9. REACTJS – HOL

React Hands-On Lab: cricketapp

1. Introduction

This React hands-on lab demonstrates ES6 JavaScript features such as map(), arrow functions, destructuring, let/const/var, and ES6 classes in a web app named "cricketapp". The app displays and filters cricket players using dynamic components**.**

2. Technologies Used

React.js (via Create React App)

JavaScript (ES6 features)

Node.js and NPM

Visual Studio Code

Chrome browser

3. Project Structure

cricketapp/

├── src/

| ├── App.js

| ├── components/

| | ├── ListofPlayers.js

| | └── IndianPlayers.js

4. Features Implemented

1. ListofPlayers Component:

- Uses ES6 arrow functions and map()

- Displays all players

- Filters players with scores below 70

2. IndianPlayers Component:

- Uses ES6 destructuring and spread operator
- Separates odd/even team players
- Merges two player lists (T20 and Ranji Trophy)

5. Sample Code Snippets

App.js

```
import ListofPlayers from './components/ListofPlayers';
import IndianPlayers from './components/IndianPlayers';

function App() {
  const flag = true;
  return (
    <div>
      {flag ? <ListofPlayers /> : <IndianPlayers />}
    </div>
  );
}

export default App;
```

ListofPlayers.js

```
const players = [
  { name: 'Rohit', score: 80 },
  { name: 'Virat', score: 55 },
  { name: 'Dhoni', score: 68 },
];

const ListofPlayers = () => {
```

```
  return (
    <div>
      <h2>All Players</h2>
      {players.map(p => (
        <p key={p.name}>{p.name} - {p.score}</p>
      ))}

      <h2>Score Below 70</h2>
      {players.filter(p => p.score < 70).map(p => (
        <p key={p.name}>{p.name} - {p.score}</p>
      ))}
    </div>
  );
};

export default ListofPlayers;
```

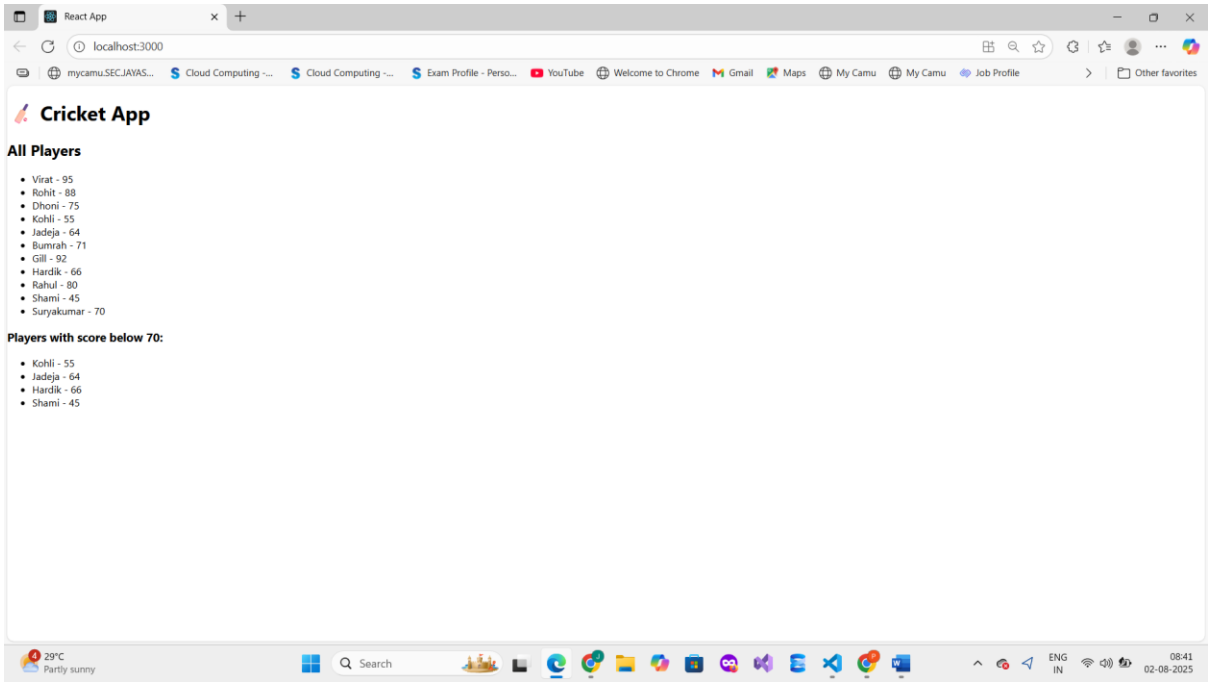
IndianPlayers.js

```
const IndianPlayers = () => {
  const team = ['Rohit', 'Virat', 'Jadeja', 'Bumrah', 'Rahul'];
  const odd = team.filter((_, i) => i % 2 !== 0);
  const even = team.filter((_, i) => i % 2 === 0);

  const T20 = ['Surya', 'Gill'];
  const RanjiTrophy = ['Shaw', 'Jaiswal'];
  const merged = [...T20, ...RanjiTrophy];

  return (
    <div>
```

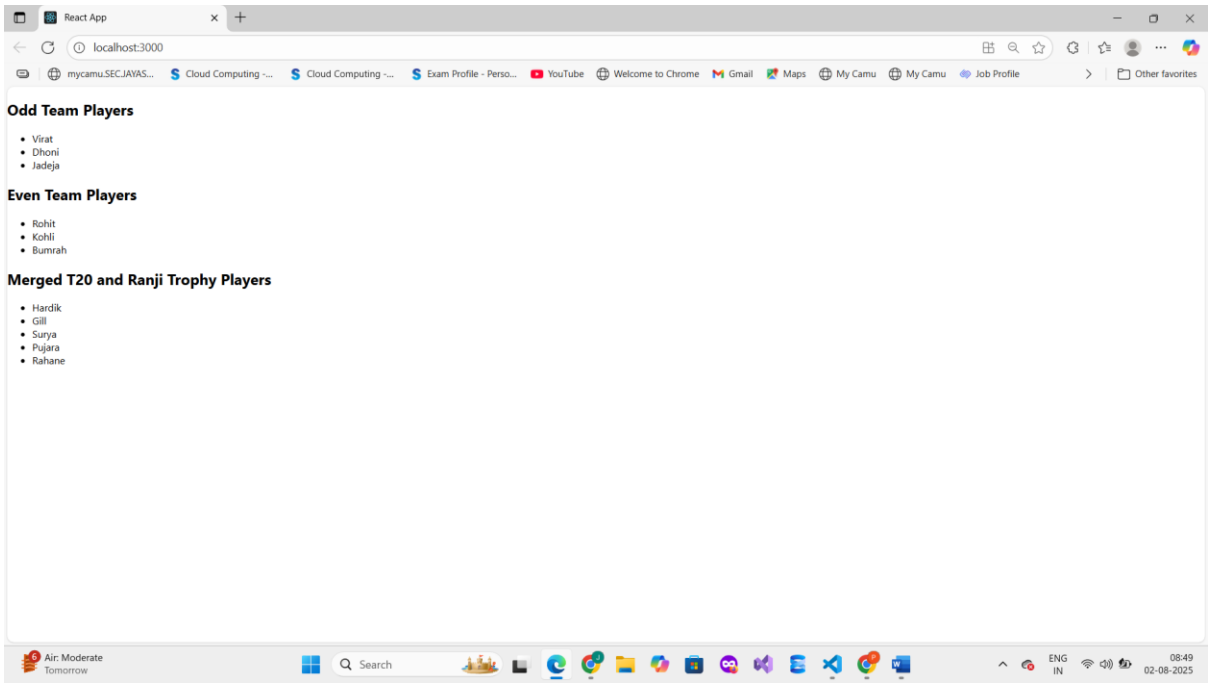
```
      <h2>Odd Team Players</h2>
      {odd.map(name => <p key={name}>{name}</p>)}

      <h2>Even Team Players</h2>
      {even.map(name => <p key={name}>{name}</p>)}

      <h2>Merged Team (T20 + Ranji Trophy)</h2>
      {merged.map(name => <p key={name}>{name}</p>)}
    </div>
  );
};

export default IndianPlayers;
```

6. Output Screenshots

ListofPlayers Component Output (flag = true)



IndianPlayers Component Output (flag = false)



10. REACTJS – HOL

React App - Office Space Rental

Define JSX

JSX (JavaScript XML) is a syntax extension for JavaScript used with React to describe what the UI should look like. It allows writing HTML-like code inside JavaScript files, making it easier to build and visualize UI components.

Example:

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

Explain ECMA Script

ECMAScript (ES) is the standard specification that JavaScript follows. Modern JavaScript uses ES6 and later versions which introduced features like let/const, arrow functions, classes, template literals, destructuring, and more.

Example (Arrow function in ES6):

const greet = () => console.log("Hello!");

Use React.createElement()

This is the standard React API to manually create a React element without JSX. It takes three arguments:

1. **Type of the element (like 'div', 'h1')**
2. **Props (attributes)**
3. **Children (text or other elements)**

Example:

React.createElement('h1', { className: 'title' }, 'Hello from React');

Create React nodes using JSX

React nodes (or elements) can be created using JSX for simpler and readable code.

Example:

const element = <h2>This is a React node</h2>;

Render JSX to DOM

To display JSX on the browser, it must be rendered using ReactDOM.render() (in React 17 or below) or createRoot().render() (React 18+).

Example:

import ReactDOM from 'react-dom/client';

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

root.render(<App />);

Use JavaScript expressions in JSX

You can embed any JavaScript expression inside {} in JSX.

Example:

const user = 'Jayashree';

const greeting = <h1>Hello, {user}!</h1>;

Use inline CSS in JSX

Inline styles are written as JavaScript objects in JSX. The keys are camelCase versions of CSS property names.

Example:

```
const style = { color: 'blue', fontSize: '20px' };

const heading = <h1 style={style}>Styled Text</h1>;
```

Step 1: Create React App

Open terminal and run:

```
npx create-react-app officespacerentalapp
```

Step 2: Open Project in VS Code

```
cd officespacerentalapp

code .
```

Step 3: Modify App.js

```
import React from "react";


const officeSpaces = [
 {
  name: "SkyView Offices",
  rent: 55000,
  address: "123 Anna Salai, Chennai",
  image: "https://via.placeholder.com/300x200?text=SkyView"
 },
 {
  name: "Green Tech Hub",
  rent: 75000,
  address: "456 T Nagar, Chennai",
  image: "https://via.placeholder.com/300x200?text=Green+Tech"
 },
 {
```

```
    name: "Ocean View Tower",
    rent: 60000,
    address: "789 OMR, Chennai",
    image: "https://via.placeholder.com/300x200?text=Ocean+View"
  }
];

function App() {
  return (
    <div style={{ padding: "20px", fontFamily: "Arial" }}>
      <h1 style={{ textAlign: "center", color: "#333" }}>
        Office Space Rental App
      </h1>

      {officeSpaces.map((office, index) => (
        <div
          key={index}
          style={{
            border: "1px solid #ccc",
            padding: "10px",
            marginBottom: "15px",
            borderRadius: "10px",
            boxShadow: "2px 2px 12px #aaa"
          }}
        >
          <img
            src={office.image}
            alt={office.name}
```

```
          style={{ width: "100%", borderRadius: "10px" }}
        />
        <h2>{office.name}</h2>
        <p>
          <strong>Address:</strong> {office.address}
        </p>
        <p style={{ color: office.rent > 60000 ? "green" : "red" }}>
          <strong>Rent:</strong> ₹{office.rent}
        </p>
      </div>
    ))}
  </div>
 );
}

export default App;
```

Step 4: Start the React App

In terminal:

```
npm start
```

Step 5: Output

11. REACTJS – HOL

Title: React Event Handling Lab

1. Explain React Events

React events are actions that a user performs — like clicking a button, typing into a textbox, or submitting a form. React handles these events using JSX syntax similar to

HTML, but with camelCase naming, and uses functions or methods to respond to these actions.

2. Explain Event Handlers

Event handlers are functions that get triggered when a specific event occurs. In React, you attach event handlers directly to elements using attributes like onClick, onChange, etc.

Example:

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

Here, handleClick is the event handler.

3. Define Synthetic Event

A Synthetic Event is a React wrapper around the browser's native event. It normalizes the event object so that it behaves consistently across all browsers, providing better performance and compatibility.

4. Identify React Event Naming Convention

React uses camelCase for event names (unlike lowercase in HTML).
For example:

- **HTML: <button onclick="doSomething()">**
- **React: <button onClick={doSomething}>**

5. Implement Event Handling in React

To handle events in React:

1. **Define an event handler method (function)**
2. **Attach it using the JSX attribute like onClick**

Example:

```
function handleClick() {
  alert('Button clicked!');
}
<button onClick={handleClick}>Click</button>
```

6. Use this Keyword

When using class components, this refers to the component instance. You often bind this in constructors to ensure the handler accesses the correct context.

Example (class component):

this.handleClick = this.handleClick.bind(this);

In functional components, this is generally not used — hooks like useState manage data instead.

7. Use Synthetic Events

Synthetic events are used by default in React. You can access event data like this:

```
function handleInputChange(event) {
  console.log(event.target.value); // Synthetic event object
}
```

Tools/Technologies Used

Tool	Purpose
React	Frontend framework
JSX	JavaScript XML for UI
Node.js + NPM	Package and runtime support
Visual Studio Code	Code editor
JavaScript (ES6)	Event handling and logic

Components and Description

Component Name	Purpose
Counter	Handles increment & decrement events with chained methods
SayWelcome	Demonstrates passing parameters in event handlers
SyntheticEvent	Handles synthetic event (onClick)
CurrencyConverter	Converts INR to Euro using event handlers

Step 1: Create the React App

```
npx create-react-app eventexamplesapp
cd eventexamplesapp
```

code .

Step 2: Clean Up Boilerplate

TYPE CODE :

```
import React from 'react';
import './App.css';
import EventExample from './EventExample';
import CurrencyConvertor from './CurrencyConvertor';

function App() {
  return (
    <div className="App">
      <h1>React Event Handling Lab</h1>
      <EventExample />
      <CurrencyConvertor />
    </div>
  );
}

export default App;
```

Step 3: Create EventExample.js

Code:

```
import React, { Component } from 'react';

class EventExample extends Component {
  constructor(props) {
    super(props);
```

```
  this.state = { count: 0 };
}

increment = () => {
  this.setState({ count: this.state.count + 1 });
};

sayHello = () => {
  alert("Hello! This is a static message.");
};

handleIncrease = () => {
  this.increment();
  this.sayHello();
};

handleDecrement = () => {
  this.setState({ count: this.state.count - 1 });
};

sayWelcome = (msg) => {
  alert(`Welcome: ${msg}`);
};

onPress = () => {
  alert("I was clicked");
};
```

```
  render() {
    return (
      <div>
        <h2>Counter: {this.state.count}</h2>
        <button onClick={this.handleIncrease}>Increment</button>
        <button onClick={this.handleDecrement}>Decrement</button>

        <br /><br />
        <button onClick={() => this.sayWelcome("Welcome")}>Say Welcome</button>

        <br /><br />
        <button onClick={this.onPress}>Synthetic Event (OnPress)</button>
      </div>
    );
  }
}

export default EventExample;
```

Step 4: Create CurrencyConvertor.js

Code:

jsx

CopyEdit

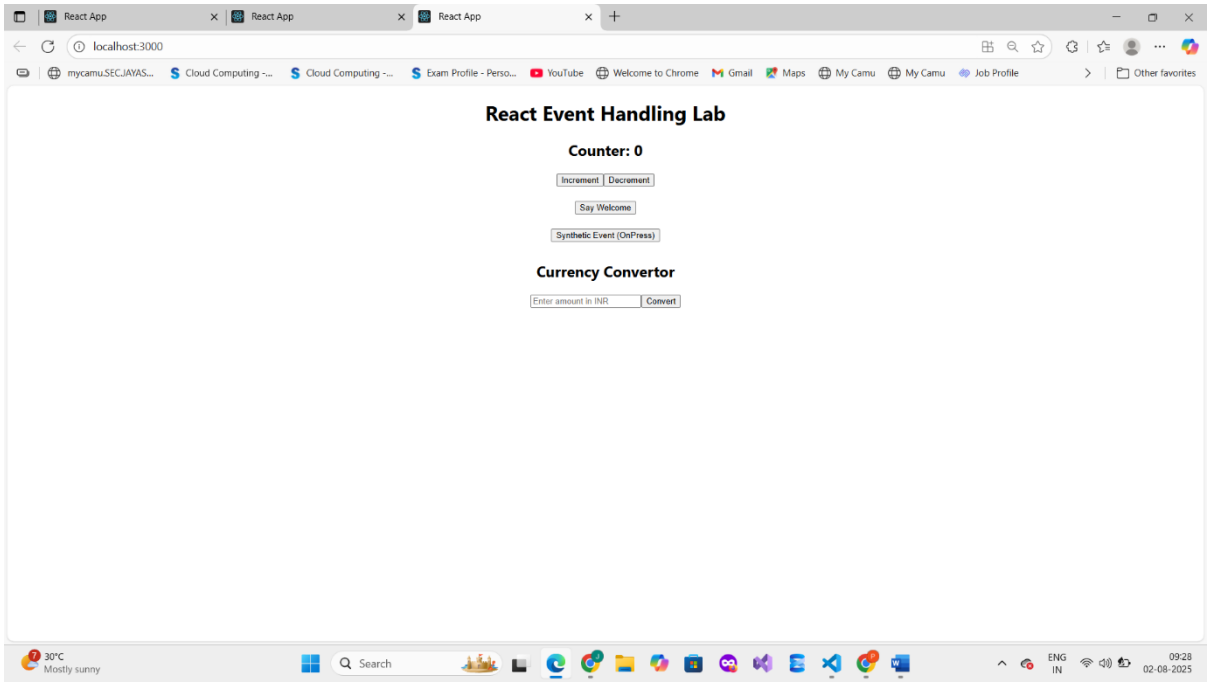
```
import React, { useState } from 'react';

function CurrencyConvertor() {
  const [rupees, setRupees] = useState('');
  const [euro, setEuro] = useState('');
```

```
  const handleSubmit = () => {
    const result = parseFloat(rupees) * 0.011; // Approx conversion
    setEuro(result.toFixed(2));
  };

  return (
    <div style={{ marginTop: '30px' }}>
      <h2>Currency Convertor</h2>
      <input
        type="number"
        placeholder="Enter amount in INR"
        value={rupees}
        onChange={(e) => setRupees(e.target.value)}
      />
      <button onClick={handleSubmit}>Convert</button>
      {euro && <p>EUR: €{euro}</p>}
    </div>
  );
}

export default CurrencyConvertor;
```

Step 5: Run the App

```
npm start
```

Step 6 : Output



12. REACTJS – HOL

React Hands-on Lab – Conditional Rendering (ticketbookingapp)

OBJECTIVES

Explain about Conditional Rendering in React

Conditional rendering in React means displaying different UI elements or components based on certain conditions (like user login status, permissions, or data availability). React uses JavaScript logic inside JSX to decide what to render and when.

Example:

{isLoggedIn ? <WelcomeUser /> : <LoginForm />}

Define Element Variables

Element variables are regular JavaScript variables that store JSX elements.
You can create them inside the component function and conditionally assign different elements to them.

Example:

```
let message;

if (isLoggedIn) {

  message = <h1>Welcome back!</h1>;

} else {
```

```
  message = <h1>Please sign in.</h1>;
}
```

Then render using:

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

Explain How to Prevent Components from Rendering

You can prevent a component from rendering by returning null or an empty fragment (<> </>) based on a condition.

Example:

```
function WarningBanner(props) {
  if (!props.showWarning) {
    return null; // Component will not render
  }
  return <div className="warning">Warning!</div>;
}
```

Implement Conditional Rendering in React Applications

Conditional rendering can be done using:

1. **if-else statements**
2. **ternary operator**
3. **logical && operator**

Ternary Example:

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

Logical AND (&&) Example:

```
{unreadMessages.length > 0 && <span>You have messages</span>}
```

Full Example:

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

```
  return (
    <div>
      {isLoggedIn ? <h1>Welcome</h1> : <h1>Please Login</h1>}
      <button onClick={() => setIsLoggedIn(!isLoggedIn)}>
        {isLoggedIn ? 'Logout' : 'Login'}
      </button>
    </div>
  );
}
```

Step 1 : Create app by typing command in terminal

```
npx create-react-app ticketbookingapp
cd ticketbookingapp
npm start
```

Step 2: Paste the Code

Add a heading: Code (App.js) and type this under it:

```
import React, { useState } from 'react';
import './App.css';
function App() {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

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

  const flightDetails = (
    <div>
      <h2>Available Flights</h2>
```

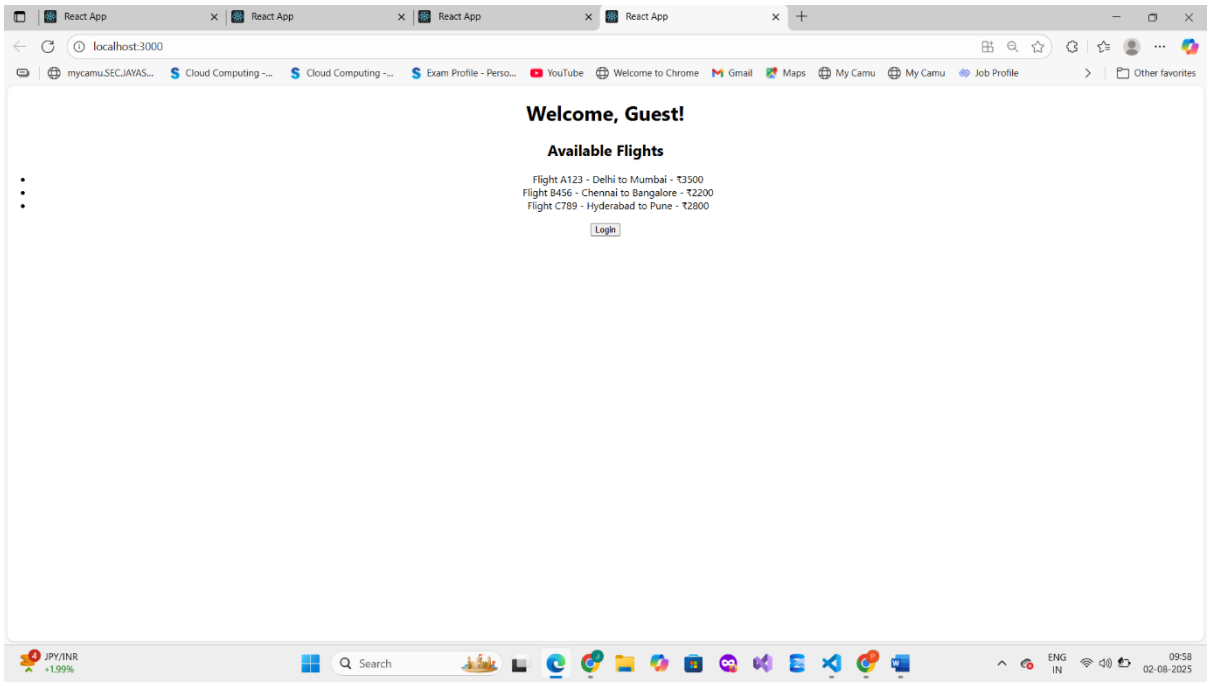
```
    <ul>
      <li>Flight A123 - Delhi to Mumbai - ₹3500</li>
      <li>Flight B456 - Chennai to Bangalore - ₹2200</li>
      <li>Flight C789 - Hyderabad to Pune - ₹2800</li>
    </ul>
  </div>
);

const guestView = (
  <div>
    <h1>Welcome, Guest!</h1>
    {flightDetails}
    <button onClick={handleLogin}>Login</button>
  </div>
);

const userView = (
  <div>
    <h1>Welcome, User!</h1>
    {flightDetails}
    <button>Book Tickets</button>
    <br /><br />
    <button onClick={handleLogout}>Logout</button>
  </div>
);

return (
  <div className="App">
```

```
    {isLoggedIn ? userView : guestView}
   </div>
  );
}
export default App;
```

Step 3: Run command to get output

`npm start`

Step 4 : Output



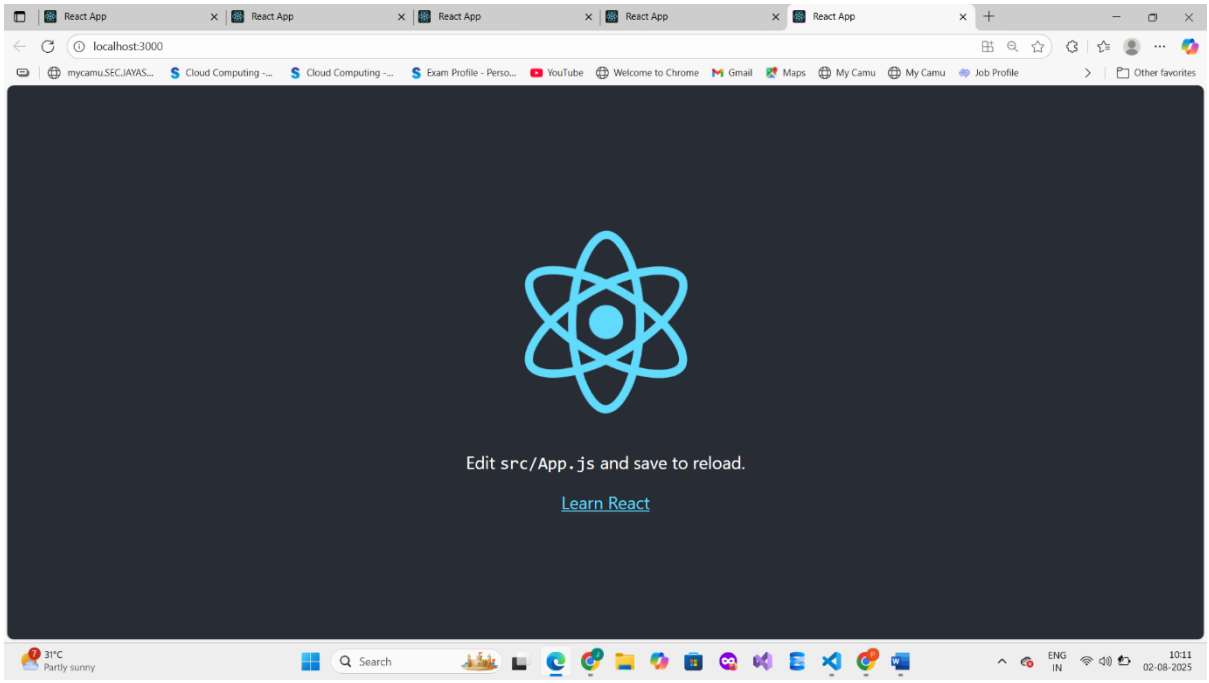
13 . REACTJS – HOL

"Conditional Rendering and List Rendering in React – BloggerApp Project"

Step 1: Create the React App

Type command :

```
npx create-react-app bloggerapp
cd bloggerapp
npm start
```



Step 2: Clean Up Default Code in app.js
replace with new code

```
import React, { useState } from 'react';
import './App.css';
import BookDetails from './BookDetails';
import BlogDetails from './BlogDetails';
import CourseDetails from './CourseDetails';

function App() {
  const [componentToShow, setComponentToShow] = useState('book');

  const renderComponent = () => {
    if (componentToShow === 'book') return <BookDetails />;
    else if (componentToShow === 'blog') return <BlogDetails />;
    else if (componentToShow === 'course') return <CourseDetails />;
    else return null;
  };
```

```
  return (
    <div className="App">
      <h1>Blogger App</h1>
      <button onClick={() => setComponentToShow('book')}>Show Book</button>
      <button onClick={() => setComponentToShow('blog')}>Show Blog</button>
      <button onClick={() => setComponentToShow('course')}>Show Course</button>

      {/* Conditional rendering using function */}
      {renderComponent()}

      {/* Ternary Operator */}
      {componentToShow === 'book' ? <p>You are viewing Book Details</p> : null}

      {/* && Operator */}
      {componentToShow === 'course' && <p>You are viewing Course Details</p>}
    </div>
  );
}

export default App;
```

Step 3: Create Components

src/BookDetails.js

```
import React from 'react';

function BookDetails() {
  const books = ['Atomic Habits', 'Deep Work', 'Clean Code'];
```

```
  return (
    <div>
      <h2>Book Details</h2>
      <ul>
        {books.map((book, index) => (
          <li key={index}> {book}</li>
        ))}
      </ul>
    </div>
  );
}

export default BookDetails;
```

src/BlogDetails.js

```
import React from 'react';
function BlogDetails() {
  const blogs = [
    { id: 1, title: 'React Basics' },
    { id: 2, title: 'Advanced Hooks' },
    { id: 3, title: 'Conditional Rendering' }
  ];

  return (
    <div>
      <h2>Blog Details</h2>
      {blogs.map((blog) => (
        <div key={blog.id}> 📝 {blog.title}</div>
```

```
      ))}
    </div>
  );
}
export default BlogDetails;
```

src/CourseDetails.js

```
import React from 'react';
function CourseDetails() {
  const show = true;
  let message;
  if (!show) return null; // Prevent rendering

  if (show) {
    message = <p>🎓 Course is available for enrollment</p>;
  }

  return (
    <div>
      <h2>Course Details</h2>
      {message}
    </div>
  );
}
export default CourseDetails;
```

Step 4: Run the App

```
npm start
```

Step 5: Output