Week-7

Skill :- React

Hands-on 1: 9.Create a React Application named “cricketapp” with the following components:

Step1:

npx create-react-app cricketapp

cd cricketapp

npm start

Step2:-

//ListofPlayers.js

// src/components/ListofPlayers.js

import React from "react";

const ListofPlayers = () => {

const players = [

{ name: "Rohit", score: 95 },

{ name: "Virat", score: 88 },

{ name: "Rahul", score: 65 },

{ name: "Shreyas", score: 50 },

{ name: "Jadeja", score: 75 },

{ name: "Ashwin", score: 60 },

{ name: "Shami", score: 45 },

{ name: "Bumrah", score: 85 },

{ name: "Gill", score: 92 },

{ name: "Surya", score: 70 },

{ name: "Pant", score: 68 },

];

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

return (

<div>

<h2>All Players</h2>

<ul>

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

<li key={idx}>

{player.name} - {player.score}

</li>

))}

</ul>

<h2>Players with Score Below 70</h2>

<ul>

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

<li key={idx}>

{player.name} - {player.score}

</li>

))}

</ul>

</div>

);

};

export default ListofPlayers;

//IndianPlayers.js

// src/components/IndianPlayers.js

import React from "react";

const IndianPlayers = () => {

const T20players = ["Rohit", "Virat", "Hardik"];

const RanjiTrophy = ["Jaiswal", "Sarfaraz", "Gaikwad"];

// Merge arrays using spread operator

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

// Destructuring example

const [odd1, even1, odd2, even2] = mergedPlayers;

return (

<div>

<h2>Merged Players</h2>

<ul>

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

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

))}

</ul>

<h2>Odd Team Players</h2>

<p>{odd1}, {odd2}</p>

<h2>Even Team Players</h2>

<p>{even1}, {even2}</p>

</div>

);

};

export default IndianPlayers;

//App.js

// src/App.js

import React from "react";

import ListofPlayers from "./components/ListofPlayers";

import IndianPlayers from "./components/IndianPlayers";

function App() {

const flag = true; // Toggle this between true/false to test

return (

<div className="App">

<h1>Cricket App</h1>

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

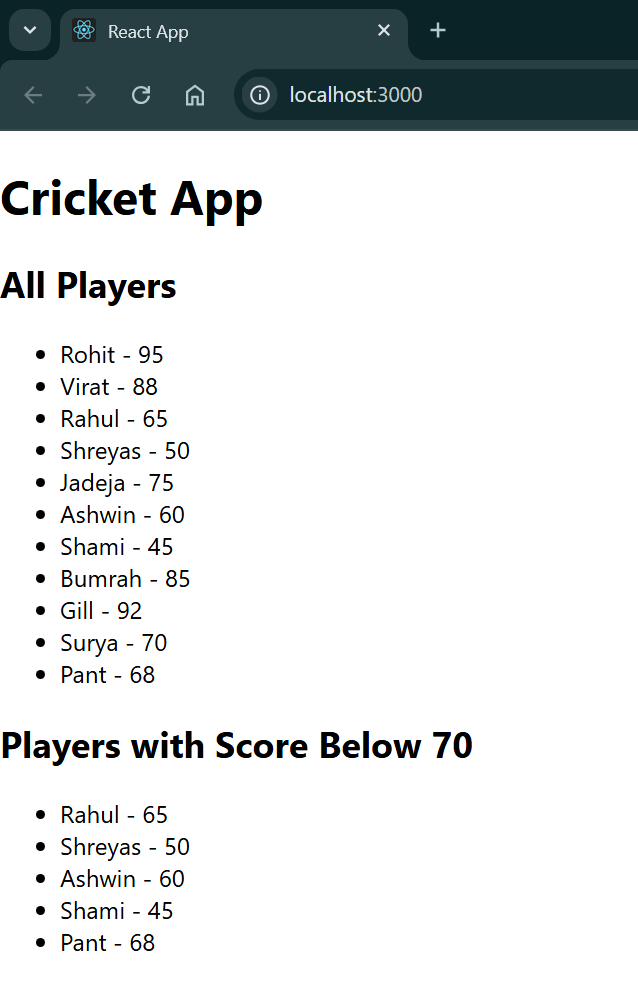
</div>

);

}

export default App;

Output:-



Handson 2:- 10.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.

Step1:

npx create-react-app officespacerentalapp

cd officespacerentalapp

Step2:-

//App.js

import React from 'react';

// Define a single office space object (can be reused later)

const singleOffice = {

name: "TechPark HQ",

rent: 75000,

address: "MG Road, Bangalore",

imageUrl: "https://via.placeholder.com/300x200?text=Office+Space"

};

// Define a list of multiple office spaces

const officeList = [

{

name: "Workspace One",

rent: 50000,

address: "Indiranagar, Bangalore"

},

{

name: "Startup Hub",

rent: 62000,

address: "Koramangala, Bangalore"

},

{

name: "Innovate Place",

rent: 58000,

address: "HSR Layout, Bangalore"

},

{

name: "Elite Towers",

rent: 70000,

address: "Whitefield, Bangalore"

}

];

// Function to apply inline style based on rent

const getRentStyle = (rent) => {

return {

color: rent < 60000 ? 'red' : 'green',

fontWeight: 'bold'

};

};

function App() {

return (

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

{/\* Heading using JSX \*/}

<h1>Office Space Rental App</h1>

{/\* Image using JSX attributes \*/}

<img src={singleOffice.imageUrl} alt="Office" style={{ width: "300px", height: "200px" }} />

{/\* Displaying object using JSX \*/}

<h2>{singleOffice.name}</h2>

<p><strong>Rent:</strong> <span style={getRentStyle(singleOffice.rent)}>{singleOffice.rent}</span></p>

<p><strong>Address:</strong> {singleOffice.address}</p>

<hr />

{/\* Loop through office list \*/}

<h2>Available Office Spaces</h2>

<ul>

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

<li key={index} style={{ marginBottom: "20px", listStyle: "none" }}>

<h3>{office.name}</h3>

<p>

<strong>Rent:</strong>{" "}

<span style={getRentStyle(office.rent)}>{office.rent}</span>

</p>

<p><strong>Address:</strong> {office.address}</p>

</li>

))}

</ul>

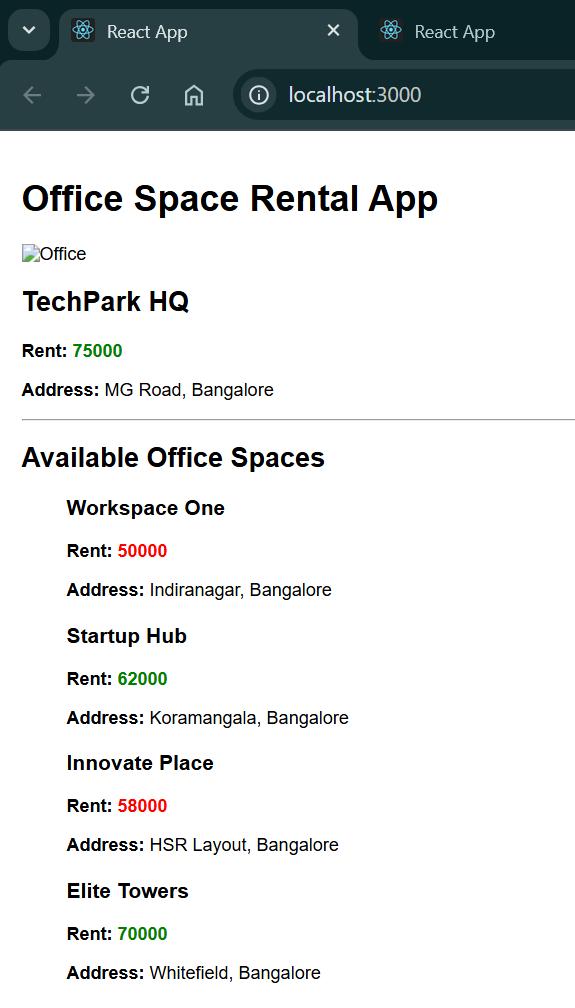
</div>

);

}

export default App;

Output:-



Hands-on 3: 11. Create a React Application “eventexamplesapp” to handle various events of the form elements in HTML.

Step1:

npx create-react-app eventexamplesapp

cd eventexamplesapp

Step2:-

// src/components/Counter.js

import React, { Component } from "react";

class Counter extends Component {

constructor(props) {

super(props);

this.state = {

count: 0,

};

// Bind the traditional function if not using arrow functions

this.increment = this.increment.bind(this);

}

increment() {

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

this.sayHello(); // Invoking another method

}

sayHello() {

alert("Hello! You clicked increment.");

}

decrement = () => {

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

};

sayWelcome = (message) => {

alert(`Message: ${message}`);

};

handleSyntheticEvent = (e) => {

// Synthetic event object

e.preventDefault();

alert("I was clicked");

};

render() {

return (

<div style={{ marginBottom: "40px" }}>

<h2>Counter Example</h2>

<p>Count: {this.state.count}</p>

<button onClick={this.increment}>Increment</button>

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

<br />

<br />

<button onClick={() => this.sayWelcome("Welcome to React!")}>

Say Welcome

</button>

<br />

<br />

<button onClick={this.handleSyntheticEvent}>Synthetic Event</button>

</div>

);

}

}

export default Counter;

// src/components/CurrencyConvertor.js

import React, { useState } from "react";

const CurrencyConvertor = () => {

const [rupees, setRupees] = useState("");

const [euros, setEuros] = useState(null);

const handleChange = (e) => {

setRupees(e.target.value);

};

const handleSubmit = (e) => {

e.preventDefault(); // Synthetic event

const conversionRate = 0.011; // 1 INR = 0.011 EURO approx

setEuros((rupees \* conversionRate).toFixed(2));

};

return (

<div>

<h2>Currency Convertor</h2>

<form onSubmit={handleSubmit}>

<label>Enter amount in INR: </label>

<input type="number" value={rupees} onChange={handleChange} />

<button type="submit">Convert</button>

</form>

{euros && <p>Converted Amount: €{euros}</p>}

</div>

);

};

export default CurrencyConvertor;

// src/App.js

import React from "react";

import Counter from "./components/Counter";

import CurrencyConvertor from "./components/CurrencyConvertor";

function App() {

return (

<div className="App" style={{ padding: "20px", fontFamily: "Arial" }}>

<h1>React Event Handling Examples</h1>

<Counter />

<hr />

<CurrencyConvertor />

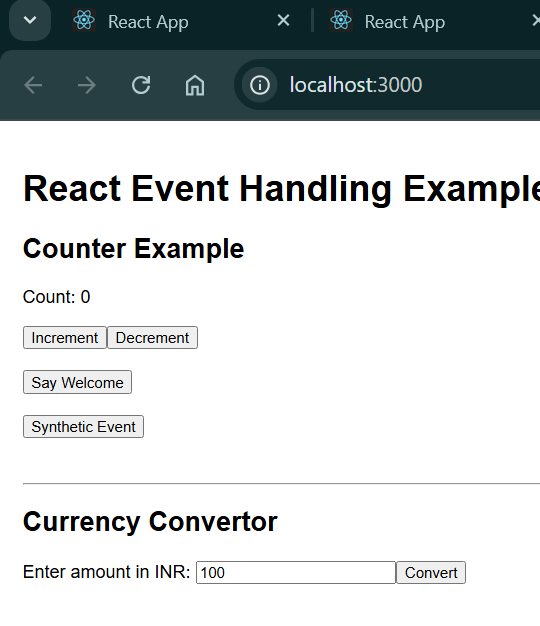
</div>

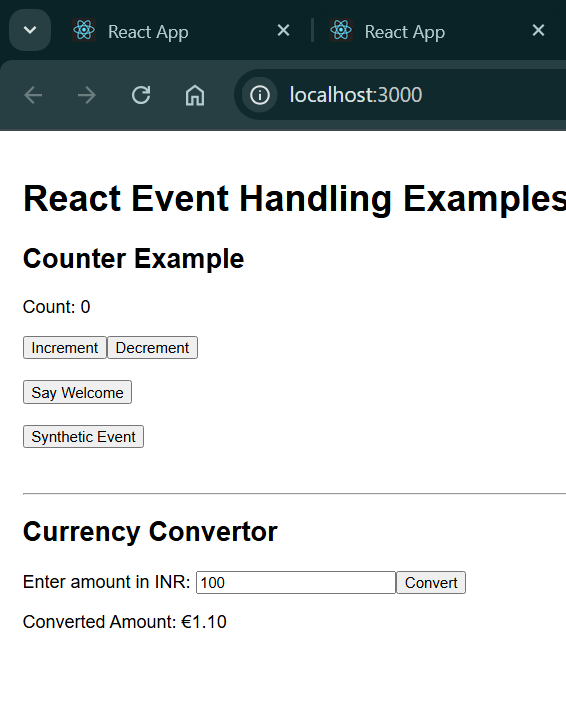
);

}

export default App;

Output:-





## Hands-on 4: ticketbookingapp

Objective: Implement conditional rendering in React based on login state.

App Name: ticketbookingapp

Functional Requirements:

- Guest user sees flight details.  
- Logged-in user can book tickets.  
- Conditional rendering based on login/logout.

### Code Summary:

//App.js

import React, { useState } from 'react';  
import GuestPage from './GuestPage';  
import UserPage from './UserPage';  
  
function App() {  
 const [isLoggedIn, setIsLoggedIn] = useState(false);  
  
 const handleLogin = () => setIsLoggedIn(true);  
 const handleLogout = () => setIsLoggedIn(false);  
  
 return (  
 <div>  
 {isLoggedIn ? <UserPage onLogout={handleLogout} /> : <GuestPage onLogin={handleLogin} />}  
 </div>  
 );  
}  
  
export default App;

//GuestPage.js  
function GuestPage({ onLogin }) {  
 return (  
 <div>  
 <h2>Flight Details</h2>  
 <p>Only browsing available.</p>  
 <button onClick={onLogin}>Login</button>  
 </div>  
 );  
}  
  
export default GuestPage;

UserPage.js

function UserPage({ onLogout }) {  
 return (  
 <div>  
 <h2>Welcome, User!</h2>  
 <p>You can now book your flight.</p>  
 <button onClick={onLogout}>Logout</button>  
 </div>  
 );  
}  
  
export default UserPage;

### Screenshots with Explanation:

1. Guest View – The user is not logged in. They can only view flight details and a Login button.

A screenshot of a computer

AI-generated content may be incorrect.

2. User View – After clicking the Login button, the page shows a booking option and a Logout button.

A screenshot of a computer

AI-generated content may be incorrect.

## Handson-5: bloggerapp

Objective: Use multiple ways of conditional rendering in a React app.

App Name: bloggerapp

Functional Requirements:

- Use components: BookDetails, BlogDetails, CourseDetails  
- Use: ternary, if-else, &&, element variables  
- Use map() with keys

### Code Summary:

//App.js  
import React, { useState } from 'react';  
import BookDetails from './BookDetails';  
import BlogDetails from './BlogDetails';  
import CourseDetails from './CourseDetails';  
  
function App() {  
 const [view, setView] = useState('book');  
  
 let element;  
 if (view === 'book') element = <BookDetails />;  
  
 return (  
 <div>  
 <button onClick={() => setView('book')}>Book</button>  
 <button onClick={() => setView('blog')}>Blog</button>  
 <button onClick={() => setView('course')}>Course</button>  
  
 {element}  
 {view === 'blog' ? <BlogDetails /> : null}  
 {view === 'course' && <CourseDetails />}  
 </div>  
 );  
}  
  
export default App;

//BookDetails.js  
function BookDetails() {  
 const books = ['Atomic Habits', 'React Explained'];  
 return (  
 <div>  
 <h3>Book List</h3>  
 <ul>  
 {books.map((book, index) => <li key={index}>{book}</li>)}  
 </ul>  
 </div>  
 );  
}  
  
export default BookDetails;

//BlogDetails.js  
function BlogDetails() {  
 return <h3>Welcome to Blogs</h3>;  
}  
  
export default BlogDetails;

//CourseDetails.js  
function CourseDetails() {  
 return <h3>ReactJS Course Information</h3>;  
}  
  
export default CourseDetails;

### Screenshots with Explanation:

1. Blog View – After clicking Blog, BlogDetails is rendered using ternary operator.

A screenshot of a computer

AI-generated content may be incorrect.

2. Course View – After clicking Course, CourseDetails is rendered using logical &&.

A screen shot of a computer

AI-generated content may be incorrect.

3. Book View – Default view shows BookDetails using an element variable.

A screenshot of a computer

AI-generated content may be incorrect.