**EXERCISE 1: REACT PLAYER TOGGLE APP: CONDITIONAL RENDERING OF PLAYER LISTS**

**Introduction:**

This React-based application displays a dynamic list of cricket players and allows users to toggle between a general list of players and a categorized list of Indian players. It demonstrates the use of React hooks (useState) and component-based UI rendering to manage conditional views efficiently.

**Objective:**

* To create a toggle-based React application using functional components.
* To manage state using the useState hook.
* To conditionally render different components (ListofPlayers and IndianPlayers).
* To demonstrate filtering, mapping, and array manipulation in React components.

**Implementation Breakdown:**

**Step 1: Create a new React app**

****

**Navigating**

****

**Step 2: Create Component Files**

Create two component files inside the src/Components folder:

* ListofPlayers.js
* IndianPlayers.js

**Step 3: Add the ListofPlayers Component**

// ListofPlayers.js

const ListofPlayers = () => {

const players = [

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

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

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

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

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

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

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

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

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

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

{ name: "Rohit", score: 80 }

];

return (

<div className="container">

<h2 className="section-title">List of Players</h2>

<ul className="player-list">

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

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

))}

</ul>

<h2 className="section-title">Players with Score < 70</h2>

<ul className="player-list">

{players.filter(p => p score < 70).map((player, index) => (

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

))}

</ul>

</div>

);

};

export default ListofPlayers;

**Step 4: Add the IndianPlayers Component**

// IndianPlayers.js

const IndianPlayers = () => {

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

const oddPlayers = [], evenPlayers = [];

teamPlayers.forEach((player, index) => {

if (index % 2 === 0) oddPlayers.push(player);

else evenPlayers.push(player);

});

const T20players = ["First Player", "Second Player", "Third Player"];

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

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

return (

<div className="container">

<h2 className="section-title">Odd Players</h2>

<ul className="player-list">

{oddPlayers.map((p, i) => <li key={i}>Player {i \* 2 + 1}: {p}</li>)}

</ul>

<h2 className="section-title">Even Players</h2>

<ul className="player-list">

{evenPlayers.map((p, i) => <li key={i}>Player {i \* 2 + 2}: {p}</li>)}

</ul>

<h2 className="section-title">Merged Indian Players</h2>

<ul className="player-list">

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

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

))}

</ul>

</div>

);

};

export default IndianPlayers;

**Step 5: Update App Component**

// App.js

import { useState } from 'react';

import './App.css';

import IndianPlayers from './Components/IndianPlayers';

import ListofPlayers from './Components/ListofPlayers';

const App = () => {

const [flag, setFlag] = useState(true);

return (

<div className="app">

<button className="toggle-button" onClick={() => setFlag(!flag)}>

Flag: {flag.toString()}

</button>

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

</div>

);

};

export default App;

**Step 6: Add Basic Styling**

/\* App.css \*/

.app {

padding: 20px;

font-family: sans-serif;

text-align: center;

}

.toggle-button {

padding: 10px 20px;

margin-bottom: 20px;

font-size: 16px;

}

.container {

margin-top: 20px;

}

.section-title {

color: #2e86de;

margin-top: 20px;

}

.player-list {

list-style-type: none;

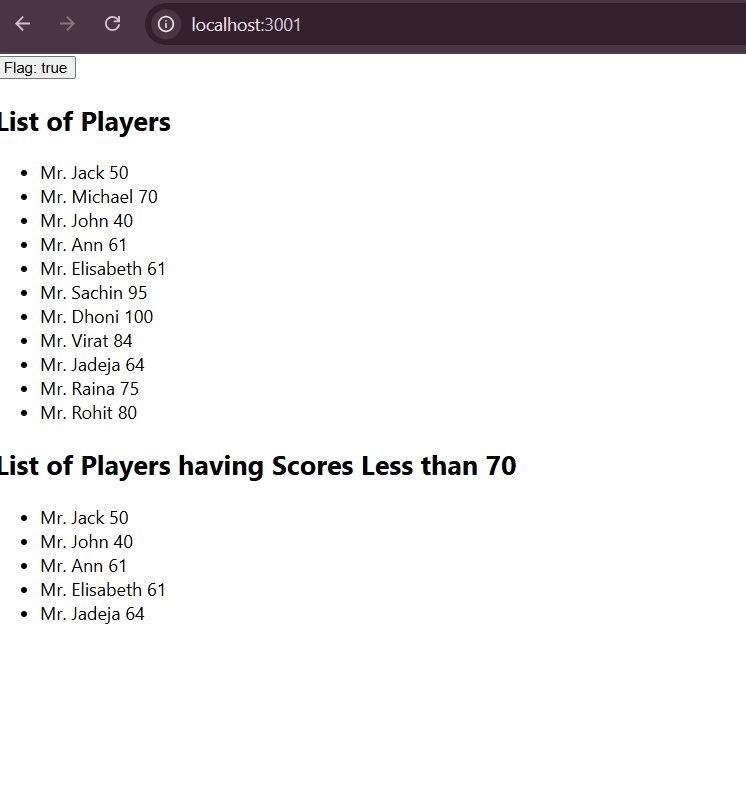
padding: 0;

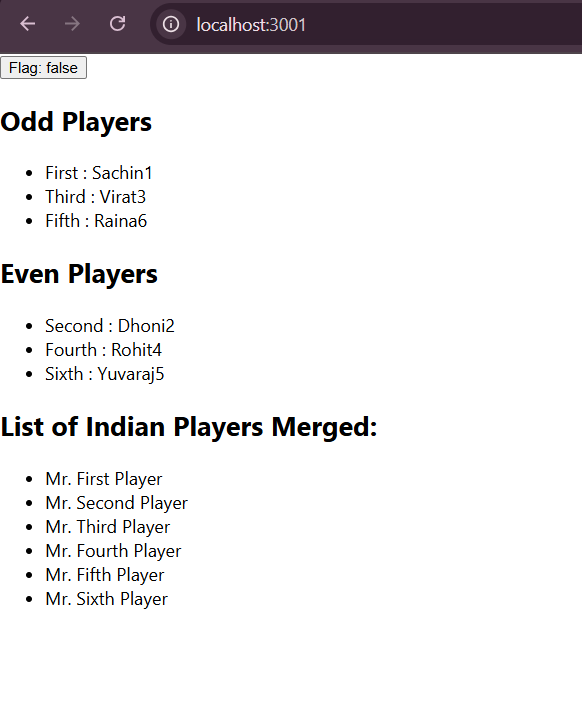
}

**Step 7: Run the Application**

****

**Output:**

****

****

**Conclusion:**

This React application demonstrates a simple use case of conditional rendering and component switching based on a boolean state. It helps reinforce React fundamentals like state management, props, and working with lists. The project can be extended to include search, sorting, or even routing to make it more dynamic and scalable.

**EXERCISE 2: OFFICE SPACE RENTAL LISTING**

**Introduction:**

This React application displays a curated list of office spaces with essential details such as name, rent, location, and images. It uses conditional styling to highlight rent values based on affordability. The application provides a clean and minimal UI, demonstrating key React concepts like map(), dynamic class assignment, and component styling.

**Objective:**

* To display a list of office spaces with key details.
* To apply conditional styling based on the rent value.
* To practice React fundamentals including JSX rendering and inline styling.
* To create a user-friendly layout that includes images and dynamic content.

**Implementation Breakdown:**

**Step 1: Create React App**

****

**Navigating**

****

**Step 2: Update App.js**

import './App.css';

function App() {

const offices = [

{

id: 1,

Name: "Regus",

Rent: 62000,

Address: "Hyderabad",

Image: "https://officebanao.com/wp-content/uploads/2024/05/bright-office-interior-with-city-view.jpg"

},

{

id: 2,

Name: "SmartWorks",

Rent: 45000,

Address: "Pune",

Image: "https://www.ecobook.io/wp-content/uploads/2022/12/1273978367-Modern-Office-Layout-compress.jpg"

}

];

return (

<div className="App">

<h1>Office Space, at Affordable Range</h1>

{offices.map((item) => {

const rentColor = item.Rent <= 60000 ? 'textRed' : 'textGreen';

return (

<div key={item.id} className="office-card">

<img

src={item.Image}

width="30%"

alt={item.Name}

style={{ margin: '20px 0' }}

/>

<h2>Name: {item.Name}</h2>

<h3 className={rentColor}>Rent: Rs. {item.Rent}</h3>

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

</div>

);

})}

</div>

);

}

export default App;

**Step 3: Add Styling in App.css**

.App {

text-align: center;

font-family: Arial, sans-serif;

}

.office-card {

border: 1px solid #ccc;

padding: 20px;

margin: 20px auto;

width: 60%;

border-radius: 10px;

background-color: #f9f9f9;

box-shadow: 2px 2px 8px rgba(0, 0, 0, 0.1);

}

.textRed {

color: red;

font-weight: bold;

}

.textGreen {

color: green;

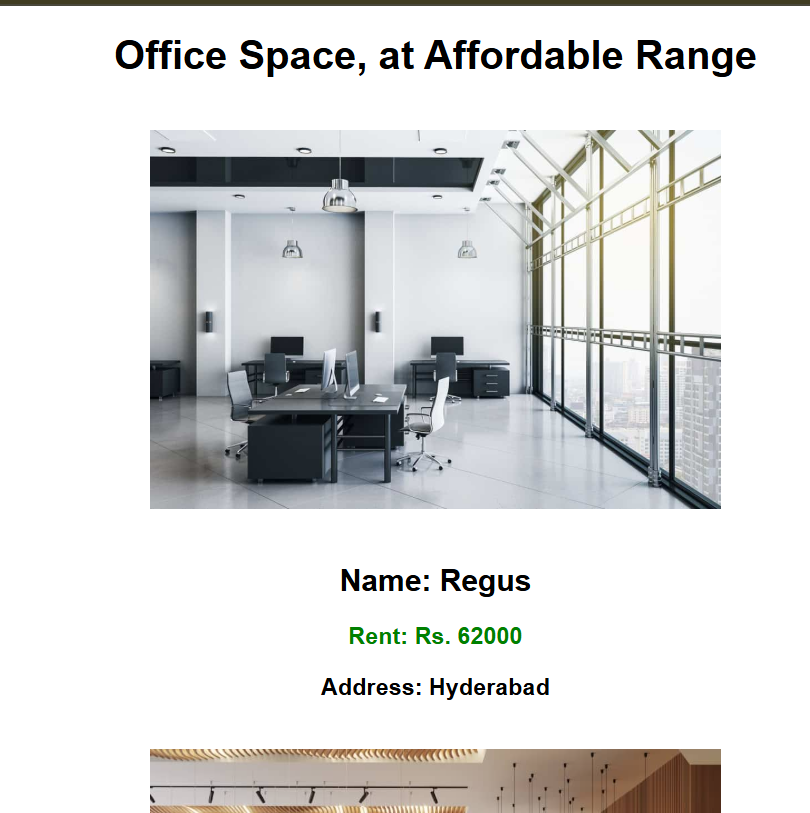
font-weight: bold;

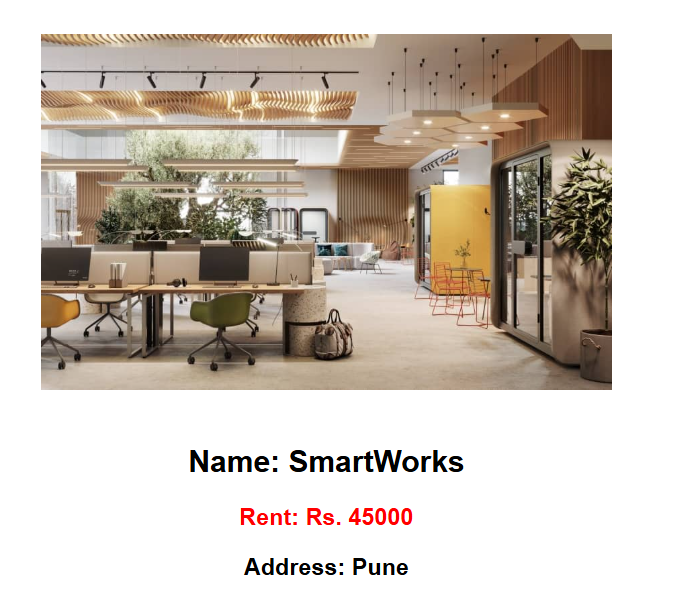
}

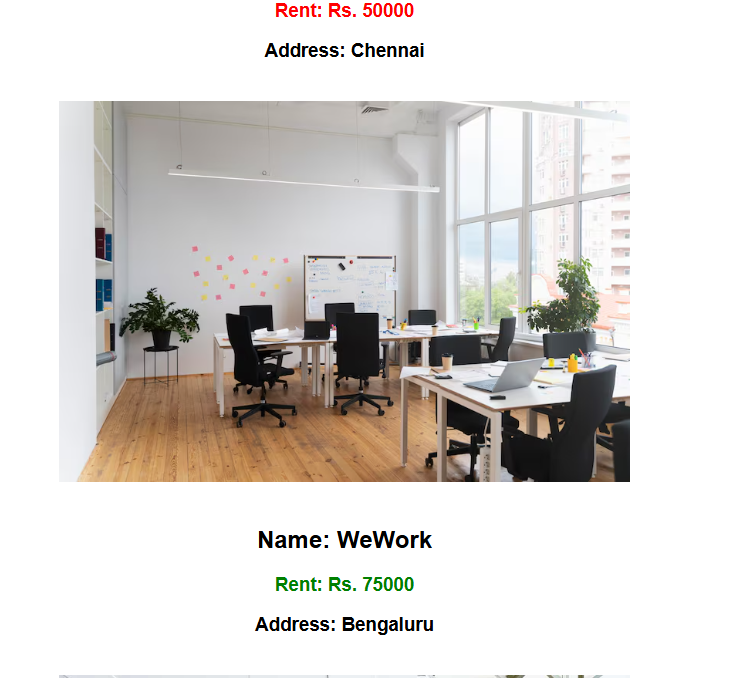
**Step 4:** **Run the application**

****

**Output:**

****

****

****

**Conclusion:**

This simple React application effectively demonstrates how to render dynamic data and conditionally apply CSS classes based on logic (e.g., rent threshold). It also provides practice in working with arrays, mapping data to components, and managing presentation with clean styling. The project can be extended to include search, filtering, or backend integration for real-world use.

**EXERCISE 3: COUNTER AND CURRENCY CONVERTER APP**

**Introduction:**

This React project combines multiple interactive features including a numeric counter, event-based alert messages, and a simple currency converter from INR to Euro. It uses React hooks (useState) for state management and applies modern CSS for a clean, user-friendly interface. The application also demonstrates handling synthetic events and form submissions in React.

**Objective:**

* To implement an interactive counter with increment/decrement functionality.
* To handle synthetic and custom events in React.
* To develop a form-based currency converter from INR to Euro.
* To apply form validation, alerts, and styled UI using CSS.

**Implementation:**

**Step 1: Set Up the React Project**

****

**Navigating**

****

**Step 2: Create Currency Converter Component**

**CurrencyConvertor.js**

import { useState } from 'react';

function CurrencyConvertor() {

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

const handleSubmit = (e) => {

e.preventDefault();

const conversionRate = 0.011; // 1 INR ≈ 0.011 EUR

const euros = parseFloat(rupees) \* conversionRate;

alert(`Converting to Euro. Amount is €${euros.toFixed(2)}`);

};

return (

<div>

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

<form onSubmit={handleSubmit}>

<label>Amount:</label>

<input type="number"

value={rupees}

onChange={(e) => setRupees(e.target.value)}

placeholder="INR"

/>

<label>Currency:</label>

<input type="text" value="Euro" readOnly />

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

</form>

</div>

);

}

export default CurrencyConvertor;

**Step 3: Update Main App Component**

**App.js**

import { useState } from 'react';

import './App.css';

import CurrencyConvertor from './CurrencyConvertor';

function App() {

const [count, setCount] = useState(0);

const increment = () => {

setCount(prev => prev + 1);

sayHello();

};

const decrement = () => {

setCount(prev => prev - 1);

};

const sayHello = () => {

alert("Hello! Counter was incremented.");

};

const sayMessage = (msg) => {

alert(msg);

};

const handleSyntheticEvent = (e) => {

e.preventDefault();

alert("I was clicked");

};

return (

<div className="App">

<h1>{count}</h1>

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

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

<br /><br />

<button onClick={() => sayMessage("Welcome!")}>Say welcome</button>

<br /><br />

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

<br /><br />

<CurrencyConvertor />

</div>

);

}

export default App;

**Step 4: Add Styling**

**App.css**

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color: #f5f5f5;

margin: 0;

padding: 0;

}

.App {

text-align: center;

margin-top: 50px;

}

h1 {

font-size: 3rem;

color: #333;

}

button {

padding: 10px 20px;

margin: 5px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 6px;

font-size: 1rem;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #45a049;

}

form {

display: inline-block;

text-align: left;

background-color: #ffffff;

padding: 20px 30px;

border-radius: 10px;

box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);

}

label {

display: block;

margin-top: 10px;

margin-bottom: 5px;

font-weight: 500;

}

input[type="number"],

input[type="text"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 6px;

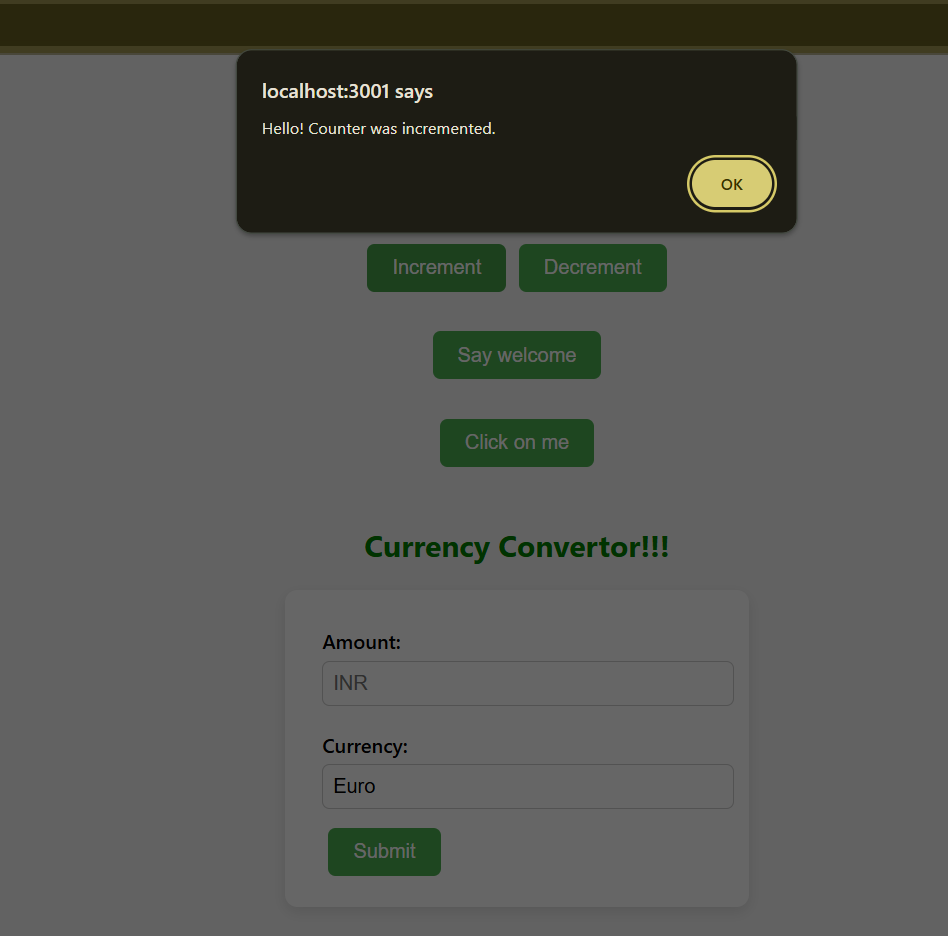
font-size: 1rem;

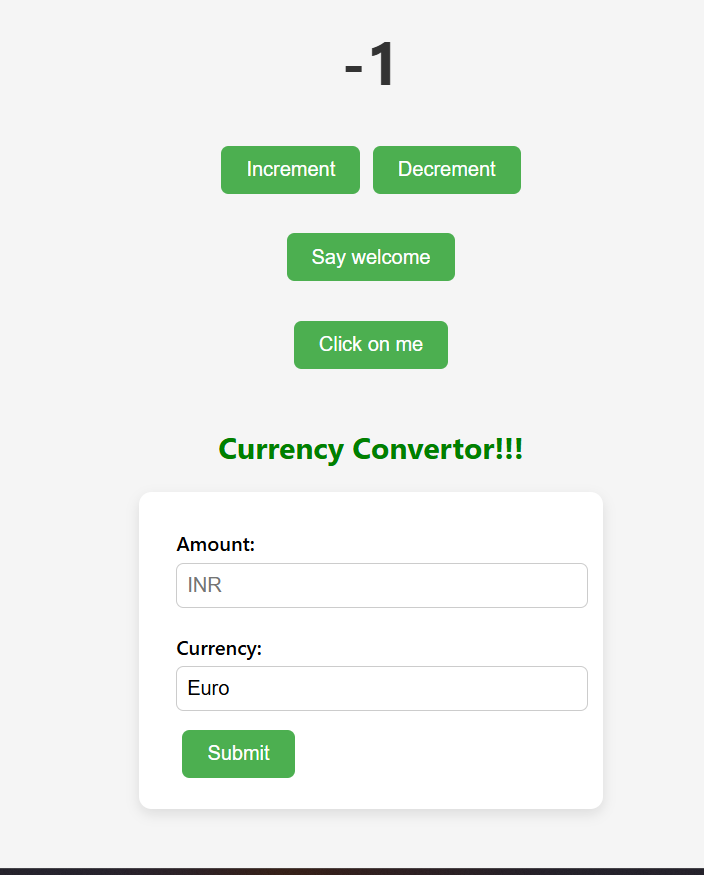
}

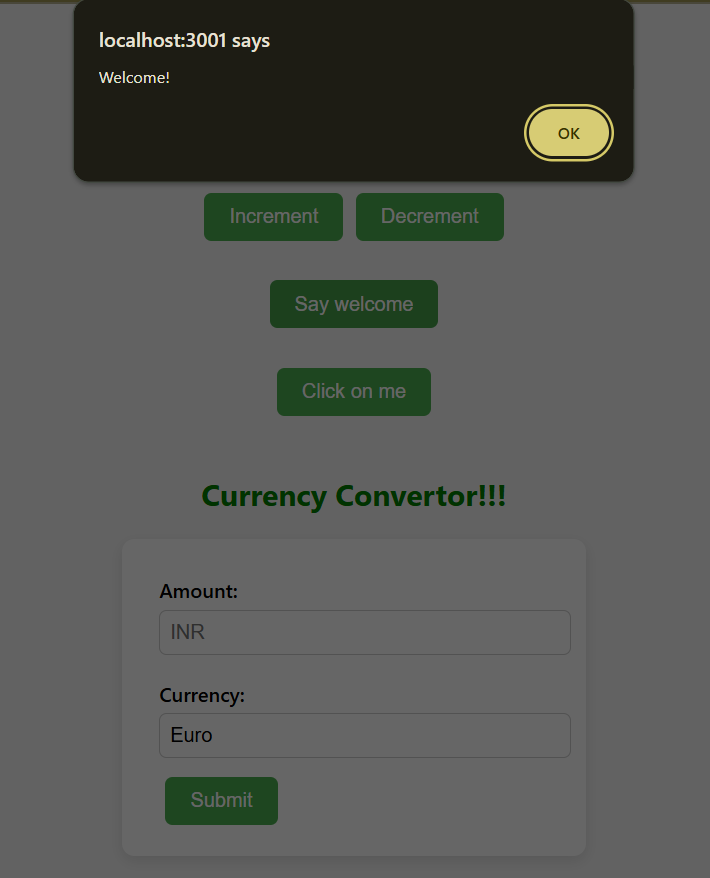
**Step 5:** **Run the Project**

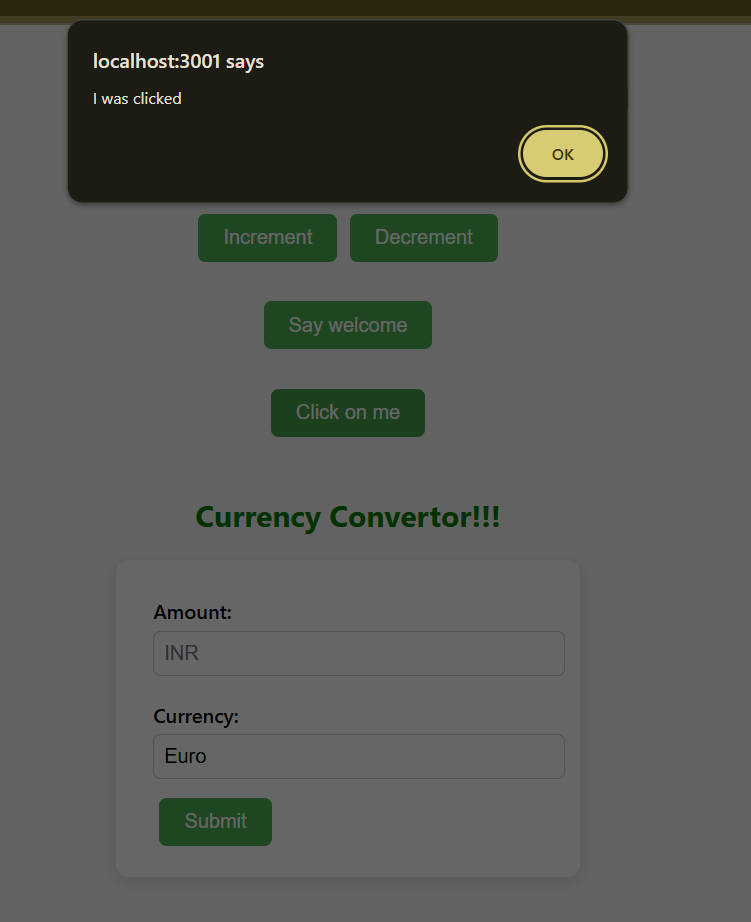


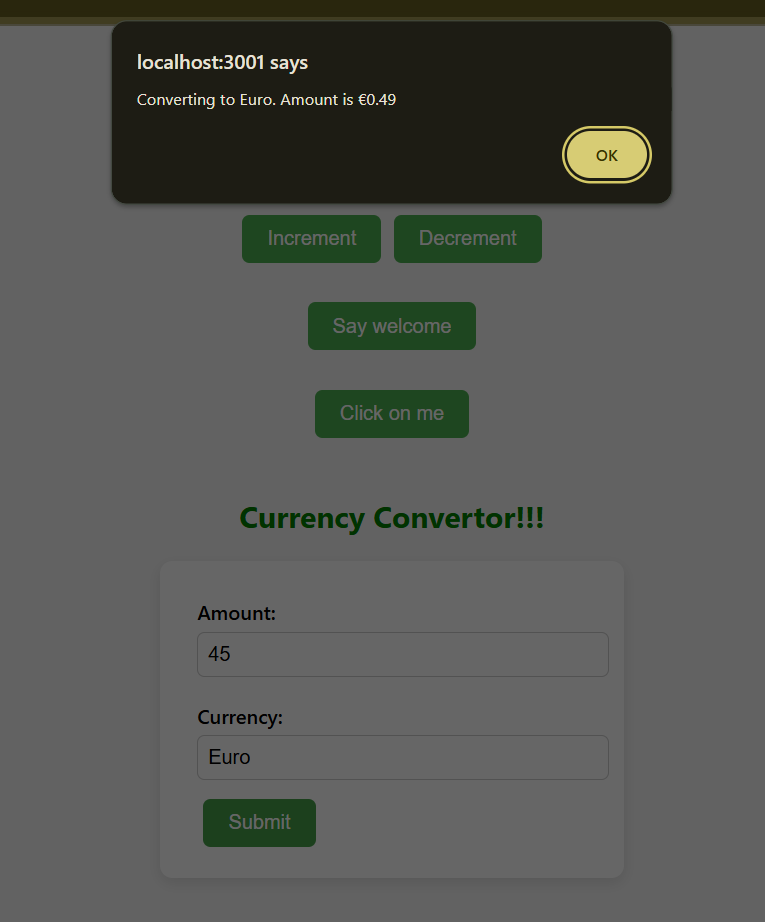
**Output:**

****

****

****

****

****

**Conclusion:**

This project showcases how React can be used to build interactive and visually appealing applications. It integrates basic arithmetic operations, form handling, alerts, and currency conversion logic. The application is scalable and can be extended to support additional currencies, input validations, or even API-based live currency exchange rates.

**EXERCISE 4 : TICKET BOOKING APP**

**Introduction:**

This React application simulates a basic flight booking system where users can log in to book flights, and guests can only view available options. It demonstrates the use of props, conditional rendering, component modularization, and event handling in a React environment. A clean UI with CSS enhances the user experience.

**Objective:**

* To implement role-based conditional rendering (Guest vs User).
* To display available flights using props and map-based rendering.
* To enable flight booking functionality only for logged-in users.
* To create reusable components (User, Guest, FlightDetails) using props.
* To apply CSS for a visually appealing and structured interface.

**Implementation:**

**Step 1: Set Up the React Project**

****

**Navigating**

****

**Step 2: Folder Structure**

Create the following components inside a new folder src/Components/:

* User.js
* Guest.js
* FlightDetails.js

**Step 3: Create the FlightDetails Component**

**FlightDetails.js**

function FlightDetails({ canBook }) {

const flights = [

{ id: 1, from: "Chennai", to: "Delhi", price: 4500 },

{ id: 2, from: "Mumbai", to: "Kolkata", price: 5000 },

{ id: 3, from: "Bangalore", to: "Hyderabad", price: 3000 },

{ id: 4, from: "Pune", to: "Ahmedabad", price: 3200 },

{ id: 5, from: "Jaipur", to: "Goa", price: 3800 },

{ id: 6, from: "Lucknow", to: "Chandigarh", price: 4100 }

];

const handleBooking = (flight) => {

alert(`Ticket booked: ${flight.from} to ${flight.to} at ₹${flight.price}`);

};

return (

<div className="flights">

<h3>Available Flights</h3>

{flights.map(flight => (

<div className="flight-card" key={flight.id}>

<p><strong>From:</strong> {flight.from}</p>

<p><strong>To:</strong> {flight.to}</p>

<p><strong>Price:</strong> ₹{flight.price}</p>

{canBook && (

<button onClick={() => handleBooking(flight)}>Book</button>

)}

</div>

))}

</div>

);

}

export default FlightDetails;

**Step 4: Create User.js**

**User.js**

import FlightDetails from './FlightDetails';

function User({ handleLogout }) {

return (

<div className="card">

<h2>Welcome Back, User!</h2>

<p>You can now book your flight tickets.</p>

<FlightDetails canBook={true} />

<button onClick={handleLogout}>Logout</button>

</div>

);

}

export default User;

**Step 5: Create Guest.js**

**Guest.js**

import FlightDetails from './FlightDetails';

function Guest({ handleLogin }) {

return (

<div className="card">

<h2>Welcome Guest!</h2>

<p>Login to book your flight tickets.</p>

<FlightDetails canBook={false} />

<button onClick={handleLogin}>Login</button>

</div>

);

}

export default Guest;

**Step 6: Update the App.js File**

**App.js**

import { useState } from 'react';

import './App.css';

import Guest from './Components/Guest';

import User from './Components/User';

function App() {

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

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

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

return (

<div className="App">

<h1>✈ Flight Booking App</h1>

{isLoggedIn ? (

<User handleLogout={handleLogout} />

) : (

<Guest handleLogin={handleLogin} />

)}

</div>

);

}

export default App;

**Step 7: Add CSS Styling**

**App.css**

body {

margin: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color: #f0f4f8;

}

.App {

text-align: center;

padding: 2rem;

}

.card {

background-color: #fff;

padding: 2rem;

margin: 1rem auto;

width: 80%;

max-width: 500px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

border-radius: 12px;

}

button {

padding: 10px 20px;

margin-top: 1rem;

border: none;

background-color: #007bff;

color: white;

border-radius: 8px;

cursor: pointer;

font-size: 1rem;

}

button:hover {

background-color: #0056b3;

}

.flights {

margin-top: 1rem;

}

.flight-card {

background-color: #eaf4fc;

padding: 1rem;

margin: 1rem auto;

border-radius: 10px;

width: 90%;

text-align: left;

}

.flight-card p {

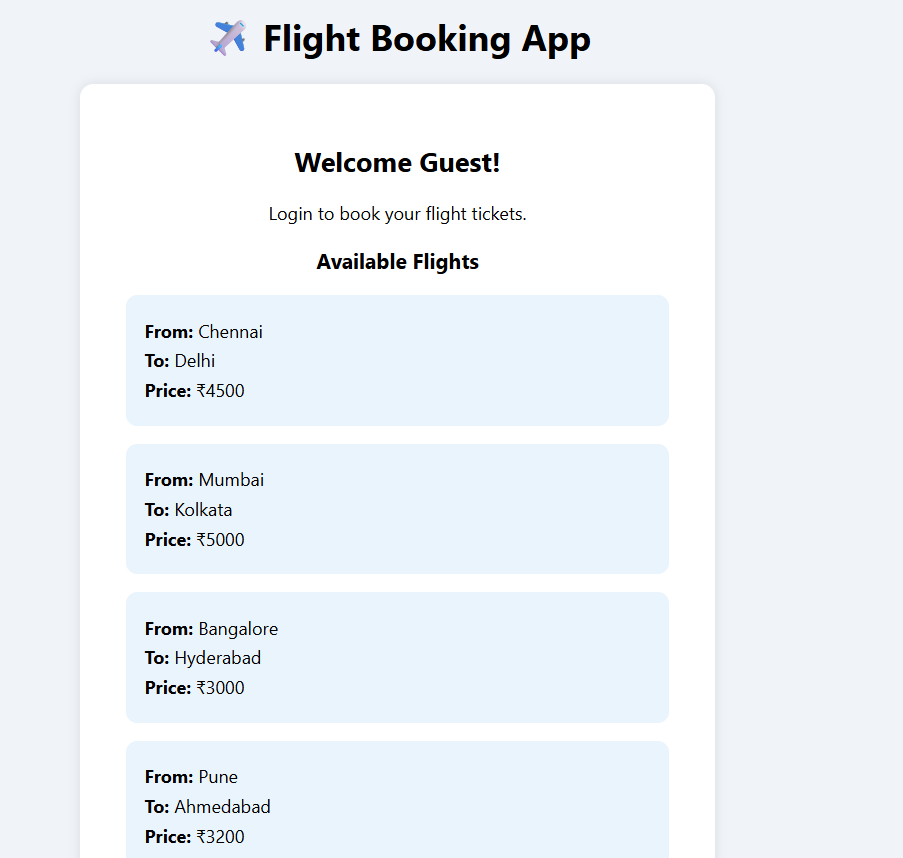
margin: 5px 0;

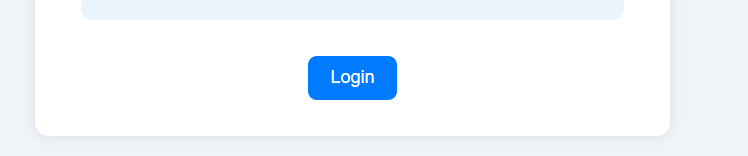
}

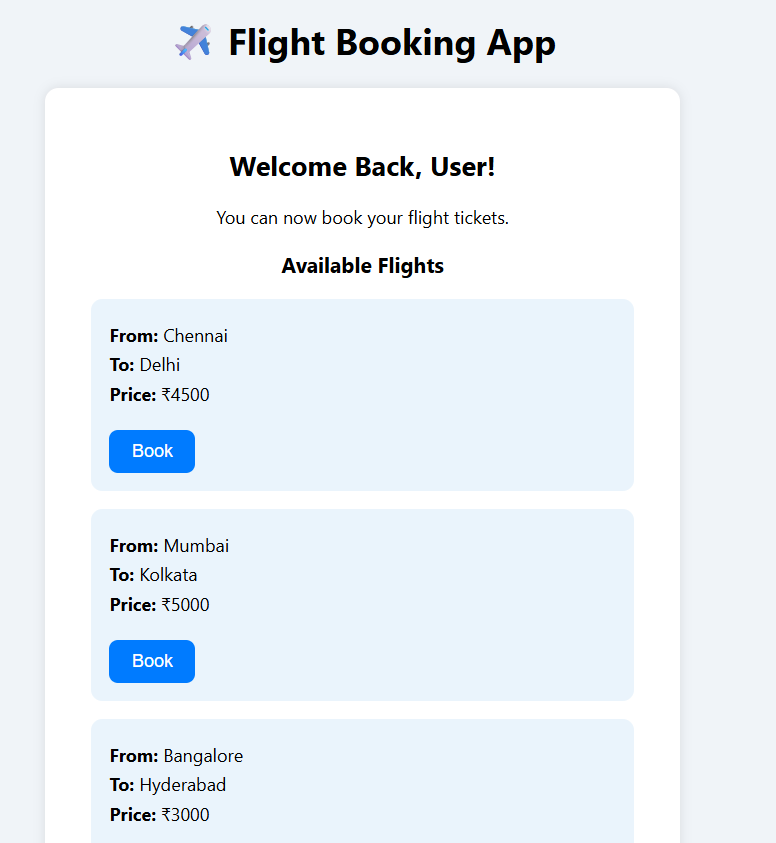
**Step 8: Run the Application**



**Output:**

****

****

****

**Conclusion:**

This React project effectively demonstrates the concept of role-based rendering, allowing users to log in and interact with data differently from guests. By separating concerns into components and using props, the app remains clean and maintainable.

**EXERCISE 5 : BOOKS, BLOGS & COURSES TOGGLE APP**

**Introduction:**

This React application allows users to switch between a list of books, blogs, and courses using buttons. It uses component-based architecture and conditional rendering to create an interactive user experience, along with clean, responsive CSS styling.

**Objective:**

* Build a modular React application using functional components.
* Implement conditional rendering with useState.
* Display categorized content (Books, Blogs, Courses) dynamically.
* Apply CSS to enhance UI layout and user interaction.

**Implementation:**

**Step 1: Create React App**

****

**Navigating**

****

**Step 2: Create Component Files**

Create three files in the src folder:

* BookDetails.js
* BlogDetails.js
* CourseDetails.js

**Step 3: Add Component Code**

**BookDetails.js**

function BookDetails() {

const books = [

{ id: 1, title: "React Explained", author: "Zac Gordon" },

{ id: 2, title: "Learning React", author: "Alex Banks" },

{ id: 3, title: "The Road to React", author: "Robin Wieruch" },

{ id: 4, title: "React Up & Running", author: "Stoyan Stefanov" },

{ id: 5, title: "Fullstack React", author: "Anthony Accomazzo" },

{ id: 6, title: "React and Redux", author: "Adam Boduch" }

];

return (

<div className="card">

<h2>Book Details</h2>

<ul>

{books.map((book) => (

<li key={book.id}>

<b>{book.title}</b> by {book.author}

</li>

))}

</ul>

</div>

);

}

export default BookDetails;

**BlogDetails.js**

function BlogDetails() {

const blogs = [

{ id: 1, title: "Understanding React State", author: "Hemanath" },

{ id: 2, title: "React Hooks Deep Dive", author: "Anjali R" },

{ id: 3, title: "Optimizing React Apps", author: "Vikram S" },

{ id: 4, title: "React vs Angular", author: "Neha K" },

{ id: 5, title: "React for Beginners", author: "John Doe" },

{ id: 6, title: "Top 10 React Libraries", author: "Ayesha K" }

];

return (

<div className="card">

<h2>Blog Details</h2>

<ul>

{blogs.map((blog) => (

<li key={blog.id}>

<b>{blog.title}</b> - {blog.author}

</li>

))}

</ul>

</div>

);

}

export default BlogDetails;

**CourseDetails.js**

function CourseDetails() {

const courses = [

{ id: 1, name: "ReactJS", duration: "6 weeks" },

{ id: 2, name: "NodeJS", duration: "4 weeks" },

{ id: 3, name: "Fullstack Development", duration: "12 weeks" },

{ id: 4, name: "Frontend with React", duration: "8 weeks" },

{ id: 5, name: "React Native", duration: "5 weeks" },

{ id: 6, name: "Advanced React Patterns", duration: "3 weeks" }

];

return (

<div className="card">

<h2>Course Details</h2>

<ul>

{courses.map((course) => (

<li key={course.id}>

<b>{course.name}</b> - Duration: {course.duration}

</li>

))}

</ul>

</div>

);

}

export default CourseDetails;

**Step 4: Main App File**

**App.js**

import { useState } from "react";

import "./App.css";

import BlogDetails from "./components/BlogDetails";

import BookDetails from "./components/BookDetails";

import CourseDetails from "./components/CourseDetails";

function App() {

const [selected, setSelected] = useState("book");

return (

<div className="App">

<h1>Blogger App</h1>

<div className="buttons">

<button onClick={() => setSelected("book")}>Book</button>

<button onClick={() => setSelected("blog")}>Blog</button>

<button onClick={() => setSelected("course")}>Course</button>

</div>

{selected === "book" && <BookDetails />}

{selected === "blog" ? <BlogDetails /> : null}

{selected === "course" ? (

<CourseDetails />

) : (

selected !== "course" && <p>Select Course to view details.</p>

)}

</div>

);

}

export default App;

**Step 5: Add Styling**

**App.css**

.App {

font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;

text-align: center;

padding: 30px;

background: linear-gradient(to right, #f0f2f5, #dbe4f0);

min-height: 100vh;

}

.buttons {

margin-bottom: 25px;

}

button {

margin: 8px;

padding: 12px 18px;

font-size: 16px;

font-weight: bold;

border: none;

border-radius: 8px;

background-color: #007bff;

color: white;

transition: all 0.3s ease;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

.card {

background: white;

margin: 20px auto;

width: 80%;

max-width: 600px;

border-radius: 15px;

padding: 25px 30px;

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.1);

text-align: left;

transition: transform 0.2s ease;

}

.card:hover {

transform: scale(1.01);

}

.card h2 {

color: #333;

margin-bottom: 15px;

border-bottom: 2px solid #e0e0e0;

padding-bottom: 10px;

}

ul {

list-style: none;

padding: 0;

}

li {

padding: 10px 0;

border-bottom: 1px solid #eaeaea;

font-size: 16px;

}

li:last-child {

border-bottom: none;

}

b {

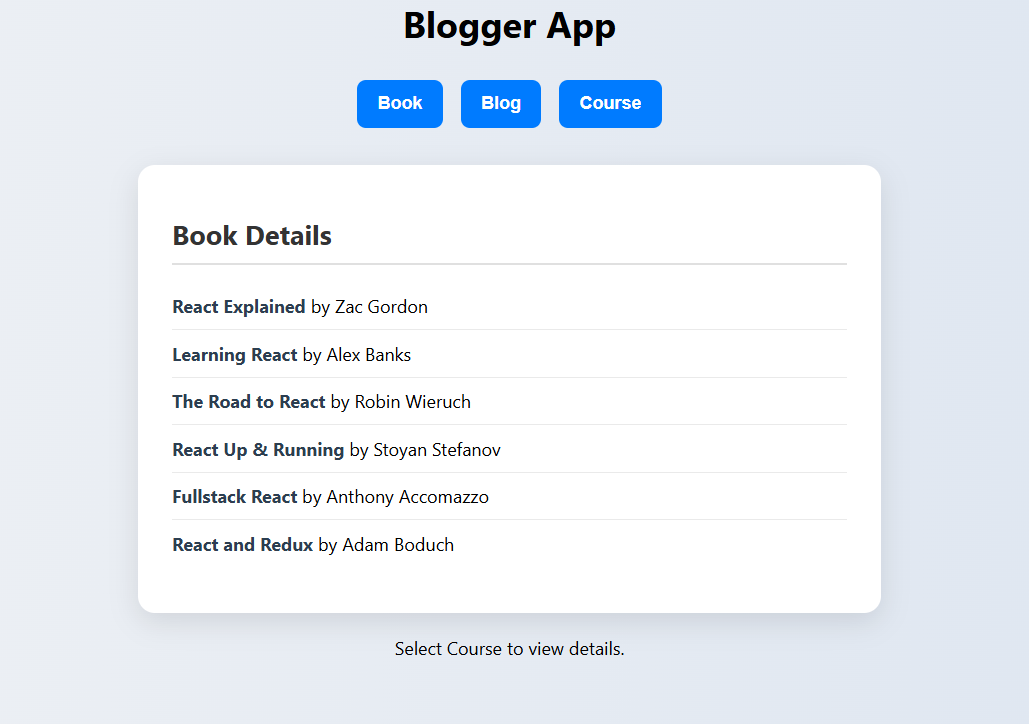
color: #2c3e50;

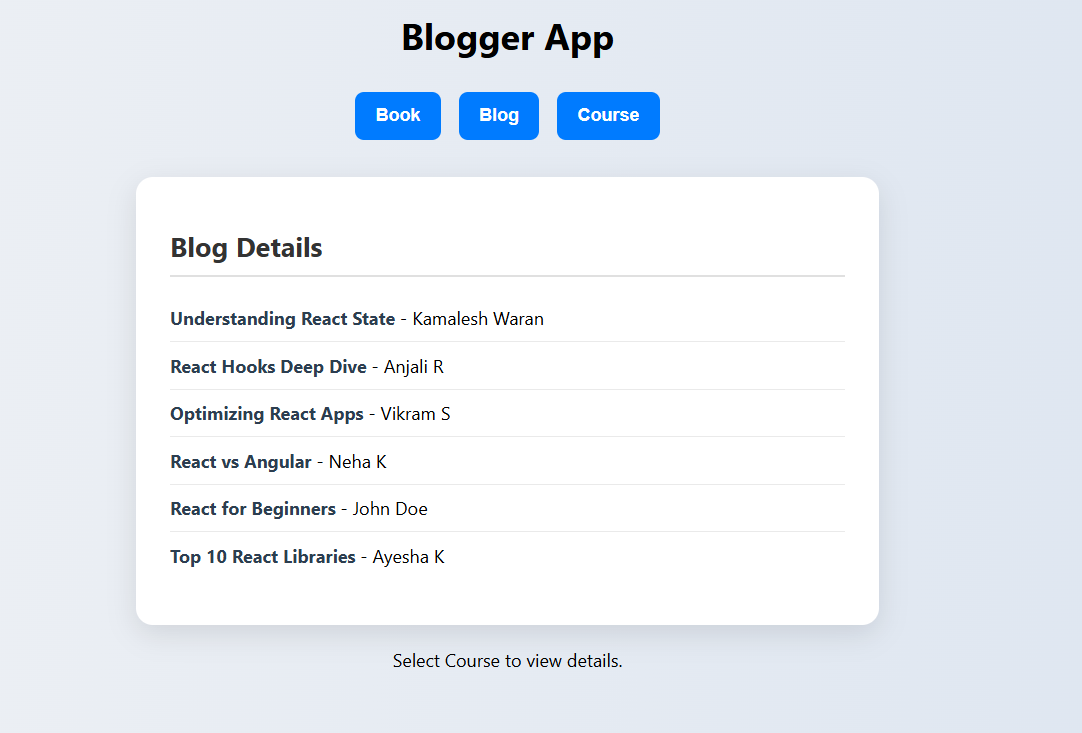
}

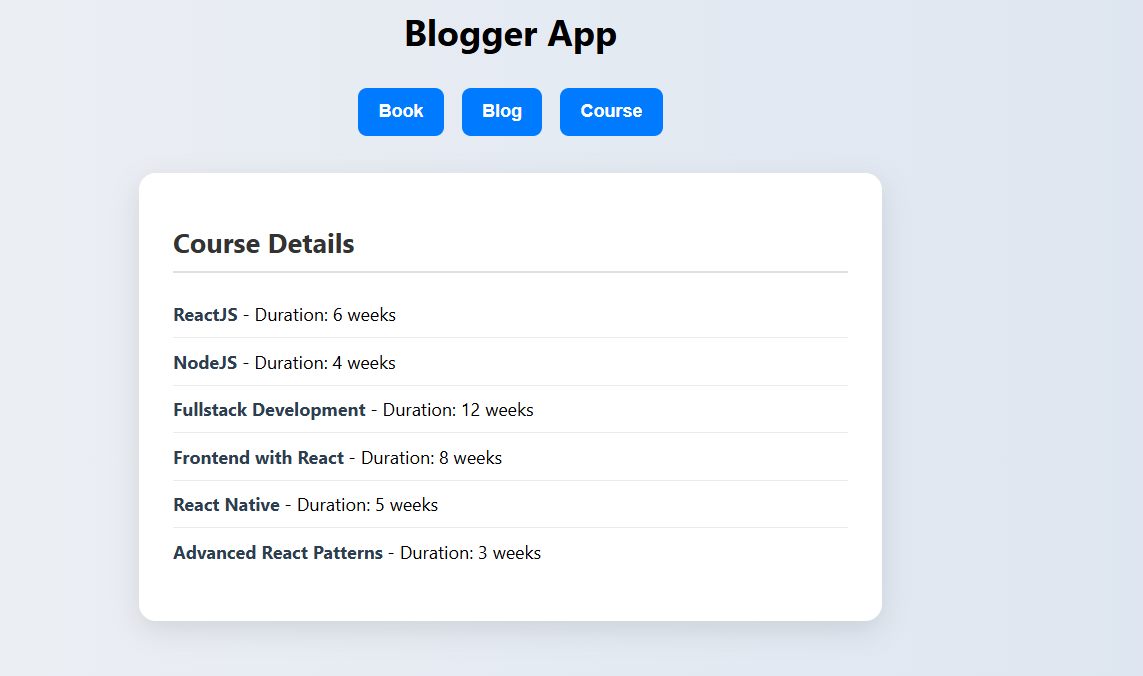
**Step 6 : Run the Application**

****

**Output:**

****

****

****

**Conclusion:**

This project demonstrates basic React skills such as conditional rendering, state management, and component-based UI design. It is a great starting point for learners exploring frontend development using React. The toggle-based interface provides a neat way to explore categorized information interactively.