**SupersetID:6383615**

**Rubina S**

**Week 7 ReactJS**

**Exercise 9** **Objectives**

* List the features of ES6
* Explain JavaScript let
* Identify the differences between var and let
* Explain JavaScript const
* Explain ES6 class fundamentals
* Explain ES6 class inheritance
* Define ES6 arrow functions
* Identify set(), map()

# Answers

1. **List the features of ES6**

ES6 introduced modern features like arrow functions, classes, template literals, default parameters, and destructuring.

These features enhance readability, simplify syntax, and improve code maintainability.

# Explain JavaScript let

let is used to declare block-scoped variables and is not hoisted like var.

It prevents variable redeclaration in the same scope, making the code more predictable.

# Identify the differences between var and let

var is function-scoped and can be redeclared; let is block-scoped and cannot. Using let avoids accidental overwriting and hoisting issues found with var.

# Explain JavaScript const

const is used to declare constants that cannot be reassigned after initialization. It’s block-scoped like let and ensures immutability of primitive values.

# Explain ES6 class fundamentals

ES6 classes offer a cleaner way to create object templates with constructors and methods.

They replace prototype-based syntax and bring object-oriented features to JavaScript.

# Explain ES6 class inheritance

ES6 enables inheritance using the extends keyword and super() to access the parent class.

This allows code reuse and supports hierarchical relationships between classes.

# Define ES6 arrow functions

Arrow functions use a compact syntax (=>) and inherit the this value from their enclosing scope.

They are ideal for short functions and callbacks in array methods.

# Identify Set() and Map()

Set is a collection of unique values with no duplicates.

Map is a key-value store that maintains insertion order and supports any type as keys.

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
   * Filter the players with scores below 70 using arrow functions of ES6.
2. IndianPlayers
3. Display the Odd Team Player and Even Team players using the Destructuring features of ES6
4. Declare two arrays T20players and RanjiTrophy players and merge the two arrays and display them using the Merge feature of ES6

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

# Code App.jsx

import React, { useState } from "react";

import ListOfPlayers from "./components/ListofPlayers"; import IndianPlayers from "./components/IndianPlayers"; function App() {

const [flag, setFlag] = useState(true); const toggle = () => setFlag(!flag); return (

<div>

<button onClick={toggle}

style={{

backgroundColor: flag ? "green" : "red", color: "white",

padding: "10px 20px", border: "none", borderRadius: "10px", fontSize: "16px", cursor: "pointer",

transition: "background-color 0.3s ease",

}}

>

CRICKET APP

</button>

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

</div>

);

}

export default App;

# IndianPlayers.jsx

import React from "react";

const indianPlayers = ["Sachin1", "Dhoni2", "Virat3", "Rohit4", "Yuvaraj5", "Raina6"];

const [first, , third, , fifth] = indianPlayers; const oddPlayers = [first, third, fifth];

const evenPlayers = indianPlayers.filter((\_, index) => index % 2 !== 0);

const T20Players = ["Mr. Dravid", "Mr. Sehwag", "Mr. Bhuvneshwar"]; const RanjiPlayers = ["Mr. Harbhajan", "Mr. Zaheer", "Mr. Gambhir"]; const mergedPlayers = [...T20Players, ...RanjiPlayers];

const IndianPlayers = () => { return (

<div>

<h1>Odd Players</h1>

<ul>

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

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

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

</ul>

<h1>Even Players</h1>

<ul>

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

<li key={index}>

{index === 0 ? "Second" : index === 1 ? "Fourth" : "Sixth"} : {player}

</li>

))}

</ul>

<h2>List of Indian Players Merged:</h2>

<ul>

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

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

))}

</ul>

</div>

);

};

export default IndianPlayers;

# ListofPlayers.jsx

import React from "react";

const players = [

{ name: "Mr. Jack", score: 50 },

{ name: "Mr. Michael", score: 70 },

{ name: "Mr. John", score: 40 },

{ name: "Mr. Ann", score: 61 },

{ name: "Mr. Elisabeth", score: 61 },

{ name: "Mr. Sachin", score: 95 },

{ name: "Mr. Dhoni", score: 100 },

{ name: "Mr. Virat", score: 84 },

{ name: "Mr. Jadeja", score: 64 },

{ name: "Mr. Raina", score: 75 },

{ name: "Mr. Rohit", score: 80 },

];

const ListOfPlayers = () => {

const lessThan70 = players.filter(player => player.score < 70); return (

<div>

<h1>List of Players</h1>

<ul>

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

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

))}

</ul>

<h2>List of Players having Scores Less than 70</h2>

<ul>

{lessThan70.map((player, index) => (

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

))}

</ul>

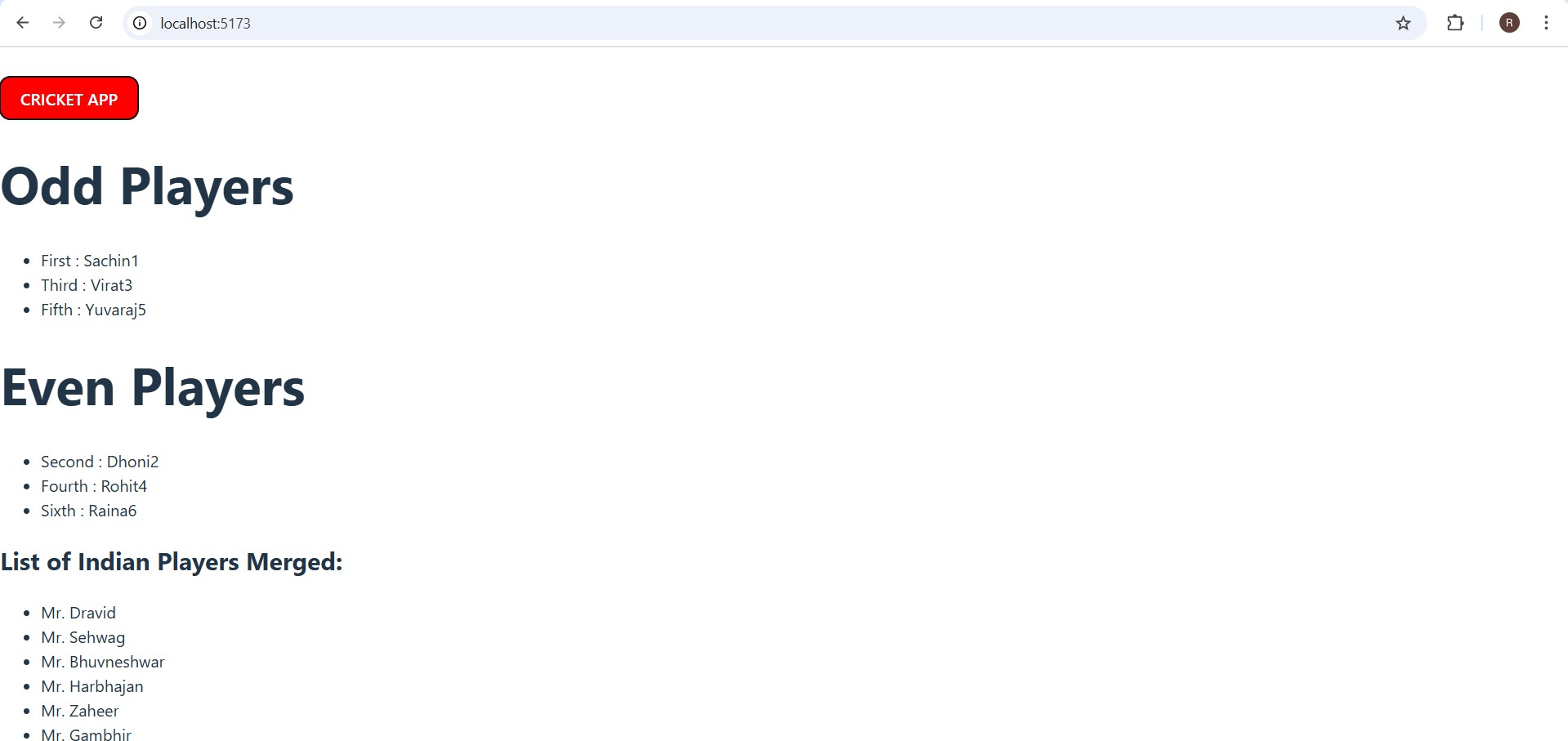
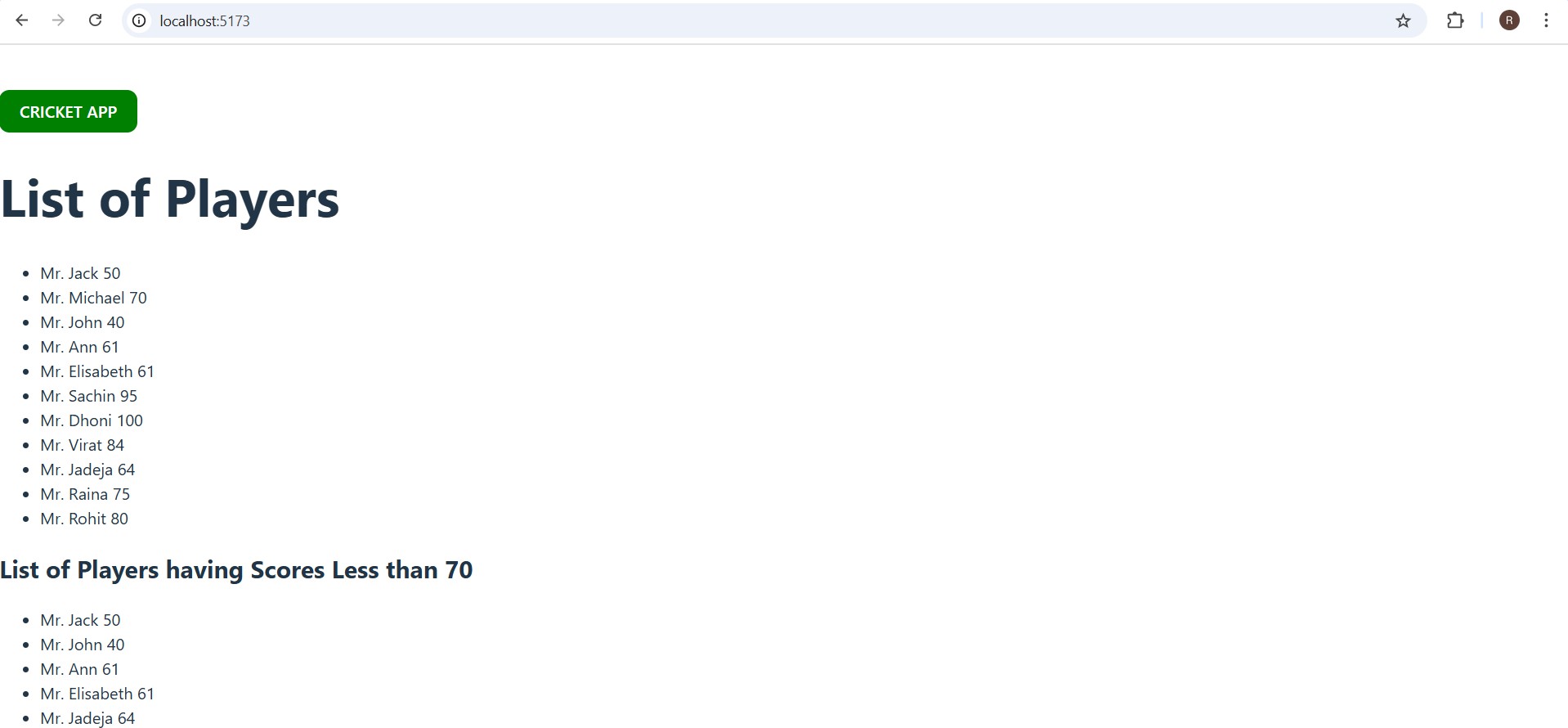
</div>

);

};

export default ListOfPlayers;

**Output**

****

**Exercise 10**

**Objectives**

* Define JSX
* Explain about ECMA Script
* Explain React.createElement()
* Explain how to create React nodes with JSX
* Define how to render JSX to DOM
* Explain how to use JavaScript expressions in JSX
* Explain how to use inline CSS in JSX

# Answers

1. **Define JSX**

JSX (JavaScript XML) is a syntax extension for JavaScript used in React to write HTML-like code inside JavaScript.

It makes the code more readable and allows developers to create React elements using a familiar HTML syntax.

# Explain about ECMA Script

ECMAScript (often abbreviated as ES) is the standardized scripting language specification that JavaScript follows.

It defines language features and updates (like ES6, ES7) to ensure consistency across JavaScript engines.

# Explain React.createElement()

React.createElement() is a React API used to create virtual DOM elements without using JSX.

It takes the element type, props, and children as arguments and returns a React element object.

# Explain how to create React nodes with JSX

React nodes can be created by writing HTML-like tags within JSX syntax. For example, <h1>Hello</h1> creates a node that React converts into React.createElement() calls internally.

# Define how to render JSX to DOM

To render JSX to the DOM, we use ReactDOM.render(jsxElement, document.getElementById('root')).

This renders the virtual DOM element into the actual DOM at the specified HTML container.

# Explain how to use JavaScript expressions in JSX

JavaScript expressions can be embedded inside JSX using curly braces {}. You can include variables, functions, or calculations, like <p>{5 + 10}</p>

# Explain how to use inline CSS in JSX

Inline CSS in JSX is applied using the style attribute with a JavaScript object. For example: <div style={{ color: 'blue', fontSize: '20px' }}>Hello</div>

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.

# Code OfficeSpaces.jsx

import React from 'react'; import './OfficeSpaces.css'; function OfficeSpaces() { const singleOffice = {

name: "Skyline Tower", rent: 55000,

address: "123 Business St, Mumbai",

image: "https://images.unsplash.com/photo-1600585154340-be6161a56a0c"

};

const officeList = [ singleOffice,

{

name: "Tech Park Hub", rent: 62000,

address: "Sector 45, Gurugram",

image: "https://images.unsplash.com/photo-1570129477492-45c003edd2be"

},

{

name: "Urban Edge", rent: 47000,

address: "MG Road, Bengaluru",

image: "https://images.unsplash.com/photo-1504384308090-c894fdcc538d"

},

{

name: "City Heights", rent: 70000,

address: "T Nagar, Chennai",

image: "https://images.unsplash.com/photo-1549924231-f129b911e442"

}

];

return (

<div className="container">

<h1 className="page-heading">🏢 Office Space,at Affordable range</h1>

<div className="office-list">

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

<div className="office-card" key={index}>

<img src={office.image} alt={office.name} className="office-image" />

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

<p>{office.address}</p>

<p

style={{

color: office.rent < 60000 ? 'red' : 'green', fontWeight: 'bold'

}}

>

Rent: ₹{office.rent.toLocaleString()}

</p>

</div>

))}

</div>

</div>

);

}

export default OfficeSpaces;

# OfficeSpaces.css

.container {

padding: 20px;

font-family: 'Segoe UI', sans-serif; background-color: #f2f2f2;

}

.page-heading { text-align: center;

margin-bottom: 30px; color: #333;

}

.office-list { display: flex; flex-wrap: wrap;

justify-content: center; gap: 20px;

}

.office-card { width: 280px;

background-color: white; border-radius: 12px;

box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15); padding: 16px;

text-align: center; transition: 0.3s ease;

}

.office-card:hover { transform: scale(1.03);

}

.office-image { width: 100%; height: 180px; border-radius: 10px; object-fit: cover;

margin-bottom: 12px;

}

# App.jsx

import OfficeSpaces from './components/OfficeSpaces'; function App() {

return (

<div>

<OfficeSpaces />

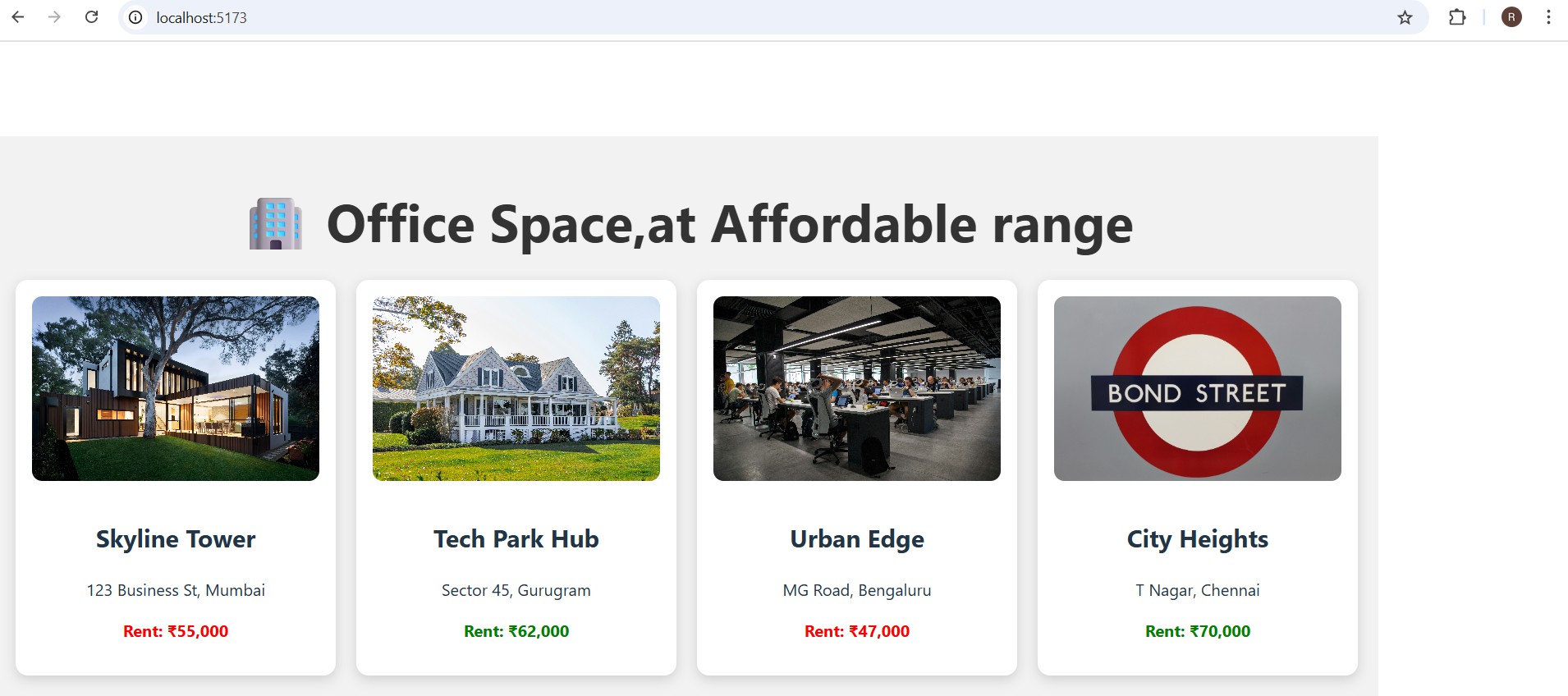
</div>

);

}

export default App;

**Output**

****

**Exercise 11**

**Objectives**

* Explain React events
* Explain about event handlers
* Define Synthetic event
* Identify React event naming convention

# Answers

1. **Explain React events**

React events are triggered by user actions like clicks, typing, or submitting a form. They are similar to native DOM events but work through React’s synthetic event system.

# Explain about event handlers

Event handlers are functions that execute in response to events in React components.

They are assigned to JSX elements using props like onClick, onChange, etc.

# Define Synthetic event

A Synthetic Event is a wrapper around the browser’s native event system created by React.

It ensures consistent event properties across all browsers and platforms.

# Identify React event naming convention

React uses camelCase for event names, such as onClick instead of HTML’s onclick.

Also, event handler values are passed as function references, not as strings.

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.
   1. To increment the value
   2. Say Hello followed by a static message.
2. Create a button “Say Welcome” which invokes the function which takes “welcome” as an argument.
3. 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.

# Code [App.js](http://app.js/)

import React from 'react'; import './App.css';

import CurrencyConvertor from './CurrencyConvertor';

class App extends React.Component { constructor(props) {

super(props); this.state = { count: 0

};

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

}

increment = () => {

this.setState({ count: this.state.count + 1 }); this.sayHello();

alert("Static Message: You clicked Increment!");

};

decrement = () => {

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

};

sayHello = () => {

console.log("Hello from sayHello()");

};

sayWelcome = (message) => { alert(message);

};

handleClick(event) { alert("I was clicked!");

}

render() { return (

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

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

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

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

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

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

<hr />

<CurrencyConvertor />

</div>

);

}

}

export default App;

# App.css

.App {

text-align: center;

}

.App-logo { height: 40vmin;

pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

.App-logo {

animation: App-logo-spin infinite 20s linear;

}

}

.App-header {

background-color: #282c34; min-height: 100vh;

display: flex;

flex-direction: column; align-items: center; justify-content: center;

font-size: calc(10px + 2vmin); color: white;

}

.App-link { color: #61dafb;

}

@keyframes App-logo-spin { from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

# [CurrencyConvertor.js](http://currencyconvertor.js/)

import React, { Component } from 'react';

class CurrencyConvertor extends Component { constructor(props) {

super(props); this.state = { amount: '',

currency: '', result: null

};

}

handleAmountChange = (event) => { this.setState({ amount: event.target.value });

};

handleCurrencyChange = (event) => { this.setState({ currency: event.target.value });

};

handleSubmit = (event) => { event.preventDefault();

const { amount, currency } = this.state; const conversionRate = 80;

const result = amount \* conversionRate;

alert(`Converting to ${currency} Amount is ${result}`); this.setState({ result: `${result} ${currency}` });

};

render() { return (

<div>

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

<form onSubmit={this.handleSubmit}>

<div>

<label>Amount: </label>

<input type="text" value={this.state.amount} onChange={this.handleAmountChange} />

</div>

<div>

<label>Currency: </label>

<input type="text" value={this.state.currency} onChange={this.handleCurrencyChange} />

</div>

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

</form>

{this.state.result && <h4>Converted Amount: {this.state.result}</h4>}

</div>

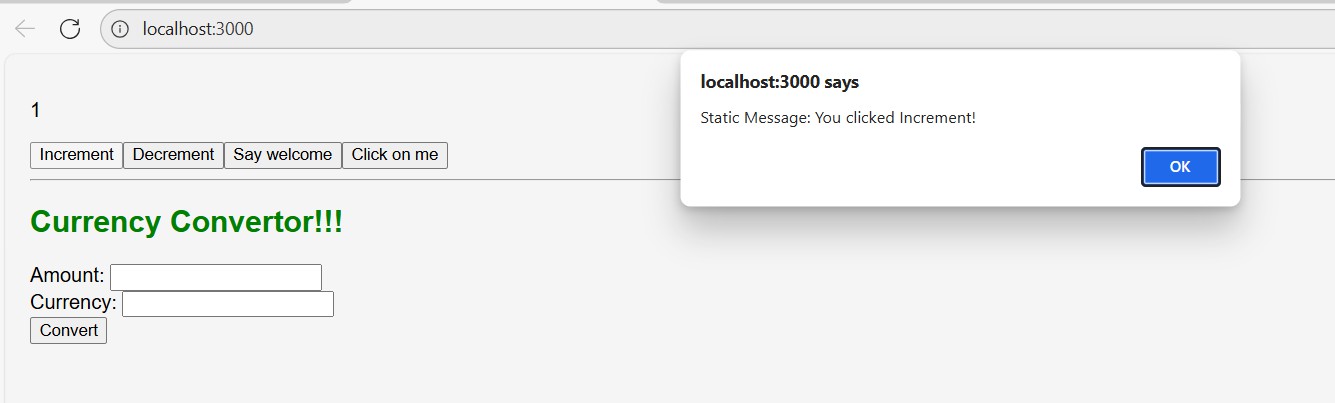
);

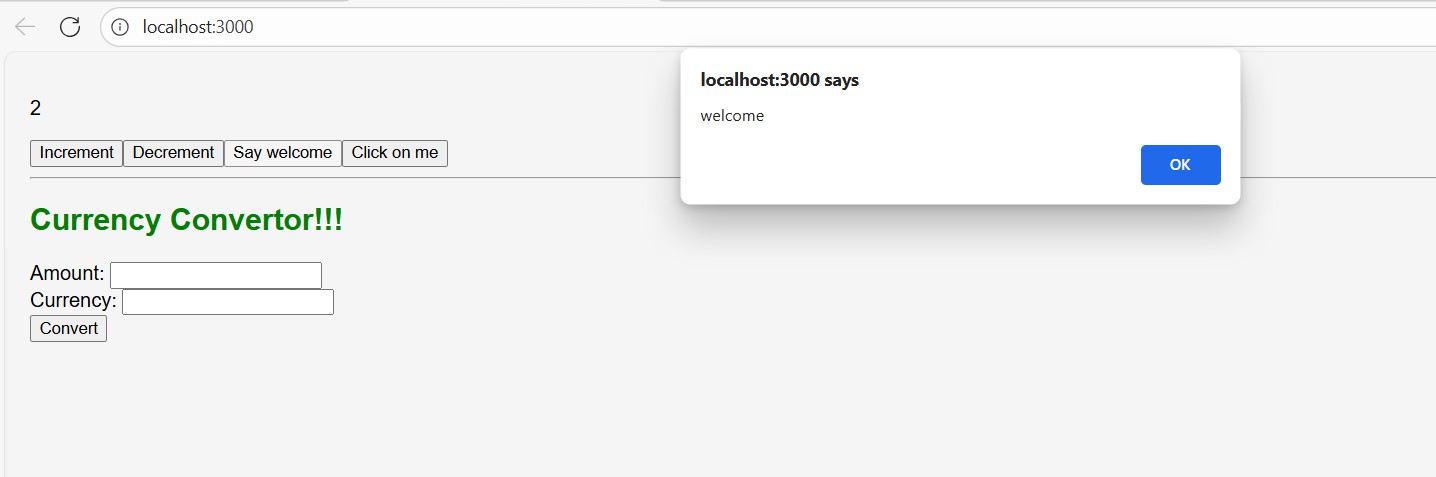
}

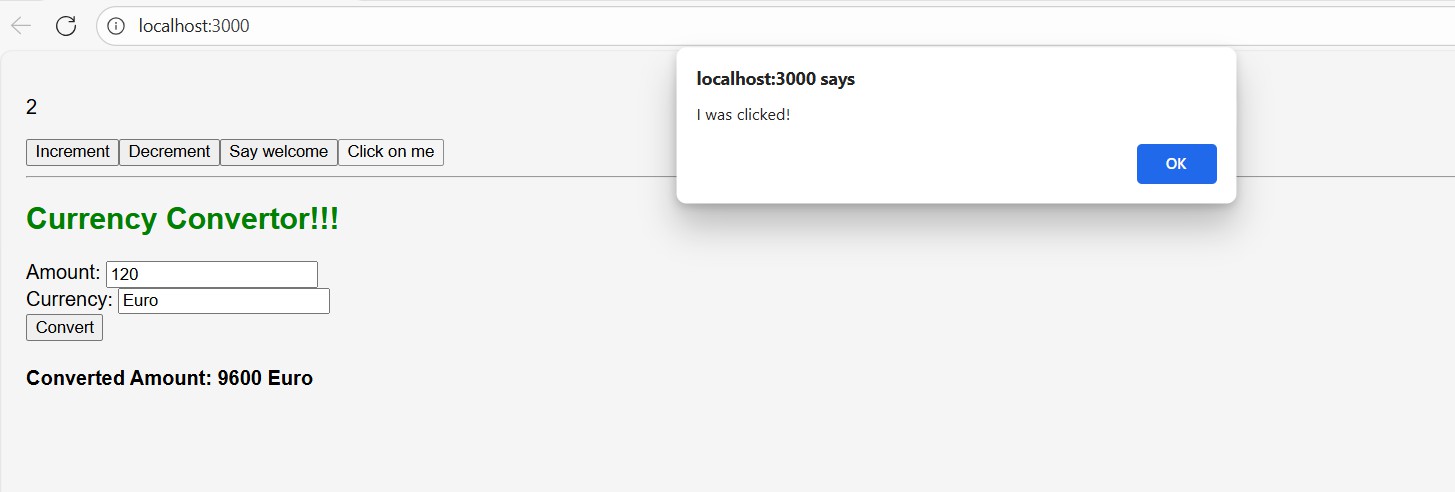
}

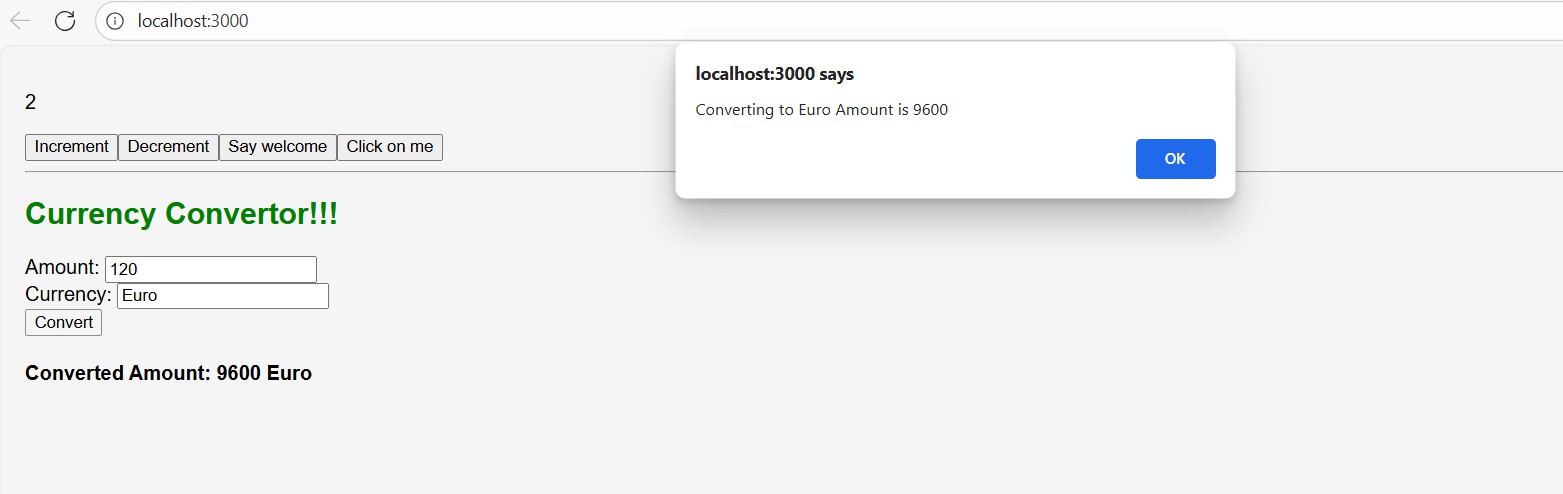
export default CurrencyConvertor;

Output









# Exercise 12 Objectives

* Explain about conditional rendering in React
* Define element variables
* Explain how to prevent components from rendering

# Answers

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

Conditional rendering in React means displaying components or elements based on certain conditions (like if, &&, or ternary operators). It helps show different UI for different states or situations.

# Define element variables

Element variables store React elements in variables, allowing you to conditionally assign and render them later in the JSX. This improves readability and makes conditional UI logic cleaner.

# Explain how to prevent components from rendering

To prevent a component from rendering, you can return null in its render method or use conditional checks in the parent component. This skips rendering while keeping the component logic intact.

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.

# Code App.jsx

import React, { useState } from "react";

import GuestPage from "./components/GuestPage"; import UserPage from "./components/UserPage"; function App() {

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

const handleLogin = () => { setIsLoggedIn(true);

};

const handleLogout = () => { setIsLoggedIn(false);

};

return (

<div style={styles.container}>

<h1>Ticket Booking App</h1>

{isLoggedIn ? (

<>

<button onClick={handleLogout} style={styles.btnLogout}> Logout

</button>

<UserPage />

</>

) : (

<>

<button onClick={handleLogin} style={styles.btnLogin}> Login

</button>

<GuestPage />

</>

)}

</div>

);

}

const styles = { container: { textAlign: "center",

fontFamily: "Arial, sans-serif", margin: "20px",

},

btnLogin: {

padding: "8px 16px", marginBottom: "10px", backgroundColor: "#28a745", color: "#fff",

border: "none", cursor: "pointer",

},

btnLogout: {

padding: "8px 16px",

marginBottom: "10px", backgroundColor: "#dc3545", color: "#fff",

border: "none", cursor: "pointer",

},

};

export default App;

# GuestPage.jsx

import React from "react"; const flightData = [

{ id: 1, airline: "Air India", from: "Delhi", to: "Mumbai", price: 5500 },

{ id: 2, airline: "IndiGo", from: "Bangalore", to: "Chennai", price: 3200 },

{ id: 3, airline: "Vistara", from: "Kolkata", to: "Hyderabad", price: 4800 },

];

function GuestPage() { return (

<div>

<h2>Available Flights</h2>

<table style={styles.table}>

<thead>

<tr>

<th>Airline</th>

<th>From</th>

<th>To</th>

<th>Price (₹)</th>

</tr>

</thead>

<tbody>

{flightData.map((flight) => (

<tr key={flight.id}>

<td>{flight.airline}</td>

<td>{flight.from}</td>

<td>{flight.to}</td>

<td>{flight.price}</td>

</tr>

))}

</tbody>

</table>

<p style={{ color: "red" }}>Login to book tickets</p>

</div>

);

}

const styles = { table: {

width: "80%", margin: "auto",

borderCollapse: "collapse",

},

};

export default GuestPage;

# UserPage.jsx

import React, { useState } from "react";

const flightData = [

{ id: 1, airline: "Air India", from: "Delhi", to: "Mumbai", price: 5500 },

{ id: 2, airline: "IndiGo", from: "Bangalore", to: "Chennai", price: 3200 },

{ id: 3, airline: "Vistara", from: "Kolkata", to: "Hyderabad", price: 4800 },

];

function UserPage() {

const [bookedFlight, setBookedFlight] = useState(null); const handleBook = (flight) => { setBookedFlight(flight);

alert(`Ticket booked for ${flight.airline} from ${flight.from} to ${flight.to}`);

};

return (

<div>

<h2>Book Your Flight</h2>

<table style={styles.table}>

<thead>

<tr>

<th>Airline</th>

<th>From</th>

<th>To</th>

<th>Price (₹)</th>

<th>Action</th>

</tr>

</thead>

<tbody>

{flightData.map((flight) => (

<tr key={flight.id}>

<td>{flight.airline}</td>

<td>{flight.from}</td>

<td>{flight.to}</td>

<td>{flight.price}</td>

<td>

<button style={styles.btnBook} onClick={() => handleBook(flight)}> Book

</button>

</td>

</tr>

))}

</tbody>

</table>

{bookedFlight && (

<div style={styles.confirmation}>

<h3>Booking Confirmation</h3>

<p>

You booked <strong>{bookedFlight.airline}</strong> from{" "}

{bookedFlight.from} to {bookedFlight.to}.

</p>

</div>

)}

</div>

);

}

const styles = { table: {

width: "80%", margin: "auto",

borderCollapse: "collapse",

},

btnBook: {

backgroundColor: "#007bff", color: "#fff",

padding: "5px 10px", border: "none", cursor: "pointer",

},

confirmation: { marginTop: "20px",

padding: "10px", backgroundColor: "#d4edda", color: "#155724",

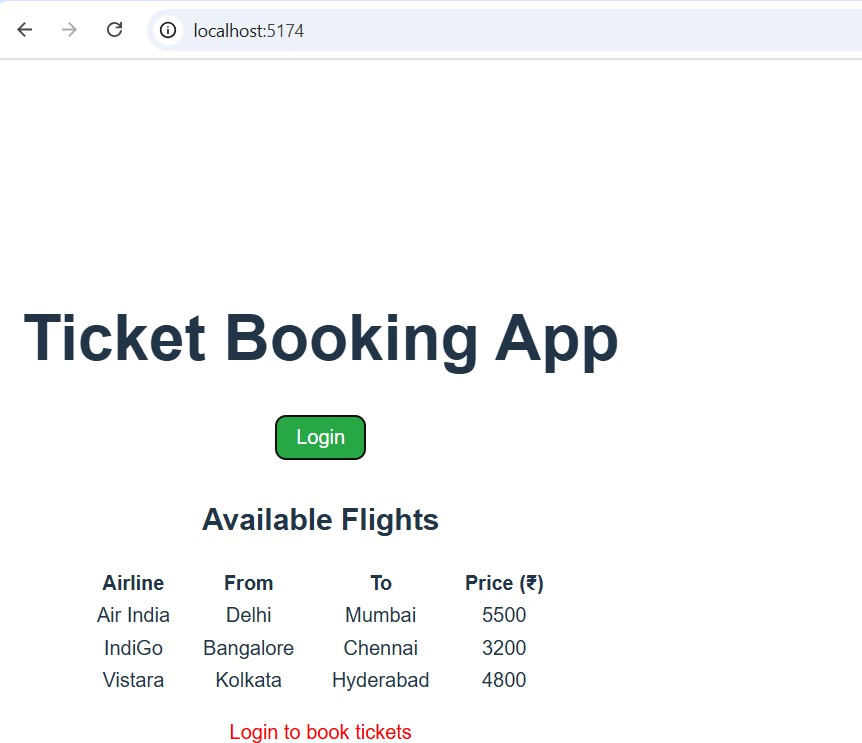
display: "inline-block", borderRadius: "5px",

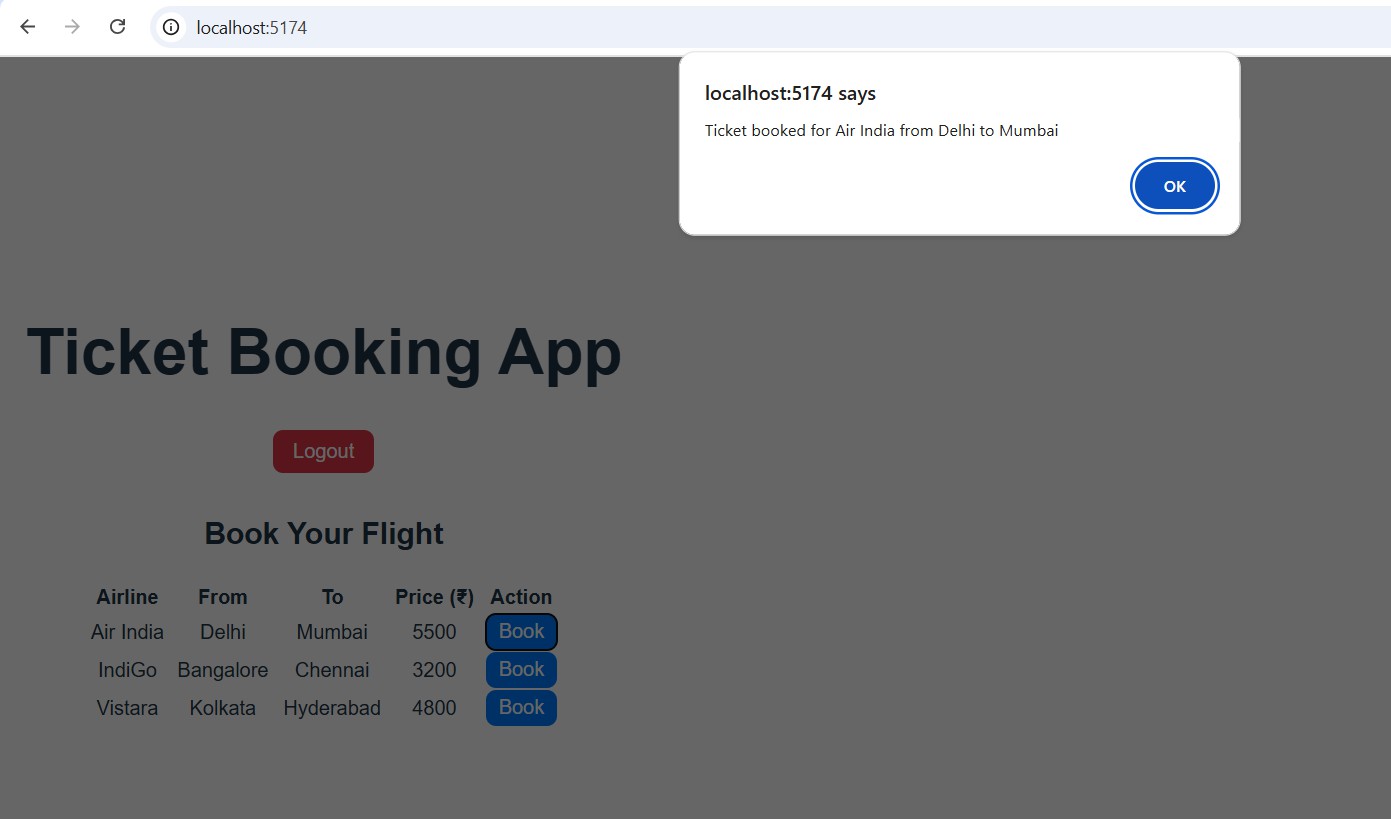
},

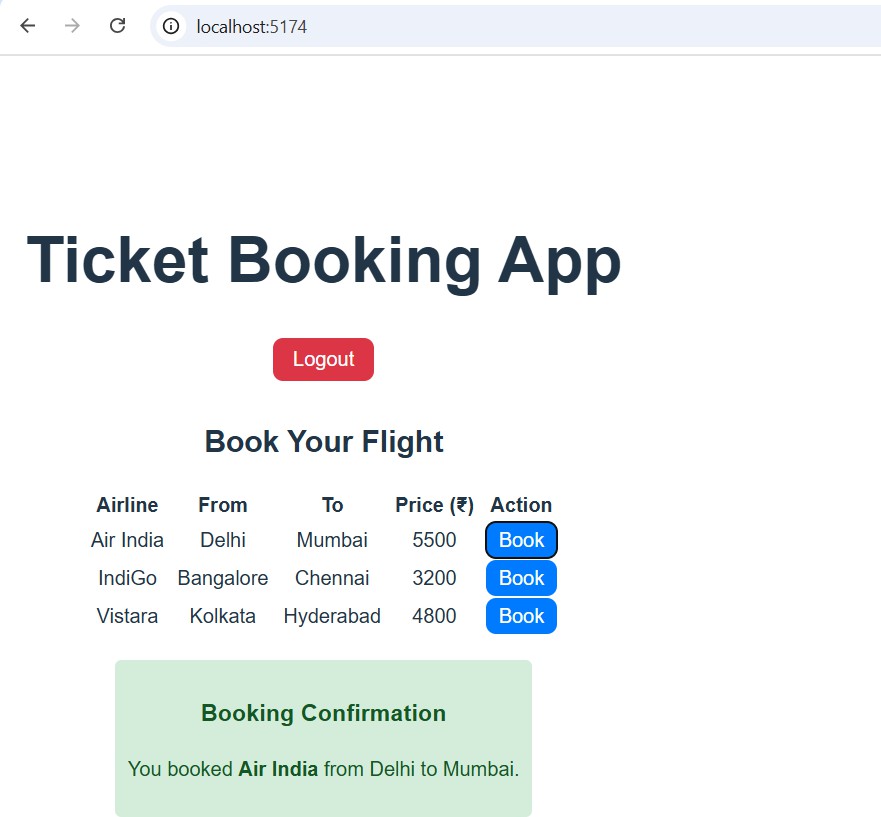
};

export default UserPage;

**Output**

****



****

**Exercise 13**

**Objectives**

* Explain various ways of conditional rendering
* Explain how to render multiple components
* Define list component
* Explain about keys in React applications
* Explain how to extract components with keys
* Explain React Map, map() function

# Answers

1. **Explain various ways of conditional rendering**

In React, conditional rendering can be done using if statements, ternary operators, logical &&, or switch statements. These methods decide which elements or components to show based on conditions.

# Explain how to render multiple components

Multiple components can be rendered by including them inside a single parent element or using React.Fragment. This allows grouping UI parts without adding unnecessary HTML elements.

# Define list component

A list component in React displays a collection of data items using iteration, often with map(). Each item is usually represented as a React element inside the list.

# Explain about keys in React applications

Keys are unique identifiers given to elements in a list to help React identify which

items have changed, added, or removed. They improve rendering performance and ensure stable component identity.

# Explain how to extract components with keys

When breaking a list into smaller components, pass the key prop to the outermost element of each component. This helps React track each extracted component correctly during updates.

# Explain React Map, map() function

In React, map() is used to transform an array of data into an array of JSX elements. It is often used to render dynamic lists from data sources.

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

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

Implement this with as many ways possible of Conditional Rendering.

# Code [App.js](http://app.js/)

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

function App() {

const [showAll, setShowAll] = useState(true); return (

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

{showAll && <CourseDetails />}

{showAll && <BookDetails />}

{showAll && <BlogDetails />}

</div>

);

}

export default App;

# [BlogDetails.js](http://blogdetails.js/)

import { blogs } from './data';

function BlogDetails() { return (

<div className="v1">

<h1>Blog Details</h1>

{blogs.map((blog, index) => (

<div key={index}>

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

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

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

</div>

))}

</div>

);

}

export default BlogDetails;

BookDetails.js

import { books } from './data';

function BookDetails() { return (

<div className="st2">

<h1>Book Details</h1>

{books.map((book) => (

<div key={book.id}>

<b>{book.bname}</b><br />

{book.price}

</div>

))}

</div>

);

}

export default BookDetails;

# [CourseDetails.js](http://coursedetails.js/)

import { courses } from './data';

function CourseDetails() { return (

<div className="mystyle1">

<h1>Course Details</h1>

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

<div key={index}>

<b>{course.name}</b><br />

{course.date}

</div>

))}

</div>

);

}

export default CourseDetails;

data.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 }

];

export const courses = [

{ name: 'Angular', date: '4/5/2021' },

{ name: 'React', date: '6/3/20201' } // typo: should be 2021

];

export const blogs = [

{

title: 'React Learning', author: 'Stephen Biz',

content: 'Welcome to learning React!'

},

{

title: 'Installation', author: 'Schwezdenier',

content: 'You can install React from npm.'

}];

Output

