachin',  score: 85 },
  { name: 'Dravid',  score: 68 },
  { name: 'John',    score: 78 },
  { name: 'Mathews', score: 88 },
  { name: 'Rahul',   score: 62 },
  { name: 'Rohit',   score: 75 },
  { name: 'Ann',     score: 82 },
  { name: 'Dhoni',   score: 92 },
  { name: 'Michael', score: 90 },
  { name: 'Jade',    score: 86 },
  { name: 'Raina',   score: 73 },
]
export default function ScoreBelow70() {
  const lowScorers = players.filter(p => p.score < 70);
  return (
    <div>
      <h3>Scores Below 70</h3>
      <ul>
        {lowScorers.map((p, i) => (
          <li key={i}>
            Mr. {p.name} <span>{p.score}</span>
          </li>
        ))}
      </ul>
    </div>
  );
}

```

2. **<u>IndianPlayers</u>**

   a. Display the Odd Team Player and Even Team players using the Destructuring features of ES6 as IndianPlayers.js in Components

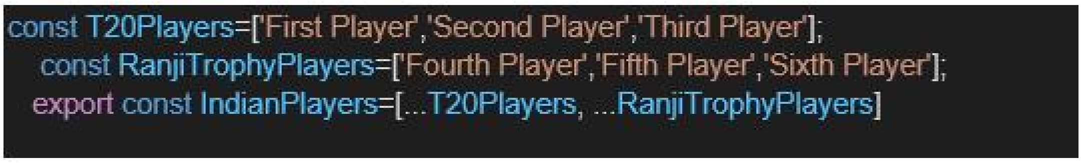


```
export function OddPlayers([first,, third,,fifth]) {
  return(
  <div>
    <li> First : {first} </li>
    <li> Third : {third} </li>
    <li> Fifth : {fifth}</li>
  </div>)}
```

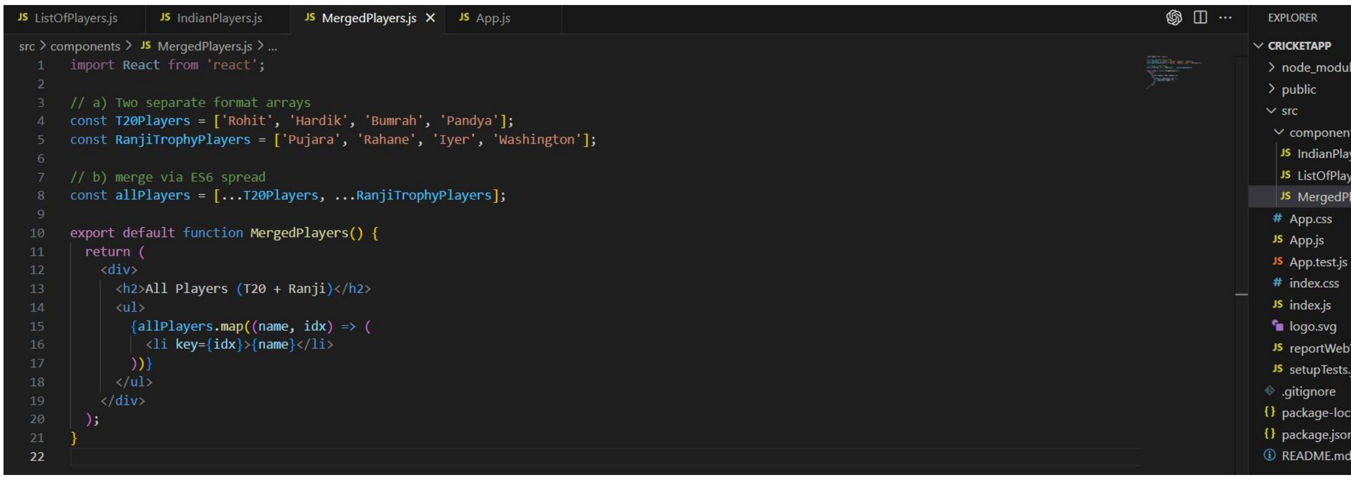


```
import React from 'react';

// a) our “raw” list of 11 players
const players = [
  'Sachin',
  'Dravid',
  'Ganguly',
  'Sehwag',
  'Dhoni',
  'Pathan',
  'Yuvraj',
  'Kohli',
  'Rohit',
  'Jadeja',
  'Ashwin'
];

// b) destructure into odd-index vs even-index teams
const [oddTeam, evenTeam] = [
  players.filter((_, i) => i % 2 === 0),  // indices 0,2,4…
  players.filter((_, i) => i % 2 !== 0)   // indices 1,3,5…
];

export default function IndianPlayers() {
  return (
    <div>
      <h2>Odd Team</h2>
      <ul>
        {oddTeam.map((name, idx) => (
          <li key={idx}>{name}</li>
        ))}
      </ul>

      <h2>Even Team</h2>
      <ul>
        {evenTeam.map((name, idx) => (
          <li key={idx}>{name}</li>
        ))}
      </ul>
    </div>
  );
}
```

b. Declare two arrays T20players and RanjiTrophy players and merge the two arrays and display them using the Merge feature of ES6 as MergedPlayers.js



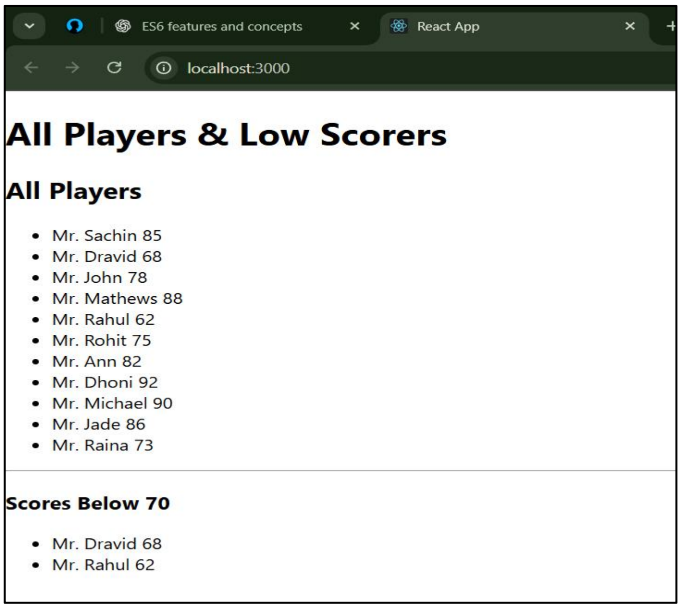
```
const T20Players=['First Player','Second Player','Third Player'];
   const RanjiTrophyPlayers=['Fourth Player','Fifth Player','Sixth Player'];
  export const IndianPlayers=[...T20Players, ...RanjiTrophyPlayers]
```



```
import React from 'react';

// a) Two separate format arrays
const T20Players = ['Rohit', 'Hardik', 'Bumrah', 'Pandya'];
const RanjiTrophyPlayers = ['Pujara', 'Rahane', 'Iyer', 'Washington'];

// b) merge via ES6 spread
const allPlayers = [...T20Players, ...RanjiTrophyPlayers];

export default function MergedPlayers() {
  return (
    <div>
      <h2>All Players (T20 + Ranji)</h2>
      <ul>
        {allPlayers.map((name, idx) => (
          <li key={idx}>{name}</li>
        ))}
      </ul>
    </div>
  );
}
```

Display these two components in the same home page using a simple if else in the flag variable.

Output:
When Flag=true



All Players & Low Scorers

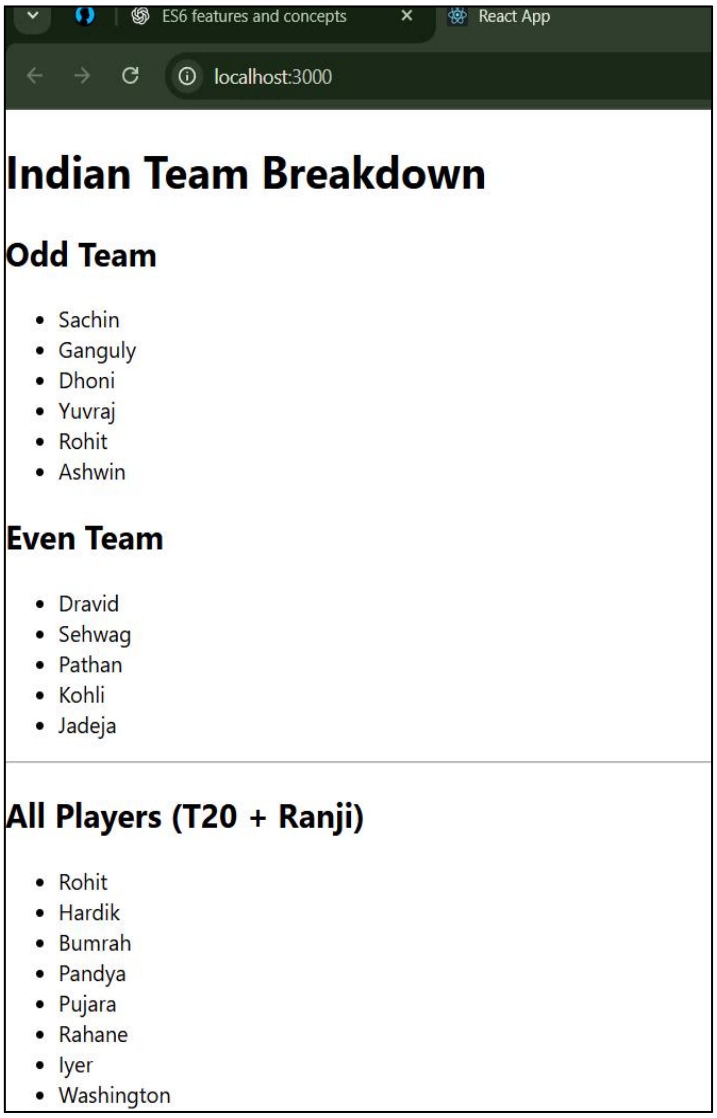
All Players

- Mr. Sachin 85
- Mr. Dravid 68
- Mr. John 78
- Mr. Mathews 88
- Mr. Rahul 62
- Mr. Rohit 75
- Mr. Ann 82
- Mr. Dhoni 92
- Mr. Michael 90
- Mr. Jade 86
- Mr. Raina 73

Scores Below 70

- Mr. Dravid 68
- Mr. Rahul 62

When Flag = false



Indian Team Breakdown

Odd Team

- Sachin
- Ganguly
- Dhoni
- Yuvraj
- Rohit
- Ashwin

Even Team

- Dravid
- Sehwag
- Pathan
- Kohli
- Jadeja

All Players (T20 + Ranji)

- Rohit
- Hardik
- Bumrah
- Pandya
- Pujara
- Rahane
- Iyer
- Washington

REACTJS_HOL_10(LAB 2)

THEORIES FOR LAB:

Here's a complete explanation of your React + ECMAScript + JSX-related questions, in a clear and concise format:

✅ Define JSX (JavaScript XML)

JSX stands for **JavaScript XML**.

It is a **syntax extension** for JavaScript used with **React**.

JSX allows writing **HTML-like code** inside JavaScript.

Makes it easier to create React elements visually close to actual HTML.

Example:

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

Transpiles to:

```
React.createElement('h1', null, 'Hello, world!');
```

✅ Explain ECMA Script

ECMAScript (ES) is the **standardized scripting language specification** upon which **JavaScript** is based.

Developed by **ECMA International**.

ES versions:

- **ES5 (2009)**: Standard JavaScript we used pre-2015.
- **ES6 (2015)** (also called **ES2015**): Introduced `let`, `const`, arrow functions, classes, promises, etc.
- New versions released **yearly** (e.g., ES2017, ES2020, ES2023).

React and modern JS development heavily rely on **ES6+ features**.

✅ Explain `React.createElement()`

It's the **core function** used by React to **create virtual DOM elements**.

JSX is syntactic sugar for `React.createElement`.

Syntax:

```
React.createElement(
  type,         // HTML tag or Component
  props,        // attributes (null if none)
  ...children   // child elements or text
)
```

Example:

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

Equivalent JSX:

```
const element = <h1 className="greet">Hello!</h1>;
```

✅ How to Create React Nodes with JSX

A **React Node** is created using JSX by writing HTML-like syntax in JavaScript.

You can use tags like `<div>`, `<h1>`, `<p>`, or even your own component

Example:

```
const node = <p>This is a React node created using JSX.</p>;
```

Component example:

```
function Greeting() {
  return <h2>Welcome to React</h2>;
}
```

✅ How to Render JSX to the DOM

Use the `ReactDOM.render()` method (for older React versions) or `createRoot().render()` (React 18+):

React 18+ Example:

```
import React from 'react';
import { createRoot } from 'react-dom/client';

const element = <h1>Hello React!</h1>;
const root = createRoot(document.getElementById('root'));
root.render(element);
```

Note: For React <18, use ReactDOM.render(element, container).

✅ How to Use JavaScript Expressions in JSX

You can embed any **JavaScript expression** inside `{}` within JSX.

Examples:

```
const name = "Arya";
const element = <h1>Hello, {name}</h1>;

const total = <p>Total is {5 + 10}</p>;
```

You **cannot** use full statements (like if, for) inside {}—only expressions.

✅ How to Use Inline CSS in JSX

1. Inline styles are written as a **JavaScript object**.
2. Property names must be in **camelCase** (e.g., `backgroundColor`).

Example:

```
const divStyle = {
  color: 'white',
  backgroundColor: 'blue',
  padding: '10px'
};

const element = <div style={divStyle}>Styled text</div>;
```

Direct inline:

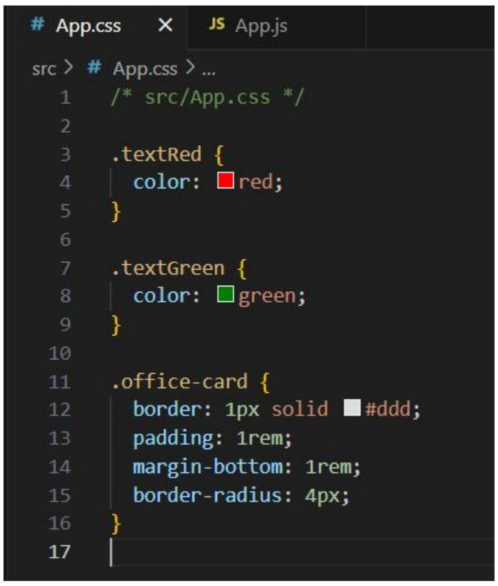
```
<div style={{ fontSize: '20px', color: 'red' }}>Inline styled</div>
```

MY WORK LAB2

Create a React Application named “officespacerentalapp” which uses React JSX to create elements, attributes and renders DOM to display the page.

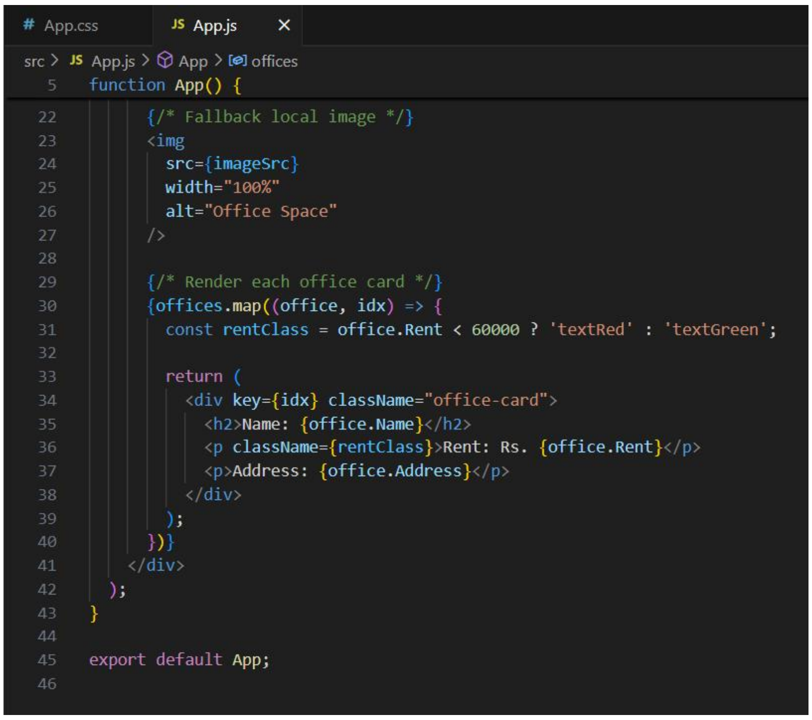
Create an element to display the heading of the page.
Attribute to display the image of the office space
Create an object of office to display the details like Name, Rent and Address.
Create a list of Object and loop through the office space item to display more data.
To apply Css, Display the color of the Rent in Red if it’s below 60000 and in Green if it’s above 60000.

Src\App.css



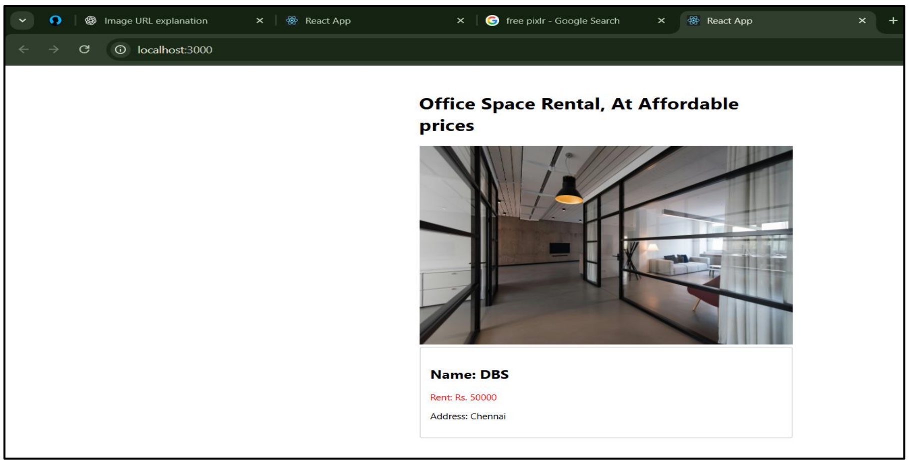
```
/* src/App.css */

.textRed {
  color: red;
}

.textGreen {
  color: green;
}

.office-card {
  border: 1px solid #ddd;
  padding: 1rem;
  margin-bottom: 1rem;
  border-radius: 4px;
}
```

Src\App.js



```
function App() {
      {/* Fallback local image */}
      <img
        src={imageSrc}
        width="100%"
        alt="Office Space"
      />

      {/* Render each office card */}
      {offices.map((office, idx) => {
        const rentClass = office.Rent < 60000 ? 'textRed' : 'textGreen';

        return (
          <div key={idx} className="office-card">
            <h2>Name: {office.Name}</h2>
            <p className={rentClass}>Rent: Rs. {office.Rent}</p>
            <p>Address: {office.Address}</p>
          </div>
        );
      })}
    </div>
  );
}

export default App;
```

Output:



Office Space Rental, At Affordable prices

Name: DBS

Rent: Rs. 50000

Address: Chennai

REACTJS-HOL11(LAB 3)

THEORIES FOR LAB:

Here’s a complete and beginner-friendly breakdown of your React events-related questions:

✅ Explain React Events

React **events** are similar to DOM events (like `onclick`, `onchange`) but are handled using **React’s event system**.

Events in React are **wrapped in a cross-browser wrapper** called a **SyntheticEvent**.

React supports all the standard DOM events like:

`onClick`, `onChange`, `onSubmit`, `onMouseOver`, `onKeyDown`, etc.

Example:

```
function handleClick() {
  alert("Button clicked!");
}

function App() {
  return <button onClick={handleClick}>Click Me</button>;
}
```

✅ Explain Event Handlers in React

An **event handler** is a function that is called when an event occurs.

In React, you typically define a function and pass it to an element as a **prop** using **camelCase syntax**.

Example:

```
function App() {
  const handleChange = (event) => {
    console.log("Input changed to:", event.target.value);
  };

  return <input type="text" onChange={handleChange} />;
}
```

✅ Define Synthetic Event

A **SyntheticEvent** is a **cross-browser wrapper** around the native browser event.

It ensures consistent behavior across different browsers.

It contains all the standard properties and methods of native events.

React automatically **pools** SyntheticEvents for performance, meaning the event object might be reused (unless you call `event.persist()`).

Example:

```
function handleClick(e) {
  console.log(e); // SyntheticEvent object
}
```

✅ React Event Naming Convention

Feature	React Event Convention	DOM Convention
Naming	CamelCase	Lowercase
Assignment	Pass a **function**, not string	Pass a string in HTML

Feature	React Event Convention	DOM Convention
Binding	Done via arrow functions or binding	N/A in HTML

Examples:

DOM (HTML)	React (JSX)
<button onclick="doSomething()">	<button onClick={doSomething}>
<form onsubmit="submitForm()">	<form onSubmit={submitForm}>

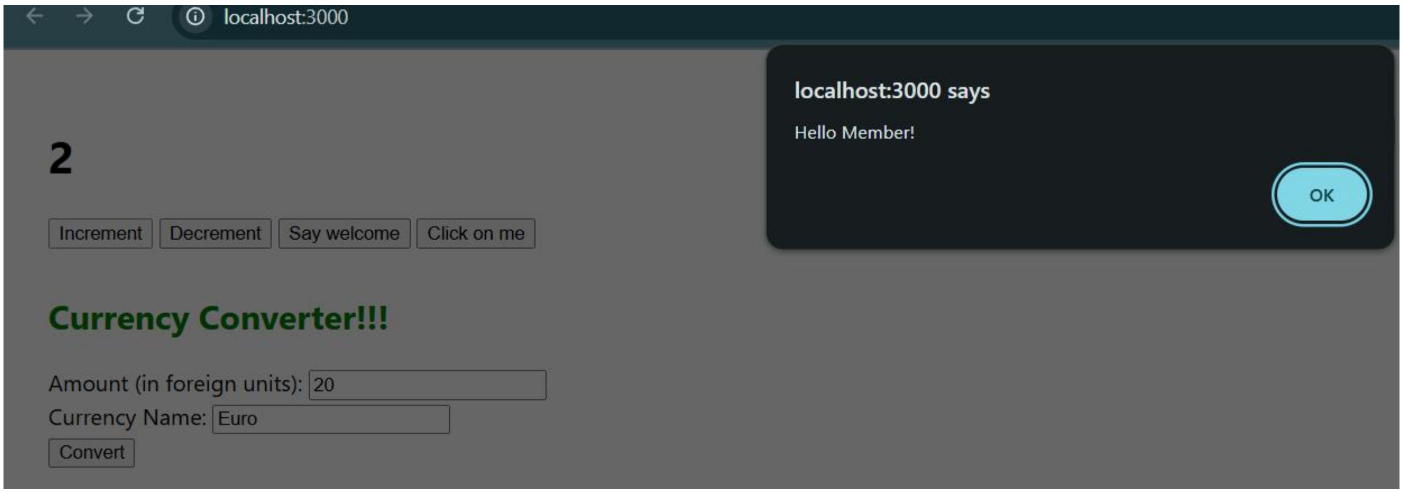
✅ Quick Recap:

Concept	Description
React Events	React's way to handle user interactions, similar to DOM events.
Event Handler	A function that gets triggered when a specific event occurs.
Synthetic Event	React's wrapper for native events, for consistent cross-browser behavior.
Naming	Use camelCase like onClick, onChange, and pass functions directly.

MY WORK LAB3:

Create a React Application "eventexamplesapp" to handle various events of the form elements in HTML.

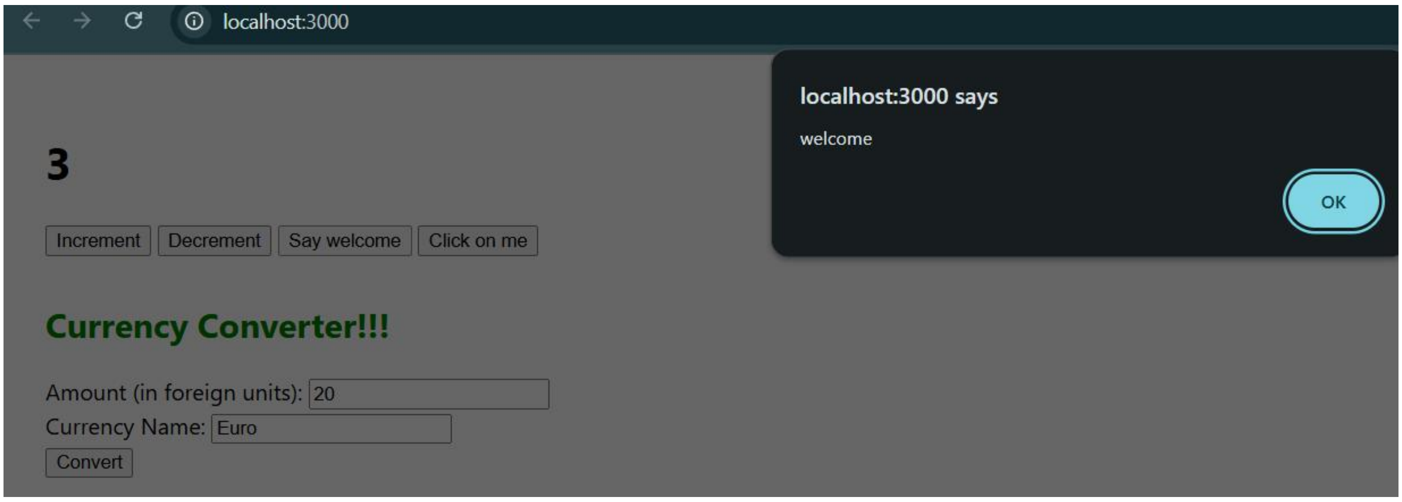
1. Create "Increment" button to increase the value of the counter and "Decrement" button to decrease the value of the counter. The "Increase" button should invoke multiple methods.
   a. To increment the value
   b. Say Hello followed by a static message.



localhost:3000 says

Hello Member!

2. Create a button "Say Welcome" which invokes the function which takes "welcome" as an argument.


localhost:3000

localhost:3000 says

welcome

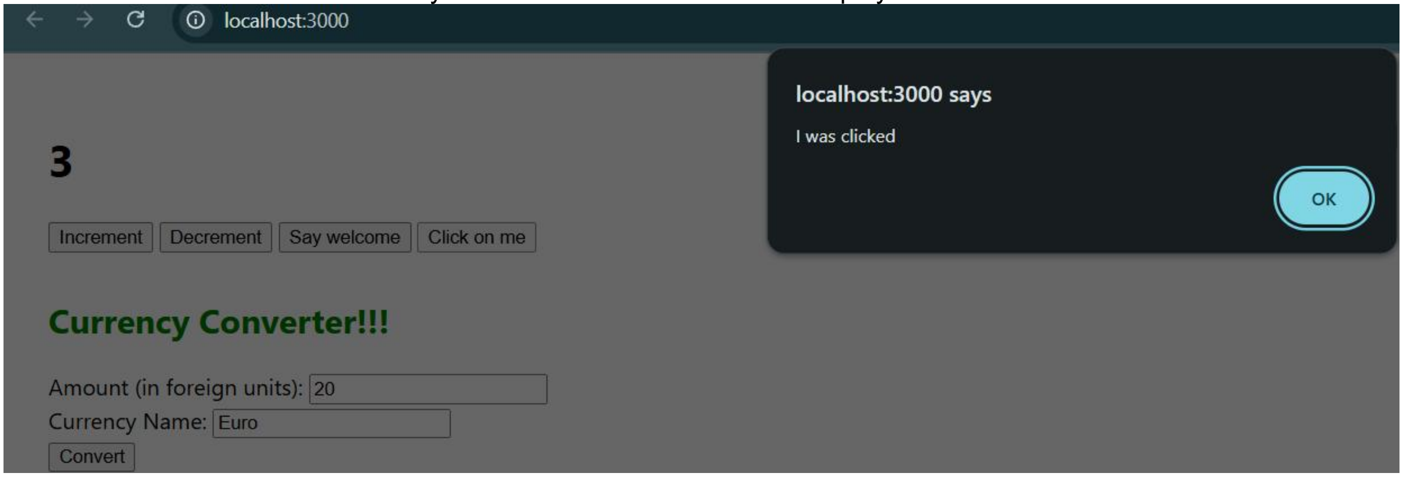
3

Currency Converter!!!

Amount (in foreign units): 20

Currency Name: Euro

3. Create a button which invokes synthetic event “OnPress” which display “I was clicked”


localhost:3000

localhost:3000 says

I was clicked

3

Currency Converter!!!

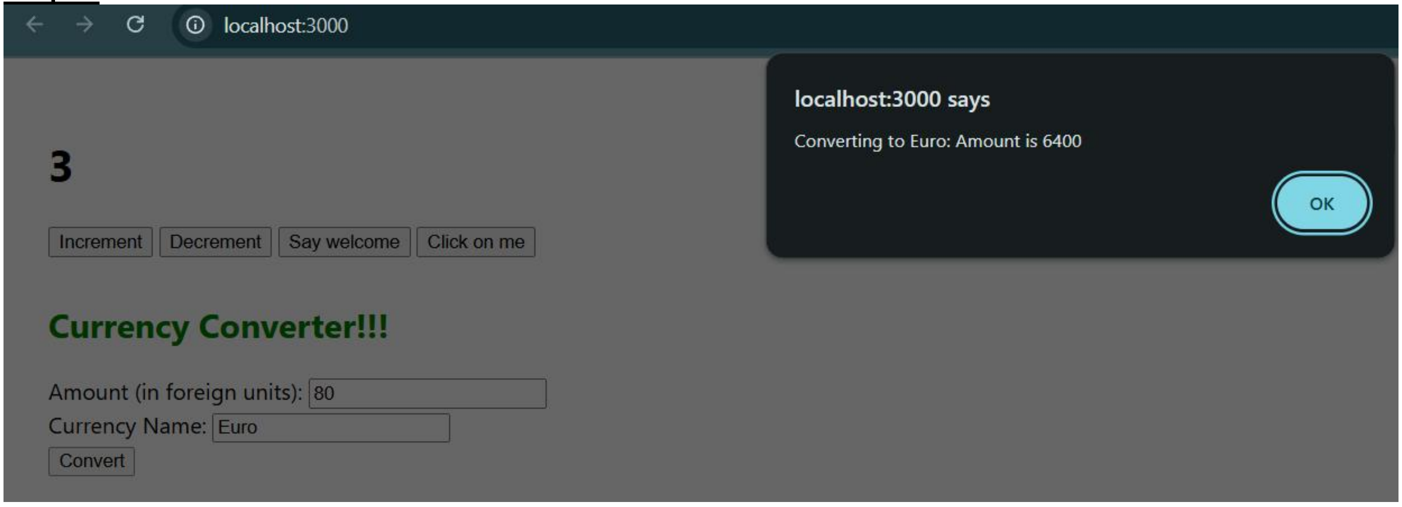
Amount (in foreign units): 20

Currency Name: Euro

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.

Output:


localhost:3000

localhost:3000 says

Converting to Euro: Amount is 6400

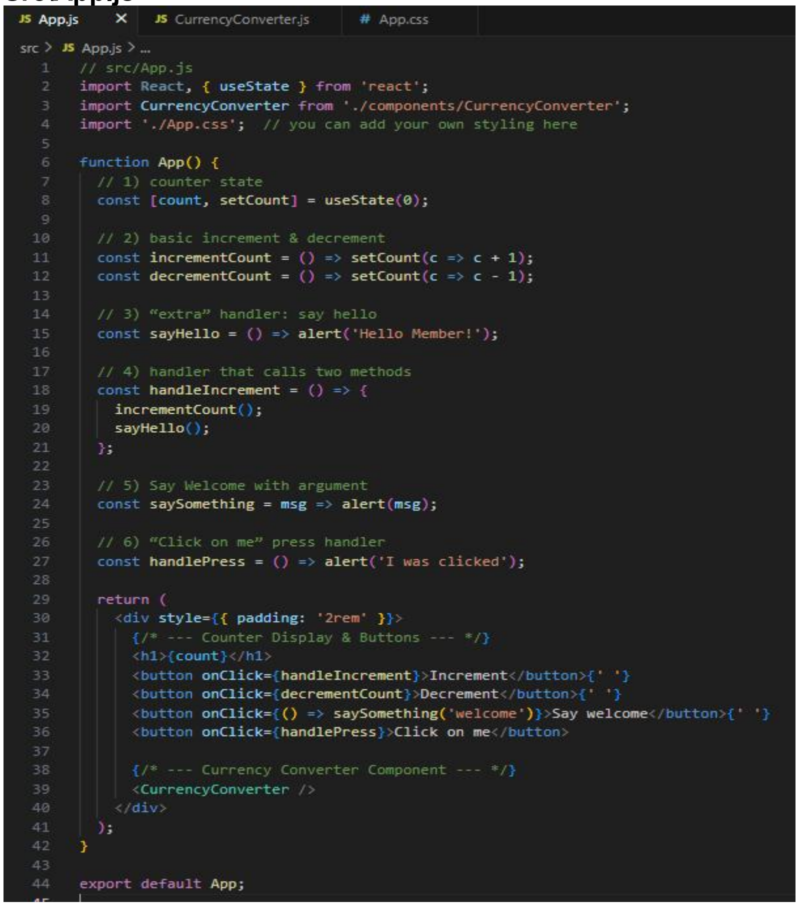
3

Currency Converter!!!

Amount (in foreign units): 80

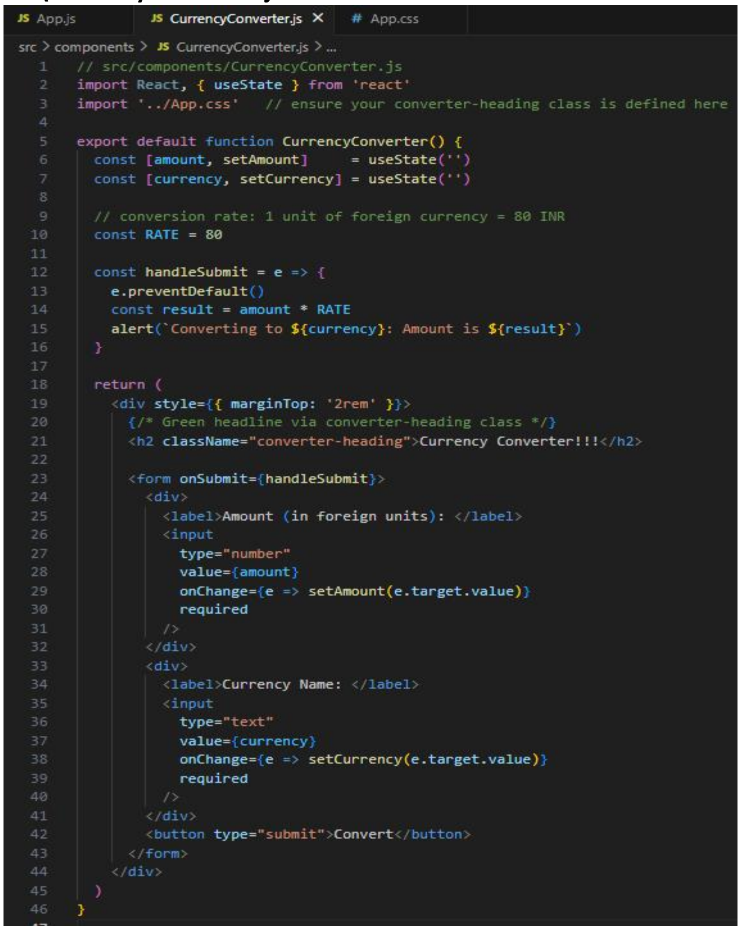
Currency Name: Euro

Src\App.js



```javascript
// src/App.js
import React, { useState } from 'react';
import CurrencyConverter from './components/CurrencyConverter';
import './App.css';  // you can add your own styling here

function App() {
  // 1) counter state
  const [count, setCount] = useState(0);

  // 2) basic increment & decrement
  const incrementCount = () => setCount(c => c + 1);
  const decrementCount = () => setCount(c => c - 1);

  // 3) “extra” handler: say hello
  const sayHello = () => alert('Hello Member!');

  // 4) handler that calls two methods
  const handleIncrement = () => {
    incrementCount();
    sayHello();
  };

  // 5) Say Welcome with argument
  const saySomething = msg => alert(msg);

  // 6) “Click on me” press handler
  const handlePress = () => alert('I was clicked');

  return (
    <div style={{ padding: '2rem' }}>
      {/* --- Counter Display & Buttons --- */}
      <h1>{count}</h1>
      <button onClick={handleIncrement}>Increment</button>{' '}
      <button onClick={decrementCount}>Decrement</button>{' '}
      <button onClick={() => saySomething('welcome')}>Say welcome</button>{' '}
      <button onClick={handlePress}>Click on me</button>

      {/* --- Currency Converter Component --- */}
      <CurrencyConverter />
    </div>
  );
}

export default App;
```

Src\CurrencyConverter.js



```javascript
// src/components/CurrencyConverter.js
import React, { useState } from 'react'
import '../App.css'   // ensure your converter-heading class is defined here

export default function CurrencyConverter() {
  const [amount, setAmount]     = useState('')
  const [currency, setCurrency] = useState('')

  // conversion rate: 1 unit of foreign currency = 80 INR
  const RATE = 80

  const handleSubmit = e => {
    e.preventDefault()
    const result = amount * RATE
    alert(`Converting to ${currency}: Amount is ${result}`)
  }

  return (
    <div style={{ marginTop: '2rem' }}>
      {/* Green headline via converter-heading class */}
      <h2 className="converter-heading">Currency Converter!!!</h2>

      <form onSubmit={handleSubmit}>
        <div>
          <label>Amount (in foreign units): </label>
          <input
            type="number"
            value={amount}
            onChange={e => setAmount(e.target.value)}
            required
          />
        </div>
        <div>
          <label>Currency Name: </label>
          <input
            type="text"
            value={currency}
            onChange={e => setCurrency(e.target.value)}
            required
          />
        </div>
        <button type="submit">Convert</button>
      </form>
    </div>
  )
}
```

REACTJS - HOL12 (LAB4)

THEORIES FOR LAB:

Here's a clear explanation of the three React concepts you've asked for:

✅ Conditional Rendering in React

> **Conditional rendering** means showing or hiding components/elements **based on a condition**, like `if`, `? :`, or logical `&&`.

Common Methods:

Using if/else:

```
function Greeting(props) {
  if (props.isLoggedIn) {
    return <h1>Welcome back!</h1>;
  }
  return <h1>Please sign in.</h1>;
}
```

Using ternary operator:

```
<p>{isAdmin ? "Admin Panel" : "User Dashboard"}</p>
```

Using logical AND (`&&`):

```
{notifications.length > 0 && <p>You have {notifications.length} new messages</p>}
```

Using a function that returns JSX conditionally:

```
function renderMessage(user) {
  return user ? <p>Hello, {user.name}</p> : <p>Welcome, guest</p>;
}
```

✅ Define Element Variables

> **Element variables** are used to **store JSX** in a variable, so it can be conditionally rendered later.
>
> This helps keep the JSX cleaner and easier to manage.

Example:

```
function LoginControl(props) {
  let button;

  if (props.isLoggedIn) {
    button = <button onClick={props.handleLogout}>Logout</button>;
  } else {
    button = <button onClick={props.handleLogin}>Login</button>;
  }

  return <div>{button}</div>;
}
```

✅ How to Prevent Components from Rendering

You can **prevent rendering** in React by:

1. Returning **`null`** from a component:

This means **render nothing**. The component will still run logic/lifecycle but won't output anything.

```
function WarningBanner(props) {
  if (!props.showWarning) {
    return null;
  }

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

2. Using conditional rendering:

Only render a component when a condition is met.

```
{isVisible && <MyComponent />}
```

3. Using **`shouldComponentUpdate`** (in class components):

You can stop re-rendering based on certain conditions.

```
shouldComponentUpdate(nextProps, nextState) {
  return nextProps.someValue !== this.props.someValue;
}
```

4. React.memo for function components:

Prevents re-rendering unless props change.

```
const MyComponent = React.memo(function MyComponent(props) {
  return <div>{props.value}</div>;
});
```

✅ Summary:

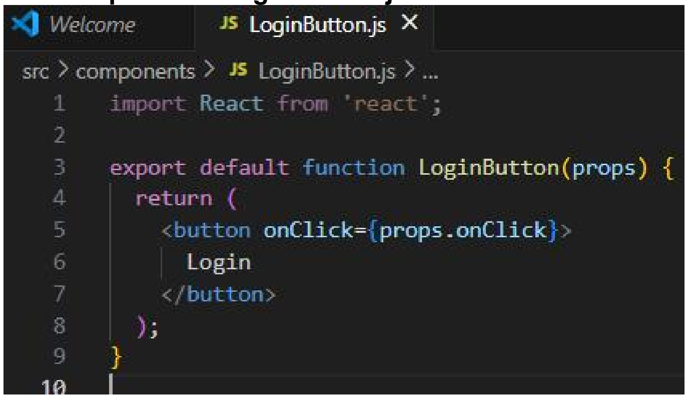
Concept	Purpose	Method
Conditional Rendering	Show/hide elements/components	if, ? :, &&, functions
Element Variables	Store JSX for reuse or conditional use	let button = <JSX />
Prevent Rendering	Skip rendering when not needed	Return null, conditionals, memo, shouldComponentUpdate

MY WORK LAB4:

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.

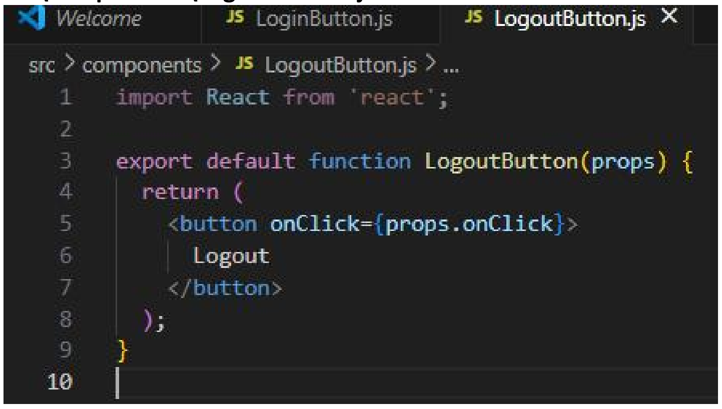
The Login and Logout buttons should accordingly display different pages. Once the user is logged in the User page should be displayed. When the user clicks on Logout, the Guest page should be displayed.

Src\components\LoginButton.js



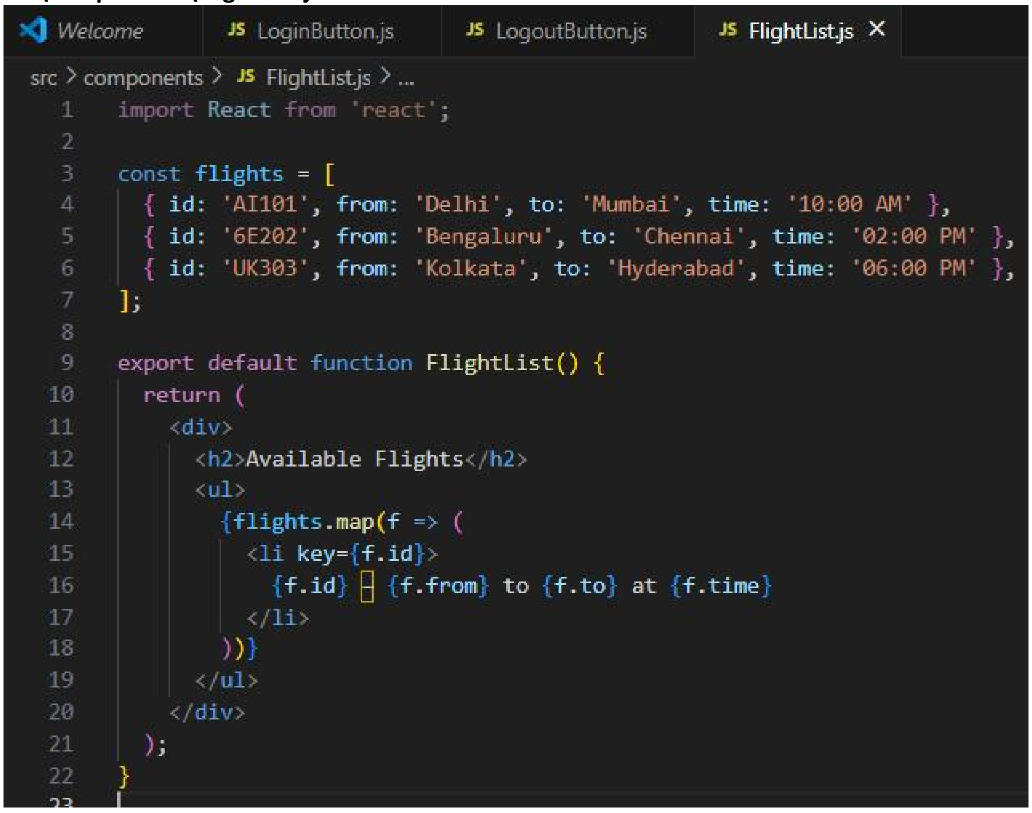
```javascript
import React from 'react';

export default function LoginButton(props) {
  return (
    <button onClick={props.onClick}>
      Login
    </button>
  );
}
```

Src\components\LogoutButton.js



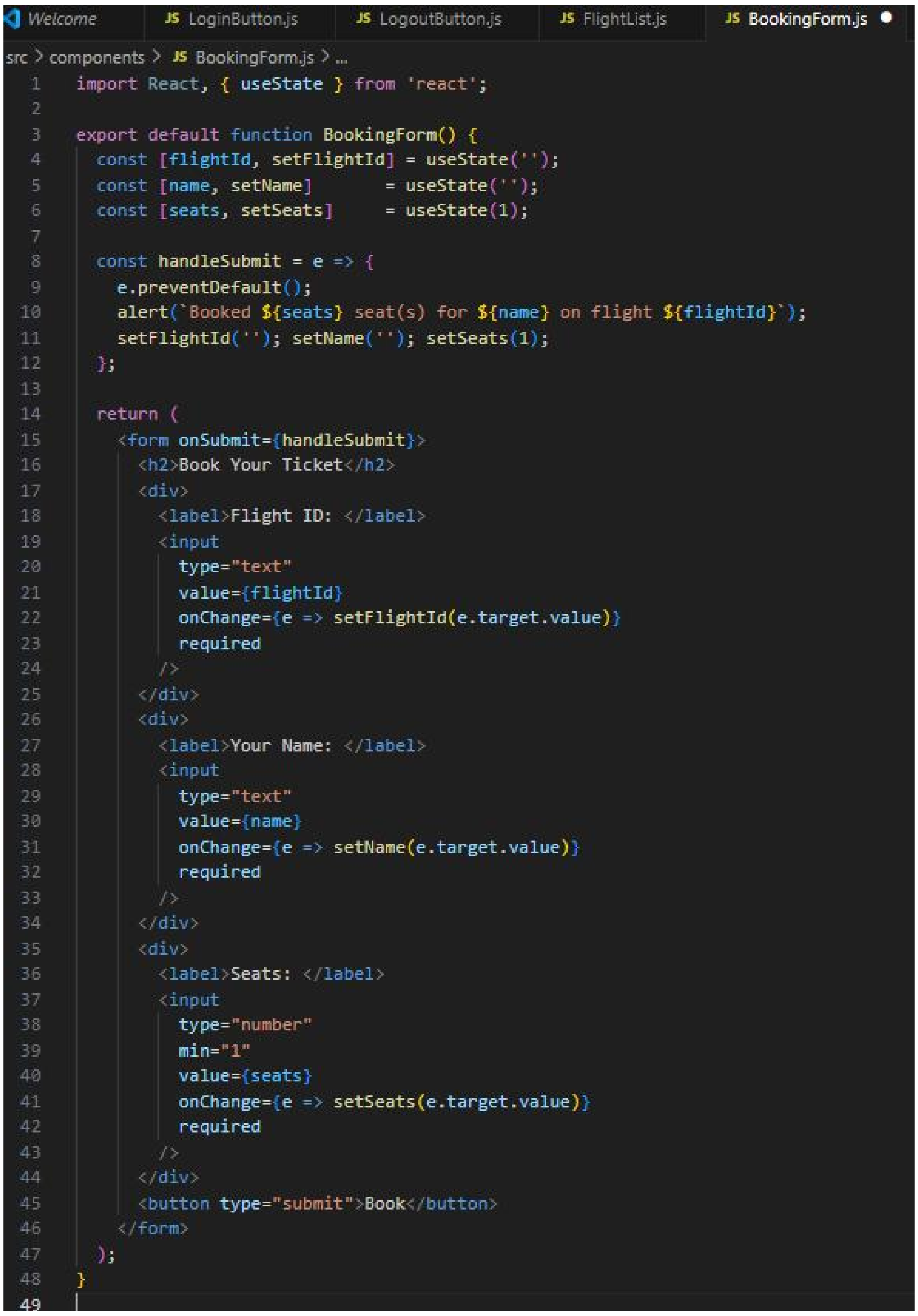
```javascript
import React from 'react';

export default function LogoutButton(props) {
  return (
    <button onClick={props.onClick}>
      Logout
    </button>
  );
}
```

Src\components\FlightList.js



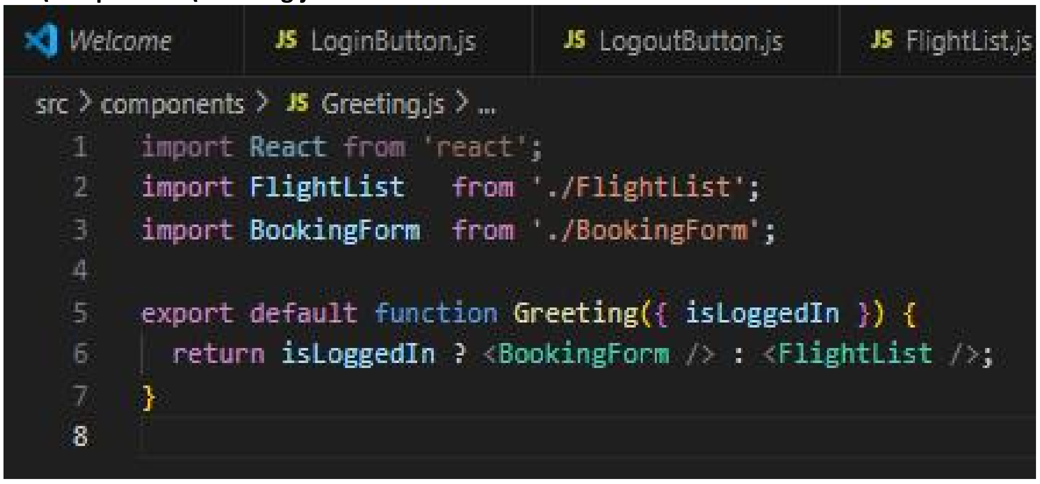
```javascript
import React from 'react';

const flights = [
  { id: 'AI101', from: 'Delhi', to: 'Mumbai', time: '10:00 AM' },
  { id: '6E202', from: 'Bengaluru', to: 'Chennai', time: '02:00 PM' },
  { id: 'UK303', from: 'Kolkata', to: 'Hyderabad', time: '06:00 PM' },
];

export default function FlightList() {
  return (
    <div>
      <h2>Available Flights</h2>
      <ul>
        {flights.map(f => (
          <li key={f.id}>
            {f.id} ▯ {f.from} to {f.to} at {f.time}
          </li>
        ))}
      </ul>
    </div>
  );
}
```

Src\components\BookingForm.js



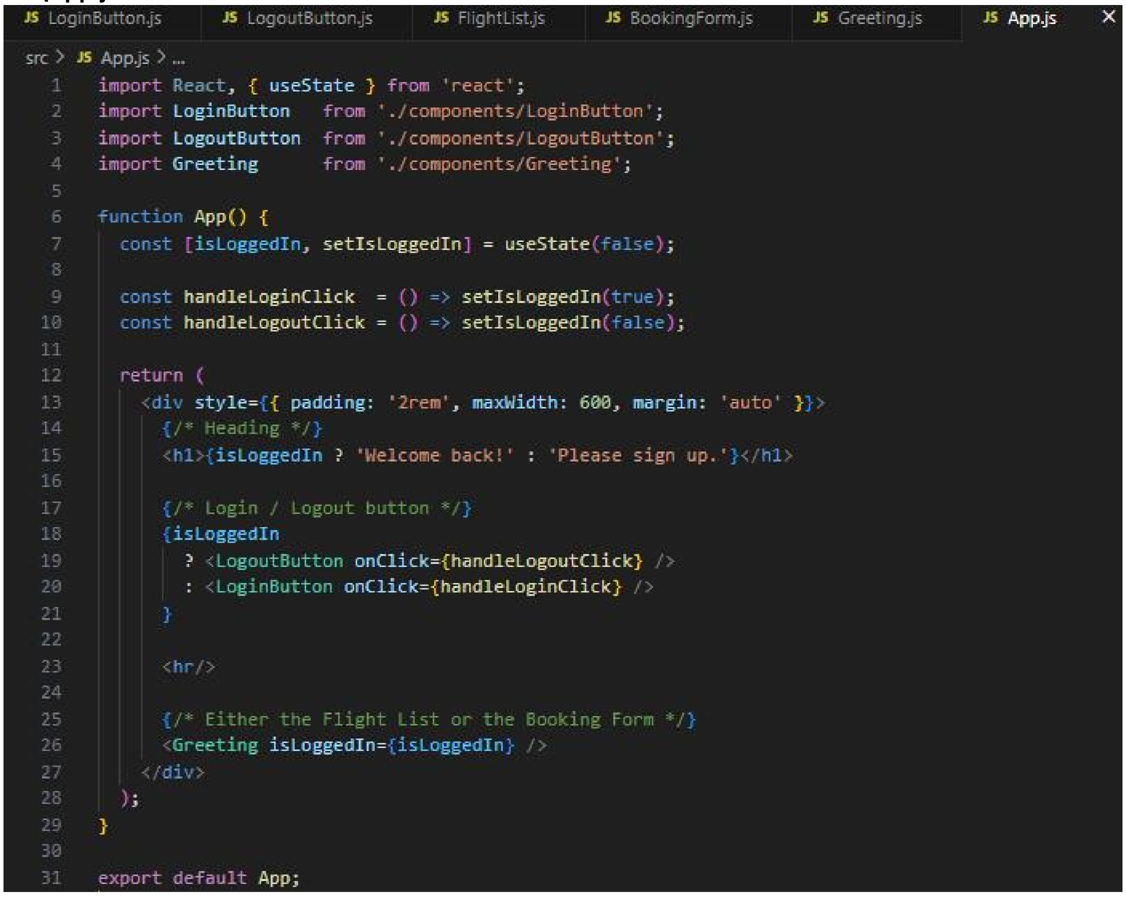
```js
import React, { useState } from 'react';

export default function BookingForm() {
  const [flightId, setFlightId] = useState('');
  const [name, setName]       = useState('');
  const [seats, setSeats]     = useState(1);

  const handleSubmit = e => {
    e.preventDefault();
    alert(`Booked ${seats} seat(s) for ${name} on flight ${flightId}`);
    setFlightId(''); setName(''); setSeats(1);
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Book Your Ticket</h2>
      <div>
        <label>Flight ID: </label>
        <input
          type="text"
          value={flightId}
          onChange={e => setFlightId(e.target.value)}
          required
        />
      </div>
      <div>
        <label>Your Name: </label>
        <input
          type="text"
          value={name}
          onChange={e => setName(e.target.value)}
          required
        />
      </div>
      <div>
        <label>Seats: </label>
        <input
          type="number"
          min="1"
          value={seats}
          onChange={e => setSeats(e.target.value)}
          required
        />
      </div>
      <button type="submit">Book</button>
    </form>
  );
}

```

Src\components\Greeting.js



```js
import React from 'react';
import FlightList   from './FlightList';
import BookingForm  from './BookingForm';

export default function Greeting({ isLoggedIn }) {
  return isLoggedIn ? <BookingForm /> : <FlightList />;
}

```

Src\App.js

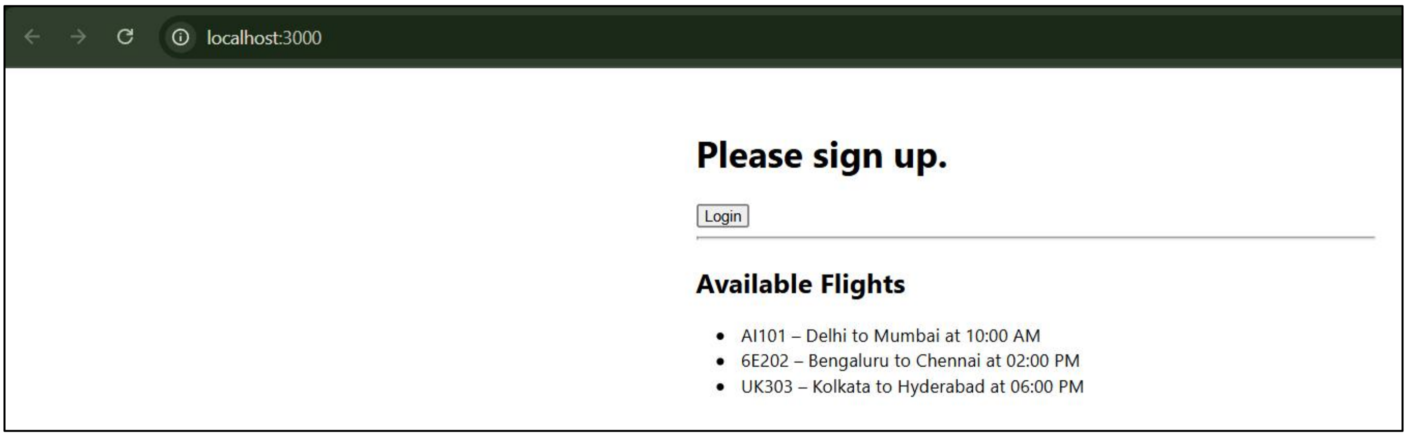


LoginButton.js LogoutButton.js FlightList.js BookingForm.js Greeting.js App.js

src > App.js > ...

```
import React, { useState } from 'react';
import LoginButton   from './components/LoginButton';
import LogoutButton  from './components/LogoutButton';
import Greeting      from './components/Greeting';

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

  const handleLoginClick  = () => setIsLoggedIn(true);
  const handleLogoutClick = () => setIsLoggedIn(false);

  return (
    <div style={{ padding: '2rem', maxWidth: 600, margin: 'auto' }}>
      {/* Heading */}
      <h1>{isLoggedIn ? 'Welcome back!' : 'Please sign up.'}</h1>

      {/* Login / Logout button */}
      {isLoggedIn
        ? <LogoutButton onClick={handleLogoutClick} />
        : <LoginButton onClick={handleLoginClick} />
      }

      <hr/>

      {/* Either the Flight List or the Booking Form */}
      <Greeting isLoggedIn={isLoggedIn} />
    </div>
  );
}

export default App;
```

OUTPUT:



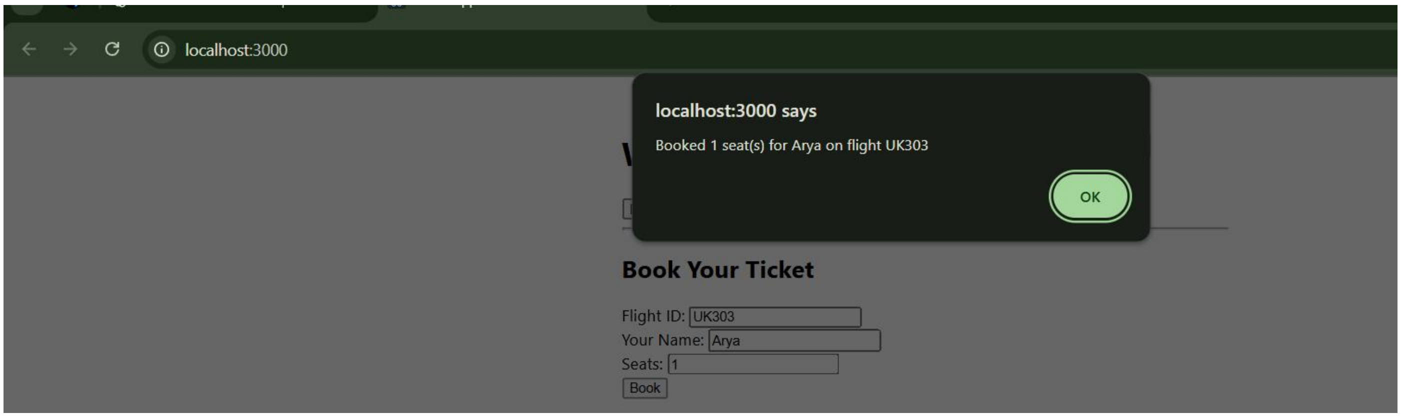
localhost:3000

Please sign up.

Login

Available Flights

- AI101 – Delhi to Mumbai at 10:00 AM
- 6E202 – Bengaluru to Chennai at 02:00 PM
- UK303 – Kolkata to Hyderabad at 06:00 PM



localhost:3000

localhost:3000 says

Booked 1 seat(s) for Arya on flight UK303

OK

Book Your Ticket

Flight ID: UK303
Your Name: Arya
Seats: 1
Book

REACTJS-HOL-13 (LAB 5)

THEORIES FOR LAB :

Here's a comprehensive guide to your React questions on **conditional rendering**, **lists**, **keys**, and **map()**:

✅ Various Ways of Conditional Rendering in React

Using `if/else` **statement**:

```
if (isLoggedIn) {
  return <Dashboard />;
} else {
  return <Login />;
}
```

Using `element variables`:

```
let button;
if (isLoggedIn) {
  button = <LogoutButton />;
} else {
  button = <LoginButton />;
}
return <div>{button}</div>;
```

Using ternary operator:

```
return (
  <div>
    {isLoggedIn ? <Dashboard /> : <Login />}
  </div>
);
```

Using logical AND (`&&`**) operator**:

```
return (
  <div>
    {isAdmin && <AdminPanel />}
  </div>
);
```

Immediately invoked function expressions (IIFE):

```
return (
  <div>
    {(() => {
      if (user.role === 'admin') return <Admin />;
      if (user.role === 'user') return <User />;
      return <Guest />;
    })()}
  </div>
);
```

✅ How to Render Multiple Components

You can render multiple components in a single return using:

Fragment (`<> </>`**)**:

```
return (
  <>
    <Header />
    <Content />
    <Footer />
  </>
);
```

Array of components:

```
return [
  <Header key="1" />,
  <Content key="2" />,
  <Footer key="3" />
];
```

Wrapper `<div>` **(not recommended always)**:

```
return (
  <div>
    <Header />
    <Content />
    <Footer />
  </div>
);
```

✅ Define List Component

A **list component** is a component that displays a collection of items.

It usually takes an array of data and renders each item using `map()`.

Example:

```
function ListComponent({ items }) {
  return (
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}
```

✅ Keys in React Applications

Keys help React **identify which items have changed, are added, or removed**.

Must be **unique** among siblings.

Improve rendering **performance** in lists.

Avoid using index as a key unless there’s no unique ID available.

Good:

```
{users.map(user => <li key={user.id}>{user.name}</li>)}
```

Not ideal (index-based):

```
{items.map((item, index) => <li key={index}>{item}</li>)}
```

✅ How to Extract Components with Keys

When mapping over data, you can extract a **child component** and still pass the `key` to it.

Example:

```
function Item({ value }) {
  return <li>{value}</li>;
}

function ItemList({ items }) {
  return (
    <ul>
      {items.map(item => (
        <Item key={item.id} value={item.name} />
      ))}
    </ul>
  );
}
```

> The key is always passed to the **outermost element** returned in the map — **not** as a prop to the child component.

✅ React `map()` Function

> `map()` is a JavaScript array method used in React to **transform arrays into lists of elements**.
>
> Used heavily in rendering dynamic lists of components.

Example:

```
const names = ['Arya', 'John', 'Jane'];

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

return <ul>{listItems}</ul>;
```

✅ Summary Table

Concept	Description
Conditional Rendering	Display components conditionally (if, ? :, &&)
Render Multiple Components	Use Fragment, arrays, or divs
List Component	Component rendering a list using map()
Keys	Unique identifier for list elements
Extract Component with	Pass key to JSX tag when using extracted

Concept	Description
Keys	components
map() Function	Transforms arrays into React elements

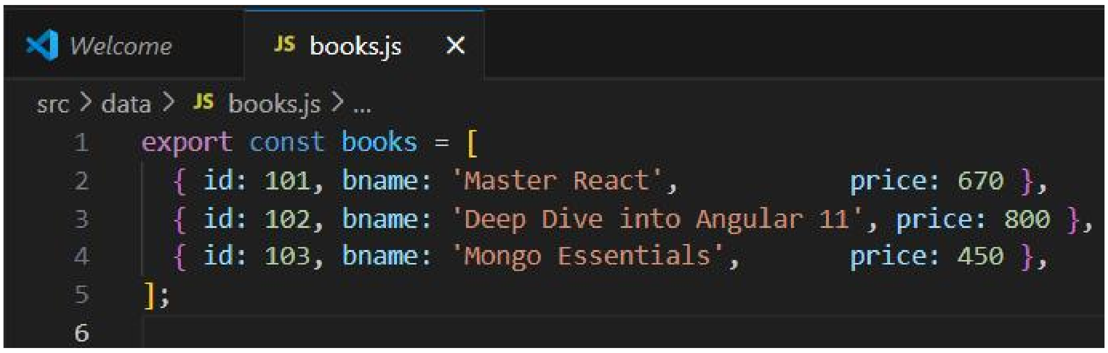
MY WORK LAB5:

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

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

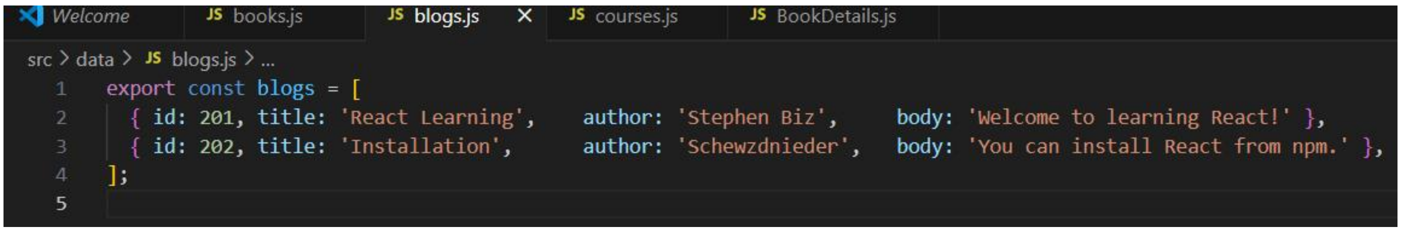
A data Folder was created within which books , blogs and courses files would be there

Src\data\books.js



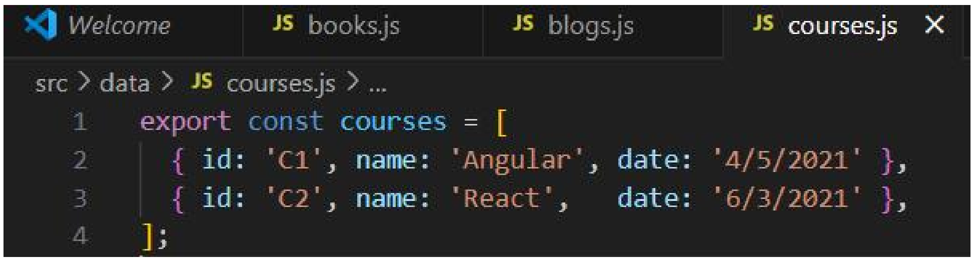
```
src > data > JS books.js > ...
export const books = [
  { id: 101, bname: 'Master React',           price: 670 },
  { id: 102, bname: 'Deep Dive into Angular 11', price: 800 },
  { id: 103, bname: 'Mongo Essentials',       price: 450 },
];
```

Src\data\blogs.js



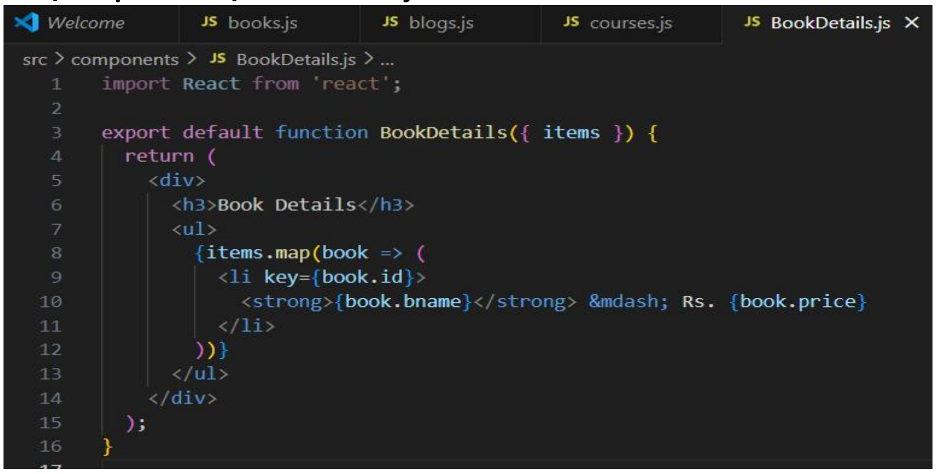
```
src > data > JS blogs.js > ...
export const blogs = [
  { id: 201, title: 'React Learning',    author: 'Stephen Biz',    body: 'Welcome to learning React!' },
  { id: 202, title: 'Installation',      author: 'Schewzdnieder',  body: 'You can install React from npm.' },
];
```

Src\data\courses.js



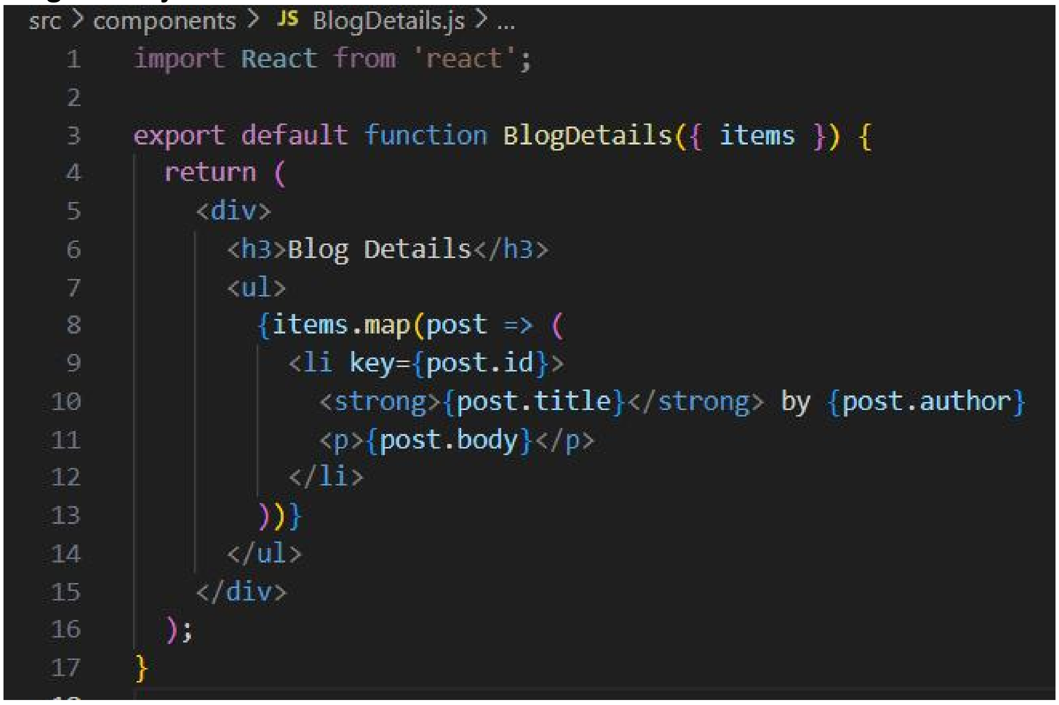
```
src > data > JS courses.js > ...
export const courses = [
  { id: 'C1', name: 'Angular', date: '4/5/2021' },
  { id: 'C2', name: 'React',   date: '6/3/2021' },
];
```

Src\components\BookDetails.js



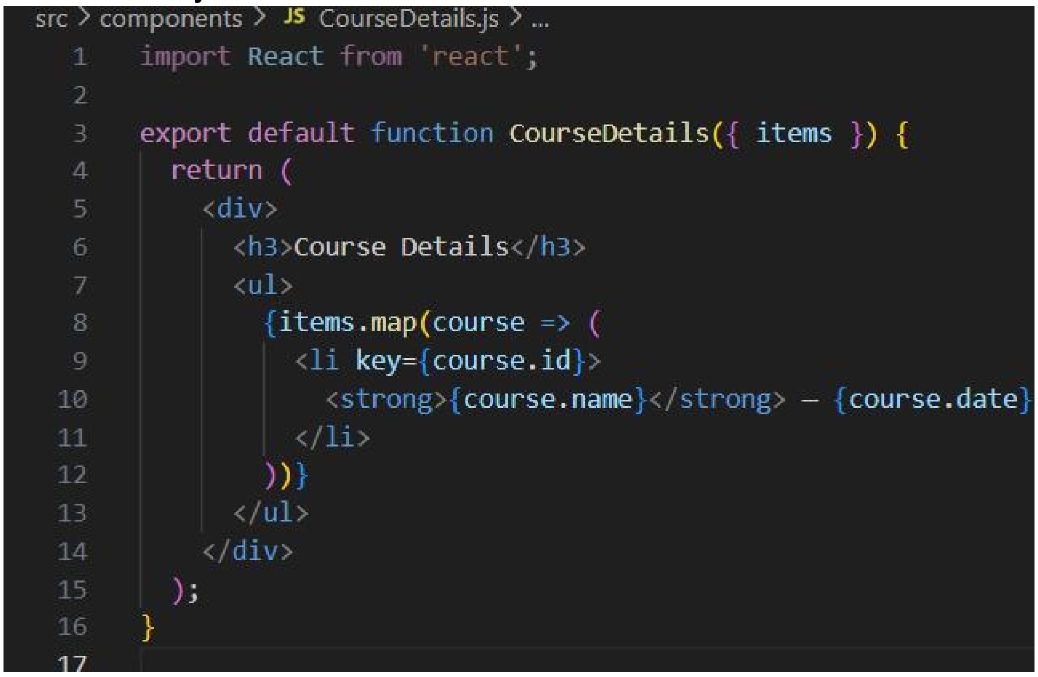
```
import React from 'react';

export default function BookDetails({ items }) {
  return (
    <div>
      <h3>Book Details</h3>
      <ul>
        {items.map(book => (
          <li key={book.id}>
            <strong>{book.bname}</strong> &mdash; Rs. {book.price}
          </li>
        ))}
      </ul>
    </div>
  );
}
```

BlogDetails.js



```
import React from 'react';

export default function BlogDetails({ items }) {
  return (
    <div>
      <h3>Blog Details</h3>
      <ul>
        {items.map(post => (
          <li key={post.id}>
            <strong>{post.title}</strong> by {post.author}
            <p>{post.body}</p>
          </li>
        ))}
      </ul>
    </div>
  );
}
```

CourseDetails.js

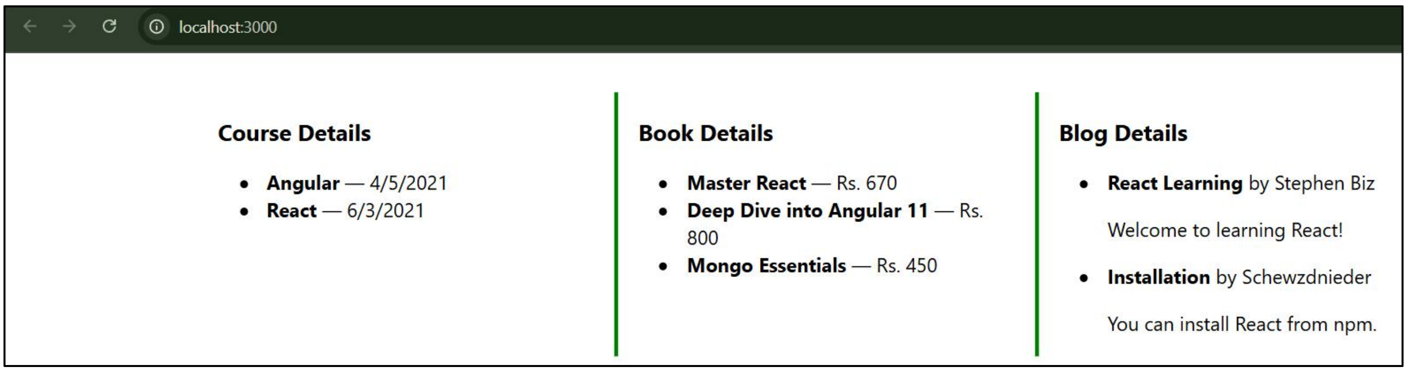


```
import React from 'react';

export default function CourseDetails({ items }) {
  return (
    <div>
      <h3>Course Details</h3>
      <ul>
        {items.map(course => (
          <li key={course.id}>
            <strong>{course.name}</strong> — {course.date}
          </li>
        ))}
      </ul>
    </div>
  );
}
```

App.js

```
// src/App.js
import React from 'react';
import './App.css';

import CourseDetails from './components/CourseDetails';
import BookDetails   from './components/BookDetails';
import BlogDetails   from './components/BlogDetails';
import { courses } from './data/courses';
import { books   } from './data/books';
import { blogs   } from './data/blogs';
function App() {
  return (
    <div className="layout-container">
      {/* Column 1 */}
      <div className="layout-column">
        <CourseDetails items={courses} />
      </div>
      {/* Green separator */}
      <div className="layout-separator" />
      {/* Column 2 */}
      <div className="layout-column">
        <BookDetails items={books} />
      </div>
      {/* Green separator */}
      <div className="layout-separator" />
      {/* Column 3 */}
      <div className="layout-column">
        <BlogDetails items={blogs} />
      </div>
    </div>
  );
}
export default App;
```

OUTPUT:



Course Details

- **Angular** — 4/5/2021
- **React** — 6/3/2021

Book Details

- **Master React** — Rs. 670
- **Deep Dive into Angular 11** — Rs. 800
- **Mongo Essentials** — Rs. 450

Blog Details

- **React Learning** by Stephen Biz

  Welcome to learning React!
- **Installation** by Schewzdnieder

  You can install React from npm.