**Mandatory Handson Week 7 Solutions:**

**Shashwat Bharadwaj**

**Superset ID:6360517**

**Roll no:2228059**

**9.**

**Indianplayers.js**

import React from 'react';

const IndianPlayers = () => {

// Array of Indian players with positions

const indianPlayers = [

{ position: 'First', name: 'Sachin1' },

{ position: 'Second', name: 'Dhoni2' },

{ position: 'Third', name: 'Virat3' },

{ position: 'Fourth', name: 'Rohit4' },

{ position: 'Fifth', name: 'Yuvaraj5' },

{ position: 'Sixth', name: 'Raina6' }

];

// Destructuring to separate odd and even players

const [first, second, third, fourth, fifth, sixth] = indianPlayers;

const oddPlayers = [first, third, fifth];

const evenPlayers = [second, fourth, sixth];

// Two arrays for T20 and Ranji Trophy players

const t20Players = [

'Mr. First Player',

'Mr. Second Player',

'Mr. Third Player'

];

const ranjiTrophyPlayers = [

'Mr. Fourth Player',

'Mr. Fifth Player',

'Mr. Sixth Player'

];

// Merge the two arrays using ES6 spread operator

const mergedPlayers = [...t20Players, ...ranjiTrophyPlayers];

return (

<div>

<h2>Odd Players:</h2>

<ul>

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

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

))}

</ul>

<h2>Even Players:</h2>

<ul>

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

<li key={index}>{player.position} : {player.name}</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.js**

import React from 'react';

const ListofPlayers = () => {

// Array with 11 players and their scores

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 }

];

// Filter players with scores below 70 using arrow function

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

return (

<div>

<h2>List of Players</h2>

<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>

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

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

))}

</ul>

</div>

);

};

export default ListofPlayers;

**App.js**

import React, { useState } from 'react';

import './App.css';

import ListofPlayers from './components/ListofPlayers';

import IndianPlayers from './components/IndianPlayers';

function App() {

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

return (

<div className="App">

<header className="App-header">

<h1>Cricket App</h1>

<button onClick={() => setFlag(!flag)}>

Toggle Flag (Current: {flag.toString()})

</button>

</header>

<main>

{flag ? (

<div>

<h2>Output: When Flag=true</h2>

<ListofPlayers />

</div>

) : (

<div>

<h2>When Flag=false</h2>

<IndianPlayers />

</div>

)}

</main>

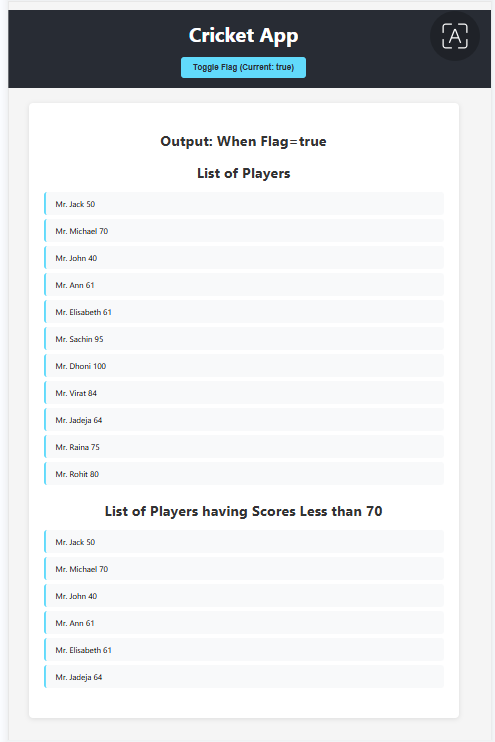
</div>

);

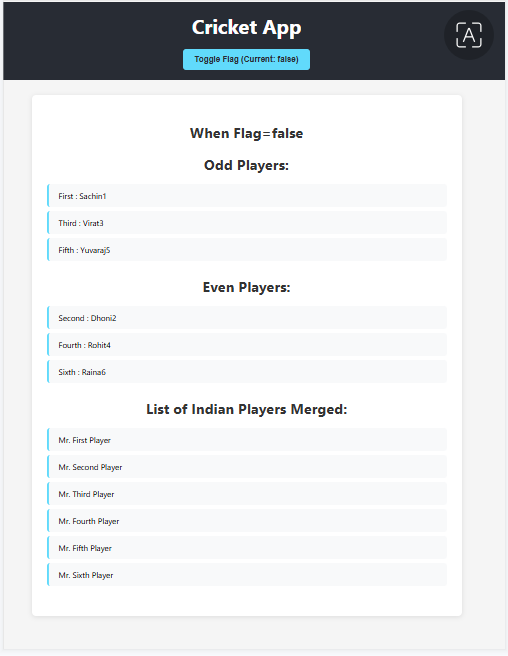
}

export default App;

OUTPUT1



OUTPUT2



**10.**

**-App.js**

import React from 'react';

import picture1 from './assets/picture1.jpg';

import picture2 from './assets/picture2.jpg';

import picture3 from './assets/picture3.jpg';

// List of office data as objects

const offices = [

{

id: 1,

name: 'DBS',

rent: 50000,

address: 'Chennai',

image: picture1,

},

{

id: 2,

name: 'Regus',

rent: 65000,

address: 'Bangalore',

image: picture2,

},

{

id: 3,

name: 'WeWork',

rent: 55000,

address: 'Mumbai',

image: picture3,

}

];

// Create style for rent dynamically

const getRentStyle = (rent) => ({

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

fontWeight: 'bold',

fontSize: '22px',

marginTop: '8px',

});

// Style objects for reuse

const headingStyle = {

textAlign: 'center',

marginTop: '40px',

fontWeight: 'bold',

fontSize: '2rem',

};

const cardStyle = {

margin: '30px auto',

padding: '22px',

border: '1px solid #eee',

borderRadius: '12px',

maxWidth: '430px',

boxShadow: '0 4px 10px #ddd',

};

const imageStyle = {

display: 'block',

margin: '0 auto 18px',

width: '68%',

height: '170px',

objectFit: 'cover',

borderRadius: '9px',

};

const nameStyle = {

fontWeight: 'bold',

fontSize: '1.4rem',

marginTop: '5px',

};

const addressStyle = {

marginTop: '10px',

fontWeight: 'bold',

};

function App() {

return (

<div style={{ margin: '0 auto', maxWidth: '900px', padding: '35px 0' }}>

{/\* Heading element \*/}

<h1 style={headingStyle}>Office Space , at Affordable Range</h1>

{/\* Loop through the offices list \*/}

{offices.map(office => (

<div key={office.id} style={cardStyle}>

{/\* Attribute to display image of the office space \*/}

<img src={office.image} alt={office.name} style={imageStyle} />

{/\* Display office details with proper styles \*/}

<div style={nameStyle}>Name: {office.name}</div>

{/\* Display Rent with conditional styling \*/}

<div style={getRentStyle(office.rent)}>Rent: Rs. {office.rent}</div>

<div style={addressStyle}>Address: {office.address}</div>

</div>

))}

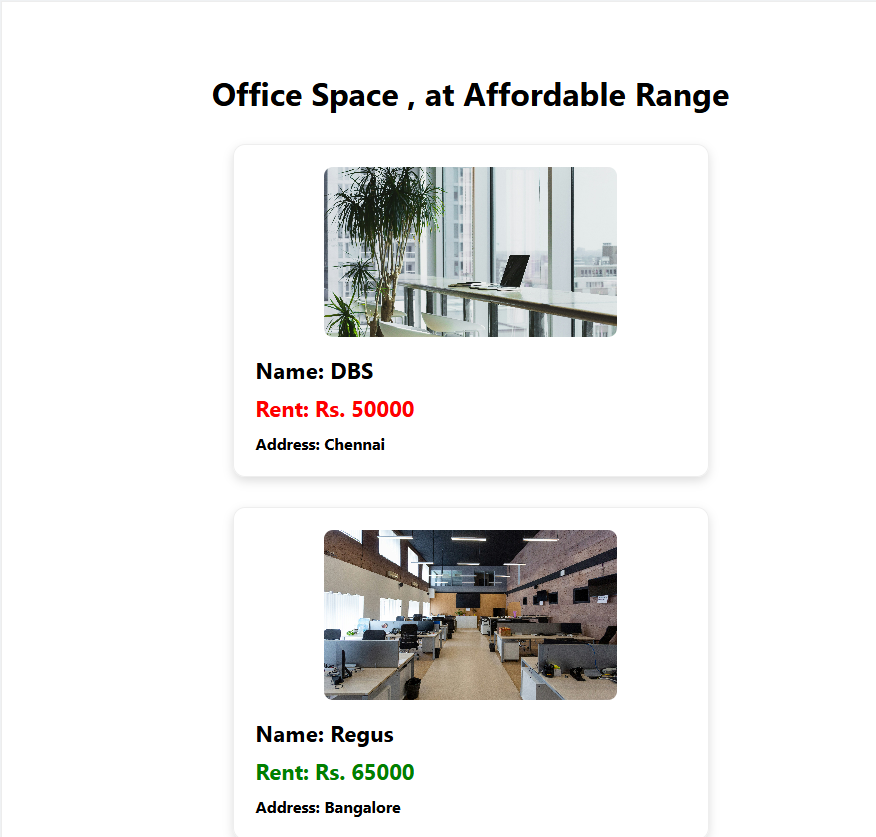
</div>

);

}

export default App;

OUTPUT



**11.**

**-App.js**

import React, { useState } from 'react';

const INR\_TO\_EURO = 0.011;

const EURO\_TO\_INR = 90.91;

function App() {

const [count, setCount] = useState(5);

const [amount, setAmount] = useState('');

const [currency, setCurrency] = useState('INR');

const [result, setResult] = useState('');

const increment = () => setCount(prev => prev + 1);

const sayHello = () => alert("localhost:3000 says: \nhello member");

const handleIncrement = () => {

increment();

sayHello();

};

const handleDecrement = () => setCount(prev => prev - 1);

const handleWelcome = (msg) => alert(`localhost:3000 says\n${msg}`);

const handleOnPress = (event) => {

alert('localhost:3000 says\nI was clicked');

};

const handleCurrencySubmit = (e) => {

e.preventDefault();

let output = '';

if (currency === 'INR') {

const euro = (parseFloat(amount) \* INR\_TO\_EURO).toFixed(2);

output = `Converting to Euro amount is €${euro}`;

} else {

const inr = (parseFloat(amount) \* EURO\_TO\_INR).toFixed(2);

output = `Converting to Rupees amount is ₹${inr}`;

}

alert(`localhost:3000 says\n${output}`);

setResult(output);

};

return (

<div style={{ margin: 20, fontFamily: 'Arial' }}>

{/\* Counter and Buttons \*/}

<div>

<button onClick={handleIncrement} style={{ marginRight: 5 }}>Increment</button>

<button onClick={handleDecrement} style={{ marginRight: 5 }}>Decrement</button>

<button onClick={() => handleWelcome('welcome')} style={{ marginRight: 5 }}>Say welcome</button>

<button onClick={handleOnPress} style={{ marginRight: 5 }}>Click on me!</button>

<br /><br />

<span style={{ fontWeight: 'bold', fontSize: 18 }}>{count}</span>

</div>

{/\* Currency Converter Section \*/}

<h2 style={{ color: 'green', marginTop: 38 }}>Currency Convertor!!!</h2>

<form onSubmit={handleCurrencySubmit}>

<label>

Amount:<input

type="number"

min="0"

required

value={amount}

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

style={{ marginLeft: 6, marginRight: 12 }}

/>

</label>

<label>

Currency:

<select value={currency}

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

style={{ marginLeft: 6, marginRight: 12 }}>

<option value="INR">INR</option>

<option value="EURO">EURO</option>

</select>

</label>

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

</form>

{/\* Show result text below \*/}

{result && (

<p style={{ color: '#555', marginTop: 12 }}>{result}</p>

)}

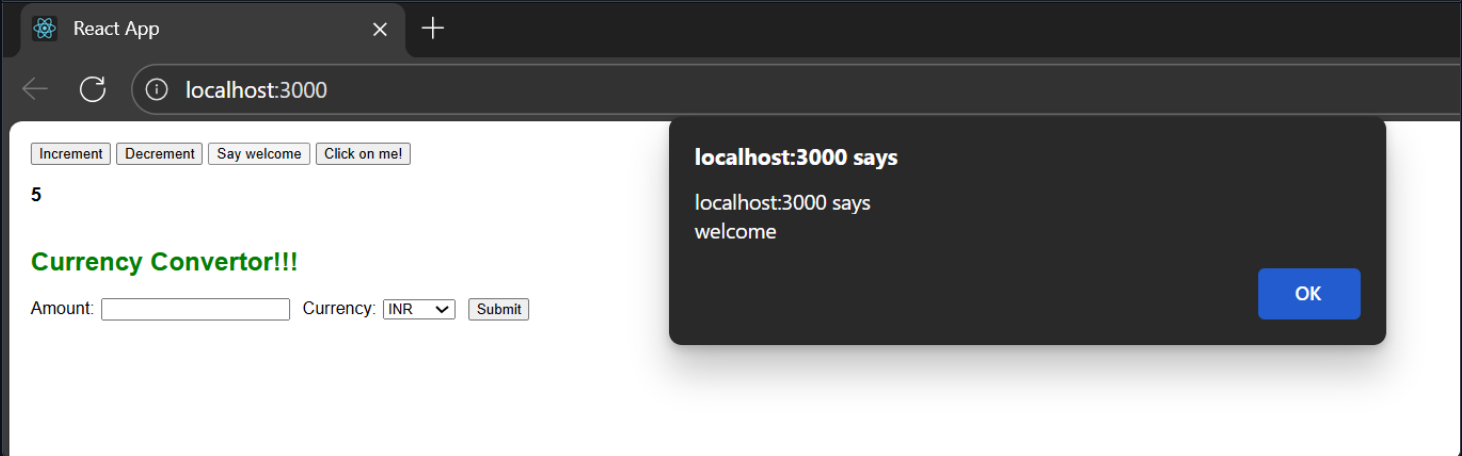
</div>

);

}

export default App;

**OUTPUT**



**-Guestpage.css**

.guest-page {

max-width: 100%;

}

.guest-header {

background: rgba(255, 255, 255, 0.95);

backdrop-filter: blur(10px);

border-radius: 20px;

padding: 2rem;

margin-bottom: 2rem;

text-align: center;

box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);

border: 1px solid rgba(255, 255, 255, 0.2);

}

.guest-header h2 {

color: #2c3e50;

font-size: 2.5rem;

margin-bottom: 1rem;

font-weight: 700;

}

.guest-header p {

color: #7f8c8d;

font-size: 1.2rem;

margin-bottom: 2rem;

}

.login-btn {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

color: white;

border: none;

padding: 1rem 2rem;

font-size: 1.1rem;

font-weight: 600;

border-radius: 50px;

cursor: pointer;

transition: all 0.3s ease;

box-shadow: 0 4px 15px rgba(102, 126, 234, 0.4);

}

.login-btn:hover {

transform: translateY(-2px);

box-shadow: 0 6px 20px rgba(102, 126, 234, 0.6);

}

.flights-container {

background: rgba(255, 255, 255, 0.95);

backdrop-filter: blur(10px);

border-radius: 20px;

padding: 2rem;

box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);

border: 1px solid rgba(255, 255, 255, 0.2);

}

.flights-container h3 {

color: #2c3e50;

font-size: 2rem;

margin-bottom: 2rem;

text-align: center;

font-weight: 600;

}

.flights-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(350px, 1fr));

gap: 2rem;

}

.flight-card {

background: white;

border-radius: 15px;

padding: 1.5rem;

box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);

transition: all 0.3s ease;

border: 1px solid #e8f4fd;

position: relative;

overflow: hidden;

}

.flight-card::before {

content: '';

position: absolute;

top: 0;

left: 0;

right: 0;

height: 4px;

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

}

.flight-card:hover {

transform: translateY(-5px);

box-shadow: 0 8px 30px rgba(0, 0, 0, 0.15);

}

.flight-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1.5rem;

}

.flight-header h4 {

color: #2c3e50;

font-size: 1.3rem;

font-weight: 600;

}

.flight-number {

background: #e8f4fd;

color: #3498db;

padding: 0.3rem 0.8rem;

border-radius: 20px;

font-size: 0.9rem;

font-weight: 600;

}

.flight-route {

display: flex;

align-items: center;

justify-content: space-between;

margin-bottom: 1.5rem;

padding: 1rem;

background: #f8f9fa;

border-radius: 10px;

}

.route-info {

display: flex;

flex-direction: column;

align-items: center;

text-align: center;

}

.city {

font-size: 1.1rem;

font-weight: 600;

color: #2c3e50;

margin-bottom: 0.3rem;

}

.time {

font-size: 0.9rem;

color: #7f8c8d;

}

.route-arrow {

font-size: 1.5rem;

color: #3498db;

font-weight: bold;

}

.flight-details {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1rem;

}

.price {

font-size: 1.5rem;

font-weight: 700;

color: #27ae60;

}

.seats {

color: #7f8c8d;

font-size: 0.9rem;

}

.booking-notice {

background: #fff3cd;

border: 1px solid #ffeaa7;

border-radius: 8px;

padding: 0.8rem;

text-align: center;

}

.booking-notice p {

color: #856404;

font-size: 0.9rem;

margin: 0;

font-weight: 500;

}

/\* Responsive Design \*/

@media (max-width: 768px) {

.guest-header h2 {

font-size: 2rem;

}

.guest-header p {

font-size: 1rem;

}

.flights-grid {

grid-template-columns: 1fr;

gap: 1.5rem;

}

.flight-card {

padding: 1rem;

}

.flight-route {

flex-direction: column;

gap: 1rem;

}

.route-arrow {

transform: rotate(90deg);

}

}

@media (max-width: 480px) {

.guest-header {

padding: 1.5rem;

}

.guest-header h2 {

font-size: 1.5rem;

}

.flights-container {

padding: 1rem;

}

.flights-container h3 {

font-size: 1.5rem;

}

}

**-UserPage.css**

.user-page {

max-width: 100%;

}

.user-header {

background: rgba(255, 255, 255, 0.95);

backdrop-filter: blur(10px);

border-radius: 20px;

padding: 2rem;

margin-bottom: 2rem;

text-align: center;

box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);

border: 1px solid rgba(255, 255, 255, 0.2);

position: relative;

}

.user-header h2 {

color: #2c3e50;

font-size: 2.5rem;

margin-bottom: 1rem;

font-weight: 700;

}

.user-header p {

color: #7f8c8d;

font-size: 1.2rem;

margin-bottom: 2rem;

}

.logout-btn {

background: linear-gradient(135deg, #e74c3c 0%, #c0392b 100%);

color: white;

border: none;

padding: 0.8rem 1.5rem;

font-size: 1rem;

font-weight: 600;

border-radius: 50px;

cursor: pointer;

transition: all 0.3s ease;

box-shadow: 0 4px 15px rgba(231, 76, 60, 0.4);

}

.logout-btn:hover {

transform: translateY(-2px);

box-shadow: 0 6px 20px rgba(231, 76, 60, 0.6);

}

.content-sections {

display: grid;

grid-template-columns: 2fr 1fr;

gap: 2rem;

}

.flights-section, .bookings-section {

background: rgba(255, 255, 255, 0.95);

backdrop-filter: blur(10px);

border-radius: 20px;

padding: 2rem;

box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);

border: 1px solid rgba(255, 255, 255, 0.2);

}

.flights-section h3, .bookings-section h3 {

color: #2c3e50;

font-size: 1.8rem;

margin-bottom: 2rem;

font-weight: 600;

}

.flights-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 1.5rem;

}

.flight-card {

background: white;

border-radius: 15px;

padding: 1.5rem;

box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);

transition: all 0.3s ease;

border: 1px solid #e8f4fd;

position: relative;

overflow: hidden;

}

.flight-card::before {

content: '';

position: absolute;

top: 0;

left: 0;

right: 0;

height: 4px;

background: linear-gradient(135deg, #27ae60 0%, #2ecc71 100%);

}

.flight-card:hover {

transform: translateY(-5px);

box-shadow: 0 8px 30px rgba(0, 0, 0, 0.15);

}

.flight-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1.5rem;

}

.flight-header h4 {

color: #2c3e50;

font-size: 1.3rem;

font-weight: 600;

}

.flight-number {

background: #e8f4fd;

color: #3498db;

padding: 0.3rem 0.8rem;

border-radius: 20px;

font-size: 0.9rem;

font-weight: 600;

}

.flight-route {

display: flex;

align-items: center;

justify-content: space-between;

margin-bottom: 1.5rem;

padding: 1rem;

background: #f8f9fa;

border-radius: 10px;

}

.route-info {

display: flex;

flex-direction: column;

align-items: center;

text-align: center;

}

.city {

font-size: 1.1rem;

font-weight: 600;

color: #2c3e50;

margin-bottom: 0.3rem;

}

.time {

font-size: 0.9rem;

color: #7f8c8d;

}

.route-arrow {

font-size: 1.5rem;

color: #3498db;

font-weight: bold;

}

.flight-details {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1rem;

}

.price {

font-size: 1.5rem;

font-weight: 700;

color: #27ae60;

}

.seats {

color: #7f8c8d;

font-size: 0.9rem;

}

.book-flight-btn {

width: 100%;

background: linear-gradient(135deg, #27ae60 0%, #2ecc71 100%);

color: white;

border: none;

padding: 1rem;

font-size: 1rem;

font-weight: 600;

border-radius: 10px;

cursor: pointer;

transition: all 0.3s ease;

box-shadow: 0 4px 15px rgba(39, 174, 96, 0.4);

}

.book-flight-btn:hover {

transform: translateY(-2px);

box-shadow: 0 6px 20px rgba(39, 174, 96, 0.6);

}

/\* Booking Modal \*/

.booking-modal {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.5);

display: flex;

align-items: center;

justify-content: center;

z-index: 1000;

padding: 1rem;

}

.booking-form {

background: white;

border-radius: 20px;

padding: 2rem;

max-width: 500px;

width: 100%;

box-shadow: 0 20px 60px rgba(0, 0, 0, 0.3);

max-height: 90vh;

overflow-y: auto;

}

.booking-form h3 {

color: #2c3e50;

font-size: 1.5rem;

margin-bottom: 2rem;

text-align: center;

font-weight: 600;

}

.form-group {

margin-bottom: 1.5rem;

}

.form-group label {

display: block;

margin-bottom: 0.5rem;

color: #2c3e50;

font-weight: 600;

}

.form-group input {

width: 100%;

padding: 0.8rem;

border: 2px solid #e8f4fd;

border-radius: 10px;

font-size: 1rem;

transition: border-color 0.3s ease;

}

.form-group input:focus {

outline: none;

border-color: #3498db;

}

.form-actions {

display: flex;

gap: 1rem;

margin-top: 2rem;

}

.book-btn, .cancel-btn {

flex: 1;

padding: 1rem;

border: none;

border-radius: 10px;

font-size: 1rem;

font-weight: 600;

cursor: pointer;

transition: all 0.3s ease;

}

.book-btn {

background: linear-gradient(135deg, #27ae60 0%, #2ecc71 100%);

color: white;

box-shadow: 0 4px 15px rgba(39, 174, 96, 0.4);

}

.book-btn:hover {

transform: translateY(-2px);

box-shadow: 0 6px 20px rgba(39, 174, 96, 0.6);

}

.cancel-btn {

background: #e74c3c;

color: white;

box-shadow: 0 4px 15px rgba(231, 76, 60, 0.4);

}

.cancel-btn:hover {

transform: translateY(-2px);

box-shadow: 0 6px 20px rgba(231, 76, 60, 0.6);

}

/\* Bookings Section \*/

.no-bookings {

text-align: center;

color: #7f8c8d;

font-style: italic;

padding: 2rem;

}

.bookings-list {

display: flex;

flex-direction: column;

gap: 1rem;

}

.booking-card {

background: white;

border-radius: 15px;

padding: 1.5rem;

box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);

border: 1px solid #e8f4fd;

position: relative;

}

.booking-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1rem;

padding-bottom: 1rem;

border-bottom: 1px solid #e8f4fd;

}

.booking-header h4 {

color: #2c3e50;

font-size: 1.2rem;

font-weight: 600;

}

.booking-id {

background: #e8f4fd;

color: #3498db;

padding: 0.2rem 0.6rem;

border-radius: 15px;

font-size: 0.8rem;

font-weight: 600;

}

.booking-details p {

margin-bottom: 0.5rem;

color: #2c3e50;

font-size: 0.9rem;

}

.booking-details strong {

color: #2c3e50;

}

.cancel-booking-btn {

width: 100%;

background: #e74c3c;

color: white;

border: none;

padding: 0.8rem;

font-size: 0.9rem;

font-weight: 600;

border-radius: 8px;

cursor: pointer;

transition: all 0.3s ease;

margin-top: 1rem;

}

.cancel-booking-btn:hover {

background: #c0392b;

transform: translateY(-1px);

}

/\* Responsive Design \*/

@media (max-width: 1024px) {

.content-sections {

grid-template-columns: 1fr;

gap: 1.5rem;

}

}

@media (max-width: 768px) {

.user-header h2 {

font-size: 2rem;

}

.user-header p {

font-size: 1rem;

}

.flights-grid {

grid-template-columns: 1fr;

gap: 1rem;

}

.flight-card {

padding: 1rem;

}

.flight-route {

flex-direction: column;

gap: 1rem;

}

.route-arrow {

transform: rotate(90deg);

}

.booking-form {

padding: 1.5rem;

}

.form-actions {

flex-direction: column;

}

}

@media (max-width: 480px) {

.user-header {

padding: 1.5rem;

}

.user-header h2 {

font-size: 1.5rem;

}

.flights-section, .bookings-section {

padding: 1rem;

}

.flights-section h3, .bookings-section h3 {

font-size: 1.5rem;

}

}

**-Guestpage.js**

import React from 'react';

import './GuestPage.css';

const GuestPage = ({ onLogin }) => {

const flights = [

{

id: 1,

airline: 'Air India',

flightNumber: 'AI-101',

from: 'Mumbai',

to: 'Delhi',

departure: '10:00 AM',

arrival: '12:00 PM',

price: 4500,

seats: 45

},

{

id: 2,

airline: 'IndiGo',

flightNumber: '6E-202',

from: 'Delhi',

to: 'Bangalore',

departure: '02:30 PM',

arrival: '05:00 PM',

price: 3800,

seats: 32

},

{

id: 3,

airline: 'SpiceJet',

flightNumber: 'SG-303',

from: 'Chennai',

to: 'Kolkata',

departure: '08:15 AM',

arrival: '10:45 AM',

price: 3200,

seats: 28

},

{

id: 4,

airline: 'Vistara',

flightNumber: 'UK-404',

from: 'Hyderabad',

to: 'Pune',

departure: '11:45 AM',

arrival: '01:15 PM',

price: 2800,

seats: 38

},

{

id: 5,

airline: 'GoAir',

flightNumber: 'G8-505',

from: 'Ahmedabad',

to: 'Jaipur',

departure: '03:20 PM',

arrival: '04:50 PM',

price: 2200,

seats: 25

}

];

return (

<div className="guest-page">

<div className="guest-header">

<h2>Welcome to Flight Ticket Booking</h2>

<p>Browse available flights below. Please login to book tickets.</p>

<button className="login-btn" onClick={onLogin}>

Login to Book Tickets

</button>

</div>

<div className="flights-container">

<h3>Available Flights</h3>

<div className="flights-grid">

{flights.map((flight) => (

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

<div className="flight-header">

<h4>{flight.airline}</h4>

<span className="flight-number">{flight.flightNumber}</span>

</div>

<div className="flight-route">

<div className="route-info">

<span className="city">{flight.from}</span>

<span className="time">{flight.departure}</span>

</div>

<div className="route-arrow">→</div>

<div className="route-info">

<span className="city">{flight.to}</span>

<span className="time">{flight.arrival}</span>

</div>

</div>

<div className="flight-details">

<div className="price">₹{flight.price}</div>

<div className="seats">Seats: {flight.seats}</div>

</div>

<div className="booking-notice">

<p>Login required to book this flight</p>

</div>

</div>

))}

</div>

</div>

</div>

);

};

export default GuestPage;

**-UserPage.js**

import React, { useState } from 'react';

import './UserPage.css';

const UserPage = ({ onLogout }) => {

const [bookedTickets, setBookedTickets] = useState([]);

const [showBookingForm, setShowBookingForm] = useState(false);

const [selectedFlight, setSelectedFlight] = useState(null);

const [bookingForm, setBookingForm] = useState({

passengerName: '',

email: '',

phone: '',

seats: 1

});

const flights = [

{

id: 1,

airline: 'Air India',

flightNumber: 'AI-101',

from: 'Mumbai',

to: 'Delhi',

departure: '10:00 AM',

arrival: '12:00 PM',

price: 4500,

seats: 45

},

{

id: 2,

airline: 'IndiGo',

flightNumber: '6E-202',

from: 'Delhi',

to: 'Bangalore',

departure: '02:30 PM',

arrival: '05:00 PM',

price: 3800,

seats: 32

},

{

id: 3,

airline: 'SpiceJet',

flightNumber: 'SG-303',

from: 'Chennai',

to: 'Kolkata',

departure: '08:15 AM',

arrival: '10:45 AM',

price: 3200,

seats: 28

},

{

id: 4,

airline: 'Vistara',

flightNumber: 'UK-404',

from: 'Hyderabad',

to: 'Pune',

departure: '11:45 AM',

arrival: '01:15 PM',

price: 2800,

seats: 38

},

{

id: 5,

airline: 'GoAir',

flightNumber: 'G8-505',

from: 'Ahmedabad',

to: 'Jaipur',

departure: '03:20 PM',

arrival: '04:50 PM',

price: 2200,

seats: 25

}

];

const handleBookFlight = (flight) => {

setSelectedFlight(flight);

setShowBookingForm(true);

};

const handleBookingSubmit = (e) => {

e.preventDefault();

if (bookingForm.passengerName && bookingForm.email && bookingForm.phone) {

const newBooking = {

id: Date.now(),

flight: selectedFlight,

...bookingForm,

totalPrice: selectedFlight.price \* bookingForm.seats,

bookingDate: new Date().toLocaleDateString(),

bookingTime: new Date().toLocaleTimeString()

};

setBookedTickets([...bookedTickets, newBooking]);

setShowBookingForm(false);

setSelectedFlight(null);

setBookingForm({

passengerName: '',

email: '',

phone: '',

seats: 1

});

alert('Ticket booked successfully!');

} else {

alert('Please fill all required fields');

}

};

const handleInputChange = (e) => {

setBookingForm({

...bookingForm,

[e.target.name]: e.target.value

});

};

const cancelBooking = (bookingId) => {

setBookedTickets(bookedTickets.filter(booking => booking.id !== bookingId));

};

return (

<div className="user-page">

<div className="user-header">

<h2>Welcome back! 🎉</h2>

<p>You can now book tickets for your preferred flights.</p>

<button className="logout-btn" onClick={onLogout}>

Logout

</button>

</div>

{showBookingForm && selectedFlight && (

<div className="booking-modal">

<div className="booking-form">

<h3>Book Flight: {selectedFlight.airline} {selectedFlight.flightNumber}</h3>

<form onSubmit={handleBookingSubmit}>

<div className="form-group">

<label>Passenger Name:</label>

<input

type="text"

name="passengerName"

value={bookingForm.passengerName}

onChange={handleInputChange}

required

/>

</div>

<div className="form-group">

<label>Email:</label>

<input

type="email"

name="email"

value={bookingForm.email}

onChange={handleInputChange}

required

/>

</div>

<div className="form-group">

<label>Phone:</label>

<input

type="tel"

name="phone"

value={bookingForm.phone}

onChange={handleInputChange}

required

/>

</div>

<div className="form-group">

<label>Number of Seats:</label>

<input

type="number"

name="seats"

min="1"

max="5"

value={bookingForm.seats}

onChange={handleInputChange}

required

/>

</div>

<div className="form-actions">

<button type="submit" className="book-btn">Book Ticket</button>

<button

type="button"

className="cancel-btn"

onClick={() => setShowBookingForm(false)}

>

Cancel

</button>

</div>

</form>

</div>

</div>

)}

<div className="content-sections">

<div className="flights-section">

<h3>Available Flights</h3>

<div className="flights-grid">

{flights.map((flight) => (

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

<div className="flight-header">

<h4>{flight.airline}</h4>

<span className="flight-number">{flight.flightNumber}</span>

</div>

<div className="flight-route">

<div className="route-info">

<span className="city">{flight.from}</span>

<span className="time">{flight.departure}</span>

</div>

<div className="route-arrow">→</div>

<div className="route-info">

<span className="city">{flight.to}</span>

<span className="time">{flight.arrival}</span>

</div>

</div>

<div className="flight-details">

<div className="price">₹{flight.price}</div>

<div className="seats">Seats: {flight.seats}</div>

</div>

<button

className="book-flight-btn"

onClick={() => handleBookFlight(flight)}

>

Book Now

</button>

</div>

))}

</div>

</div>

<div className="bookings-section">

<h3>My Bookings ({bookedTickets.length})</h3>

{bookedTickets.length === 0 ? (

<p className="no-bookings">No bookings yet. Book your first flight!</p>

) : (

<div className="bookings-list">

{bookedTickets.map((booking) => (

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

<div className="booking-header">

<h4>{booking.flight.airline} {booking.flight.flightNumber}</h4>

<span className="booking-id">#{booking.id}</span>

</div>

<div className="booking-details">

<p><strong>Passenger:</strong> {booking.passengerName}</p>

<p><strong>Route:</strong> {booking.flight.from} → {booking.flight.to}</p>

<p><strong>Date:</strong> {booking.flight.departure} - {booking.flight.arrival}</p>

<p><strong>Seats:</strong> {booking.seats}</p>

<p><strong>Total:</strong> ₹{booking.totalPrice}</p>

<p><strong>Booked on:</strong> {booking.bookingDate} at {booking.bookingTime}</p>

</div>

<button

className="cancel-booking-btn"

onClick={() => cancelBooking(booking.id)}

>

Cancel Booking

</button>

</div>

))}

</div>

)}

</div>

</div>

</div>

);

};

export default UserPage;

**-App.js**

import React, { useState } from 'react';

import './App.css';

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 className="App">

<header className="App-header">

<h1>✈️ Flight Ticket Booking</h1>

</header>

<main>

{isLoggedIn ? (

<UserPage onLogout={handleLogout} />

) : (

<GuestPage onLogin={handleLogin} />

)}

</main>

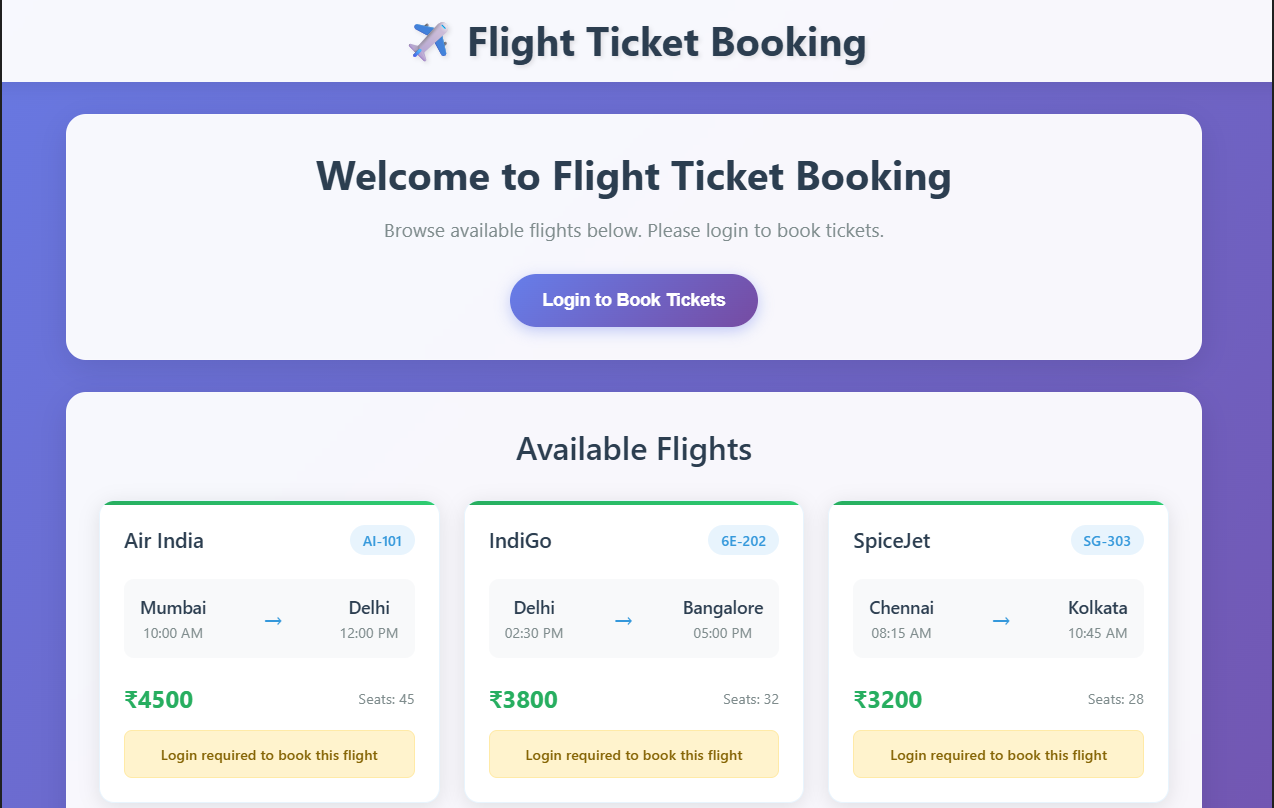
</div>

);

}

export default App;

OUTPUT



**13.**

**BlogDetails.js**

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

const BlogDetails = () => {

const [blogs, setBlogs] = useState([]);

const [userRole, setUserRole] = useState('reader'); // reader, author, admin

const [isEditing, setIsEditing] = useState(false);

const [selectedBlog, setSelectedBlog] = useState(null);

const [showDrafts, setShowDrafts] = useState(false);

const sampleBlogs = [

{

id: 1,

title: 'React Learning',

author: 'Stephen Biz',

content: 'Welcome to learning React!',

status: 'published',

date: '2024-01-15',

views: 1250

},

{

id: 2,

title: 'Installation',

author: 'Schewzdenier',

content: 'You can install React from npm.',

status: 'published',

date: '2024-01-20',

views: 890

},

{

id: 3,

title: 'Advanced React Patterns',

author: 'Stephen Biz',

content: 'Exploring advanced patterns in React development.',

status: 'draft',

date: '2024-01-25',

views: 0

},

{

id: 4,

title: 'State Management',

author: 'Schewzdenier',

content: 'Understanding state management in React applications.',

status: 'published',

date: '2024-01-30',

views: 2100

}

];

useEffect(() => {

// Simulate API call

setTimeout(() => {

setBlogs(sampleBlogs);

}, 1000);

}, [sampleBlogs]);

// Method 1: Early return pattern

const renderBlogList = () => {

if (blogs.length === 0) {

return <div className="loading">Loading blogs...</div>;

}

const filteredBlogs = showDrafts

? blogs.filter(blog => blog.status === 'draft')

: blogs.filter(blog => blog.status === 'published');

if (filteredBlogs.length === 0) {

return <p>No {showDrafts ? 'drafts' : 'published blogs'} available</p>;

}

return (

<div className="blog-list">

{filteredBlogs.map(blog => (

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

<h3>{blog.title}</h3>

<p><strong>Author:</strong> {blog.author}</p>

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

<p><strong>Date:</strong> {blog.date}</p>

<p><strong>Views:</strong> {blog.views}</p>

{/\* Method 2: Conditional rendering with object mapping for status badges \*/}

{renderStatusBadge(blog.status)}

{/\* Method 3: Conditional rendering based on user role \*/}

{userRole === 'admin' && (

<div className="admin-controls">

<button onClick={() => handleEdit(blog)}>Edit</button>

<button onClick={() => handleDelete(blog.id)}>Delete</button>

</div>

)}

{/\* Method 4: Conditional rendering for authors \*/}

{userRole === 'author' && blog.author === 'Stephen Biz' && (

<button onClick={() => handleEdit(blog)}>Edit My Blog</button>

)}

</div>

))}

</div>

);

};

// Method 5: Switch statement for conditional rendering

const renderUserControls = () => {

switch (userRole) {

case 'admin':

return (

<div className="admin-panel">

<h3>Admin Panel</h3>

<button onClick={() => setShowDrafts(!showDrafts)}>

{showDrafts ? 'Show Published' : 'Show Drafts'}

</button>

<button onClick={() => setUserRole('reader')}>Switch to Reader</button>

<button onClick={() => setUserRole('author')}>Switch to Author</button>

</div>

);

case 'author':

return (

<div className="author-panel">

<h3>Author Panel</h3>

<button onClick={() => setShowDrafts(!showDrafts)}>

{showDrafts ? 'Show Published' : 'Show My Drafts'}

</button>

<button onClick={() => setUserRole('reader')}>Switch to Reader</button>

</div>

);

case 'reader':

default:

return (

<div className="reader-panel">

<h3>Reader Panel</h3>

<button onClick={() => setUserRole('author')}>Switch to Author</button>

<button onClick={() => setUserRole('admin')}>Switch to Admin</button>

</div>

);

}

};

// Method 6: Conditional rendering with object mapping

const renderStatusBadge = (status) => {

const statusConfig = {

published: { className: 'status-published', text: 'Published' },

draft: { className: 'status-draft', text: 'Draft' },

pending: { className: 'status-pending', text: 'Pending Review' }

};

const config = statusConfig[status] || { className: 'status-unknown', text: 'Unknown' };

return <span className={config.className}>{config.text}</span>;

};

// Method 7: Conditional rendering with function composition

const renderBlogStats = () => {

const totalBlogs = blogs.length;

const publishedBlogs = blogs.filter(blog => blog.status === 'published').length;

const draftBlogs = blogs.filter(blog => blog.status === 'draft').length;

const totalViews = blogs.reduce((sum, blog) => sum + blog.views, 0);

return (

<div className="blog-stats">

<h3>Blog Statistics</h3>

<p>Total Blogs: {totalBlogs}</p>

<p>Published: {publishedBlogs}</p>

<p>Drafts: {draftBlogs}</p>

<p>Total Views: {totalViews}</p>

{/\* Method 8: Conditional rendering with logical operators and null \*/}

{totalViews > 2000 && <p className="high-views">High traffic blog!</p>}

{publishedBlogs === 0 && <p className="no-published">No published blogs yet</p>}

{draftBlogs > publishedBlogs && <p className="many-drafts">Many drafts pending</p>}

</div>

);

};

// Method 9: Conditional rendering with array methods

const renderTopAuthors = () => {

const authorStats = blogs.reduce((acc, blog) => {

acc[blog.author] = (acc[blog.author] || 0) + 1;

return acc;

}, {});

const topAuthors = Object.entries(authorStats)

.sort(([,a], [,b]) => b - a)

.slice(0, 3);

return topAuthors.length > 0 ? (

<div className="top-authors">

<h3>Top Authors</h3>

{topAuthors.map(([author, count]) => (

<p key={author}>{author}: {count} blogs</p>

))}

</div>

) : null;

};

const handleEdit = (blog) => {

setSelectedBlog(blog);

setIsEditing(true);

};

const handleDelete = (blogId) => {

setBlogs(blogs.filter(blog => blog.id !== blogId));

};

return (

<div className="blog-details">

<h2>Blog Details</h2>

{renderUserControls()}

<div className="blog-content">

{renderBlogList()}

</div>

<div className="blog-sidebar">

{renderBlogStats()}

{renderTopAuthors()}

</div>

{/\* Method 10: Conditional rendering with state \*/}

{isEditing && selectedBlog && (

<div className="edit-modal">

<h3>Editing: {selectedBlog.title}</h3>

<button onClick={() => setIsEditing(false)}>Close</button>

</div>

)}

</div>

);

};

export default BlogDetails;

**BookDetails.js**

import React, { useState } from 'react';

const BookDetails = () => {

const [showBooks, setShowBooks] = useState(true);

const [selectedCategory, setSelectedCategory] = useState('all');

const [isLoading, setIsLoading] = useState(false);

const books = [

{ id: 1, title: 'Master React', price: 670, category: 'programming', inStock: true },

{ id: 2, title: 'Deep Dive into Angular 11', price: 800, category: 'programming', inStock: true },

{ id: 3, title: 'Mongo Essentials', price: 450, category: 'database', inStock: false },

{ id: 4, title: 'JavaScript Fundamentals', price: 550, category: 'programming', inStock: true },

{ id: 5, title: 'Node.js Complete Guide', price: 720, category: 'backend', inStock: true }

];

const categories = ['all', 'programming', 'database', 'backend'];

// Method 1: Conditional rendering with ternary operator

const renderBookList = () => {

return showBooks ? (

<div className="book-list">

{books

.filter(book => selectedCategory === 'all' || book.category === selectedCategory)

.map(book => (

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

<h3>{book.title}</h3>

<p>Price: ${book.price}</p>

{/\* Method 2: Logical AND operator for conditional rendering \*/}

{book.inStock && <span className="in-stock">In Stock</span>}

{/\* Method 3: Ternary operator for conditional styling \*/}

<span className={book.inStock ? 'status-available' : 'status-unavailable'}>

{book.inStock ? 'Available' : 'Out of Stock'}

</span>

</div>

))}

</div>

) : (

<p>Books are hidden</p>

);

};

// Method 4: Conditional rendering with if-else logic

const renderCategoryFilter = () => {

if (categories.length === 0) {

return <p>No categories available</p>;

} else if (categories.length === 1) {

return <p>Only one category available</p>;

} else {

return (

<div className="category-filter">

<label>Filter by Category: </label>

<select

value={selectedCategory}

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

>

{categories.map(category => (

<option key={category} value={category}>

{category.charAt(0).toUpperCase() + category.slice(1)}

</option>

))}

</select>

</div>

);

}

};

// Method 5: Conditional rendering based on loading state

const renderContent = () => {

if (isLoading) {

return <div className="loading">Loading books...</div>;

}

return (

<>

{renderCategoryFilter()}

{renderBookList()}

</>

);

};

// Method 6: Conditional rendering with multiple conditions

const renderActionButtons = () => {

const hasBooks = books.length > 0;

const hasFilteredBooks = books.filter(book =>

selectedCategory === 'all' || book.category === selectedCategory

).length > 0;

return (

<div className="action-buttons">

<button onClick={() => setShowBooks(!showBooks)}>

{showBooks ? 'Hide' : 'Show'} Books

</button>

{/\* Method 7: Multiple conditions with logical operators \*/}

{hasBooks && hasFilteredBooks && (

<button onClick={() => setIsLoading(true)}>

Refresh Books

</button>

)}

{/\* Method 8: Conditional rendering with null \*/}

{!hasBooks ? (

<p>No books available</p>

) : !hasFilteredBooks ? (

<p>No books in selected category</p>

) : null}

</div>

);

};

return (

<div className="book-details">

<h2>Book Details</h2>

{renderContent()}

{renderActionButtons()}

</div>

);

};

export default BookDetails;

**CourseDetails.js**

import React, { useState, useEffect, useMemo } from 'react';

const CourseDetails = () => {

const [courses, setCourses] = useState([]);

const [enrolledCourses, setEnrolledCourses] = useState([]);

const [userLevel, setUserLevel] = useState('beginner'); // beginner, intermediate, advanced

const [showCompleted, setShowCompleted] = useState(false);

const [error, setError] = useState(null);

const [isLoading, setIsLoading] = useState(true);

const sampleCourses = [

{

id: 1,

name: 'Angular',

date: '4/5/2021',

level: 'intermediate',

duration: '8 weeks',

instructor: 'John Doe',

price: 299,

maxStudents: 50,

enrolledStudents: 35,

isCompleted: false

},

{

id: 2,

name: 'React',

date: '6/3/2021',

level: 'beginner',

duration: '10 weeks',

instructor: 'Jane Smith',

price: 399,

maxStudents: 40,

enrolledStudents: 40,

isCompleted: true

},

{

id: 3,

name: 'Vue.js Advanced',

date: '8/15/2021',

level: 'advanced',

duration: '6 weeks',

instructor: 'Mike Johnson',

price: 499,

maxStudents: 30,

enrolledStudents: 15,

isCompleted: false

},

{

id: 4,

name: 'JavaScript Fundamentals',

date: '9/1/2021',

level: 'beginner',

duration: '12 weeks',

instructor: 'Sarah Wilson',

price: 199,

maxStudents: 60,

enrolledStudents: 45,

isCompleted: false

}

];

useEffect(() => {

// Simulate API call with error handling

const fetchCourses = async () => {

try {

setIsLoading(true);

// Simulate network delay

await new Promise(resolve => setTimeout(resolve, 1500));

// Simulate random error (10% chance)

if (Math.random() < 0.1) {

throw new Error('Failed to fetch courses');

}

setCourses(sampleCourses);

setEnrolledCourses([1, 3]); // User is enrolled in courses with IDs 1 and 3

} catch (err) {

setError(err.message);

} finally {

setIsLoading(false);

}

};

fetchCourses();

}, []);

// Method 1: Custom hook-like pattern for conditional rendering - moved to component level

const courseFilters = useMemo(() => {

const filteredCourses = courses.filter(course => {

const levelMatch = userLevel === 'all' || course.level === userLevel;

const completionMatch = showCompleted ? course.isCompleted : !course.isCompleted;

return levelMatch && completionMatch;

});

return {

filteredCourses,

hasCourses: filteredCourses.length > 0,

totalCourses: courses.length

};

}, [courses, userLevel, showCompleted]);

// Method 2: Computed properties at component level

const courseStats = useMemo(() => {

const totalCourses = courses.length;

const availableCourses = courses.filter(course =>

course.enrolledStudents < course.maxStudents

).length;

const completedCourses = courses.filter(course => course.isCompleted).length;

const totalEnrollment = courses.reduce((sum, course) => sum + course.enrolledStudents, 0);

return { totalCourses, availableCourses, completedCourses, totalEnrollment };

}, [courses]);

// Method 3: Level-based recommendations at component level

const recommendations = useMemo(() => {

const userCourses = courses.filter(course => enrolledCourses.includes(course.id));

const userLevels = userCourses.map(course => course.level);

if (userLevels.includes('advanced')) {

return courses.filter(course => course.level === 'advanced' && !enrolledCourses.includes(course.id));

} else if (userLevels.includes('intermediate')) {

return courses.filter(course => course.level === 'intermediate' && !enrolledCourses.includes(course.id));

} else {

return courses.filter(course => course.level === 'beginner' && !enrolledCourses.includes(course.id));

}

}, [courses, enrolledCourses]);

// Method 4: Conditional rendering with error handling

const renderErrorBoundary = () => {

if (error) {

return (

<div className="error-container">

<h3>Error Loading Courses</h3>

<p>{error}</p>

<button onClick={() => window.location.reload()}>Retry</button>

</div>

);

}

return null;

};

// Method 5: Conditional rendering with loading states

const renderLoadingState = () => {

if (isLoading) {

return (

<div className="loading-container">

<div className="spinner"></div>

<p>Loading courses...</p>

</div>

);

}

return null;

};

// Method 6: Conditional rendering with enrollment status

const renderCourseCard = (course) => {

const isEnrolled = enrolledCourses.includes(course.id);

const isFull = course.enrolledStudents >= course.maxStudents;

const enrollmentPercentage = (course.enrolledStudents / course.maxStudents) \* 100;

return (

<div key={course.id} className={`course-card ${isEnrolled ? 'enrolled' : ''}`}>

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

<p><strong>Date:</strong> {course.date}</p>

<p><strong>Level:</strong> {course.level}</p>

<p><strong>Duration:</strong> {course.duration}</p>

<p><strong>Instructor:</strong> {course.instructor}</p>

<p><strong>Price:</strong> ${course.price}</p>

{/\* Method 7: Conditional rendering based on enrollment status \*/}

{isEnrolled ? (

<div className="enrollment-status">

<span className="enrolled-badge">Enrolled</span>

{course.isCompleted && <span className="completed-badge">Completed</span>}

</div>

) : (

<div className="enrollment-actions">

{isFull ? (

<span className="full-badge">Course Full</span>

) : (

<button onClick={() => handleEnroll(course.id)}>Enroll Now</button>

)}

</div>

)}

{/\* Method 8: Conditional rendering with progress indicators \*/}

<div className="enrollment-progress">

<p>Enrollment: {course.enrolledStudents}/{course.maxStudents}</p>

<div className="progress-bar">

<div

className="progress-fill"

style={{ width: `${enrollmentPercentage}%` }}

></div>

</div>

{enrollmentPercentage > 80 && <p className="almost-full">Almost Full!</p>}

</div>

</div>

);

};

// Method 9: Conditional rendering with computed properties (now using pre-computed stats)

const renderCourseStats = () => {

return (

<div className="course-stats">

<h3>Course Statistics</h3>

<p>Total Courses: {courseStats.totalCourses}</p>

<p>Available Courses: {courseStats.availableCourses}</p>

<p>Completed Courses: {courseStats.completedCourses}</p>

<p>Total Enrollments: {courseStats.totalEnrollment}</p>

{/\* Method 10: Conditional rendering with multiple conditions \*/}

{courseStats.availableCourses === 0 && <p className="no-availability">No courses available</p>}

{courseStats.completedCourses > courseStats.totalCourses / 2 && <p className="high-completion">High completion rate!</p>}

{courseStats.totalEnrollment > 100 && <p className="popular-platform">Popular learning platform</p>}

</div>

);

};

// Method 11: Conditional rendering with level-based recommendations (now using pre-computed recommendations)

const renderRecommendations = () => {

return recommendations.length > 0 ? (

<div className="recommendations">

<h3>Recommended for You</h3>

{recommendations.slice(0, 3).map(course => (

<div key={course.id} className="recommendation-item">

<h4>{course.name}</h4>

<p>{course.level} level • {course.duration}</p>

</div>

))}

</div>

) : null;

};

// Method 12: Conditional rendering with filters

const renderFilters = () => {

const levels = ['all', 'beginner', 'intermediate', 'advanced'];

return (

<div className="course-filters">

<div className="level-filter">

<label>Filter by Level: </label>

<select value={userLevel} onChange={(e) => setUserLevel(e.target.value)}>

{levels.map(level => (

<option key={level} value={level}>

{level.charAt(0).toUpperCase() + level.slice(1)}

</option>

))}

</select>

</div>

<div className="completion-filter">

<label>

<input

type="checkbox"

checked={showCompleted}

onChange={(e) => setShowCompleted(e.target.checked)}

/>

Show Completed Courses

</label>

</div>

</div>

);

};

const handleEnroll = (courseId) => {

if (!enrolledCourses.includes(courseId)) {

setEnrolledCourses([...enrolledCourses, courseId]);

}

};

const { filteredCourses, hasCourses } = courseFilters;

return (

<div className="course-details">

<h2>Course Details</h2>

{renderErrorBoundary()}

{renderLoadingState()}

{!error && !isLoading && (

<>

{renderFilters()}

{renderCourseStats()}

<div className="courses-container">

{hasCourses ? (

<div className="course-grid">

{filteredCourses.map(renderCourseCard)}

</div>

) : (

<p>No courses match your current filters</p>

)}

</div>

{renderRecommendations()}

</>

)}

</div>

);

};

export default CourseDetails;

**App.css**

/\* Reset and base styles \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

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

line-height: 1.6;

color: #333;

background-color: #f5f5f5;

}

/\* App container \*/

.App {

min-height: 100vh;

transition: all 0.3s ease;

}

.app-light {

background-color: #f5f5f5;

color: #333;

}

.app-dark {

background-color: #1a1a1a;

color: #fff;

}

/\* Header styles \*/

.App-header {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

color: white;

padding: 2rem;

text-align: center;

box-shadow: 0 2px 10px rgba(0,0,0,0.1);

}

.App-header h1 {

font-size: 2.5rem;

font-weight: 300;

margin: 0;

}

/\* Navigation tabs \*/

.tab-navigation {

display: flex;

justify-content: center;

gap: 1rem;

padding: 1rem;

background: white;

box-shadow: 0 2px 5px rgba(0,0,0,0.1);

}

.tab-button {

padding: 0.75rem 1.5rem;

border: none;

border-radius: 25px;

background: #f0f0f0;

color: #666;

cursor: pointer;

transition: all 0.3s ease;

font-weight: 500;

}

.tab-button:hover {

background: #e0e0e0;

transform: translateY(-2px);

}

.tab-button.active {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

color: white;

box-shadow: 0 4px 15px rgba(102, 126, 234, 0.4);

}

/\* App controls \*/

.app-controls {

display: flex;

justify-content: center;

gap: 1rem;

padding: 1rem;

background: white;

border-bottom: 1px solid #eee;

}

.app-controls button {

padding: 0.5rem 1rem;

border: 1px solid #ddd;

border-radius: 5px;

background: white;

cursor: pointer;

transition: all 0.3s ease;

}

.app-controls button:hover {

background: #f0f0f0;

border-color: #999;

}

/\* Main content layout \*/

.App-main {

padding: 2rem;

max-width: 1400px;

margin: 0 auto;

}

.three-column-layout {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

gap: 2rem;

margin-bottom: 2rem;

}

.single-column-layout {

max-width: 800px;

margin: 0 auto;

}

.column {

background: white;

border-radius: 10px;

padding: 1.5rem;

box-shadow: 0 4px 15px rgba(0,0,0,0.1);

transition: transform 0.3s ease;

}

.column:hover {

transform: translateY(-5px);

}

/\* Component headers \*/

.book-details h2,

.blog-details h2,

.course-details h2 {

color: #667eea;

margin-bottom: 1.5rem;

font-size: 1.8rem;

border-bottom: 2px solid #667eea;

padding-bottom: 0.5rem;

}

/\* Book Details Styles \*/

.book-list {

display: flex;

flex-direction: column;

gap: 1rem;

}

.book-item {

border: 1px solid #eee;

border-radius: 8px;

padding: 1rem;

background: #fafafa;

transition: all 0.3s ease;

}

.book-item:hover {

border-color: #667eea;

box-shadow: 0 2px 10px rgba(102, 126, 234, 0.2);

}

.book-item h3 {

color: #333;

margin-bottom: 0.5rem;

}

.in-stock {

background: #4caf50;

color: white;

padding: 0.25rem 0.5rem;

border-radius: 12px;

font-size: 0.8rem;

margin-left: 0.5rem;

}

.status-available {

color: #4caf50;

font-weight: bold;

}

.status-unavailable {

color: #f44336;

font-weight: bold;

}

.category-filter {

margin-bottom: 1rem;

}

.category-filter select {

padding: 0.5rem;

border: 1px solid #ddd;

border-radius: 5px;

margin-left: 0.5rem;

}

.action-buttons {

display: flex;

gap: 0.5rem;

margin-top: 1rem;

}

.action-buttons button {

padding: 0.5rem 1rem;

border: none;

border-radius: 5px;

background: #667eea;

color: white;

cursor: pointer;

transition: all 0.3s ease;

}

.action-buttons button:hover {

background: #5a6fd8;

}

/\* Blog Details Styles \*/

.blog-list {

display: flex;

flex-direction: column;

gap: 1rem;

}

.blog-item {

border: 1px solid #eee;

border-radius: 8px;

padding: 1rem;

background: #fafafa;

transition: all 0.3s ease;

}

.blog-item:hover {

border-color: #667eea;

box-shadow: 0 2px 10px rgba(102, 126, 234, 0.2);

}

.admin-controls,

.author-panel,

.reader-panel {

background: #f8f9fa;

padding: 1rem;

border-radius: 8px;

margin-bottom: 1rem;

}

.admin-controls button,

.author-panel button,

.reader-panel button {

margin-right: 0.5rem;

padding: 0.5rem 1rem;

border: none;

border-radius: 5px;

background: #667eea;

color: white;

cursor: pointer;

transition: all 0.3s ease;

}

.admin-controls button:hover,

.author-panel button:hover,

.reader-panel button:hover {

background: #5a6fd8;

}

.blog-stats,

.top-authors {

background: #f8f9fa;

padding: 1rem;

border-radius: 8px;

margin-top: 1rem;

}

.status-published {

background: #4caf50;

color: white;

padding: 0.25rem 0.5rem;

border-radius: 12px;

font-size: 0.8rem;

}

.status-draft {

background: #ff9800;

color: white;

padding: 0.25rem 0.5rem;

border-radius: 12px;

font-size: 0.8rem;

}

.status-pending {

background: #2196f3;

color: white;

padding: 0.25rem 0.5rem;

border-radius: 12px;

font-size: 0.8rem;

}

.edit-modal {

position: fixed;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

background: white;

padding: 2rem;

border-radius: 10px;

box-shadow: 0 10px 30px rgba(0,0,0,0.3);

z-index: 1000;

}

/\* Course Details Styles \*/

.course-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 1rem;

}

.course-card {

border: 1px solid #eee;

border-radius: 8px;

padding: 1rem;

background: #fafafa;

transition: all 0.3s ease;

}

.course-card:hover {

border-color: #667eea;

box-shadow: 0 2px 10px rgba(102, 126, 234, 0.2);

}

.course-card.enrolled {

border-color: #4caf50;

background: #f1f8e9;

}

.enrolled-badge {

background: #4caf50;

color: white;

padding: 0.25rem 0.5rem;

border-radius: 12px;

font-size: 0.8rem;

margin-right: 0.5rem;

}

.completed-badge {

background: #2196f3;

color: white;

padding: 0.25rem 0.5rem;

border-radius: 12px;

font-size: 0.8rem;

}

.full-badge {

background: #f44336;

color: white;

padding: 0.25rem 0.5rem;

border-radius: 12px;

font-size: 0.8rem;

}

.enrollment-actions button {

padding: 0.5rem 1rem;

border: none;

border-radius: 5px;

background: #667eea;

color: white;

cursor: pointer;

transition: all 0.3s ease;

}

.enrollment-actions button:hover {

background: #5a6fd8;

}

.progress-bar {

width: 100%;

height: 8px;

background: #eee;

border-radius: 4px;

overflow: hidden;

margin: 0.5rem 0;

}

.progress-fill {

height: 100%;

background: linear-gradient(90deg, #4caf50, #8bc34a);

transition: width 0.3s ease;

}

.almost-full {

color: #ff9800;

font-weight: bold;

font-size: 0.9rem;

}

.course-filters {

display: flex;

gap: 1rem;

margin-bottom: 1rem;

align-items: center;

}

.course-filters select {

padding: 0.5rem;

border: 1px solid #ddd;

border-radius: 5px;

}

.course-filters label {

display: flex;

align-items: center;

gap: 0.5rem;

}

.course-stats {

background: #f8f9fa;

padding: 1rem;

border-radius: 8px;

margin-bottom: 1rem;

}

.recommendations {

background: #e3f2fd;

padding: 1rem;

border-radius: 8px;

margin-top: 1rem;

}

.recommendation-item {

border-left: 3px solid #2196f3;

padding-left: 1rem;

margin: 0.5rem 0;

}

/\* Loading and error states \*/

.loading,

.loading-container {

text-align: center;

padding: 2rem;

color: #666;

}

.spinner {

border: 4px solid #f3f3f3;

border-top: 4px solid #667eea;

border-radius: 50%;

width: 40px;

height: 40px;

animation: spin 1s linear infinite;

margin: 0 auto 1rem;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

.error-container {

background: #ffebee;

color: #c62828;

padding: 1rem;

border-radius: 8px;

margin: 1rem 0;

text-align: center;

}

.error-container button {

background: #c62828;

color: white;

border: none;

padding: 0.5rem 1rem;

border-radius: 5px;

cursor: pointer;

margin-top: 0.5rem;

}

/\* Footer and stats \*/

.app-footer {

background: white;

padding: 1rem;

text-align: center;

border-top: 1px solid #eee;

margin-top: 2rem;

}

.app-footer button {

background: #667eea;

color: white;

border: none;

padding: 0.5rem 1rem;

border-radius: 5px;

cursor: pointer;

margin-left: 1rem;

}

.quick-stats {

background: #f8f9fa;

padding: 1rem;

border-radius: 8px;

margin: 1rem 0;

text-align: center;

}

/\* Responsive design \*/

@media (max-width: 1024px) {

.three-column-layout {

grid-template-columns: 1fr 1fr;

}

}

@media (max-width: 768px) {

.three-column-layout {

grid-template-columns: 1fr;

}

.tab-navigation {

flex-direction: column;

align-items: center;

}

.app-controls {

flex-direction: column;

align-items: center;

}

.course-filters {

flex-direction: column;

align-items: flex-start;

}

.App-header h1 {

font-size: 2rem;

}

}

/\* Dark mode adjustments \*/

.app-dark .column,

.app-dark .admin-controls,

.app-dark .author-panel,

.app-dark .reader-panel,

.app-dark .blog-stats,

.app-dark .top-authors,

.app-dark .course-stats,

.app-dark .recommendations,

.app-dark .quick-stats,

.app-dark .app-footer {

background: #2d2d2d;

color: #fff;

}

.app-dark .book-item,

.app-dark .blog-item,

.app-dark .course-card {

background: #2d2d2d;

border-color: #444;

color: #fff;

}

.app-dark .tab-navigation,

.app-dark .app-controls {

background: #2d2d2d;

border-color: #444;

}

.app-dark .tab-button {

background: #444;

color: #fff;

}

.app-dark .tab-button:hover {

background: #555;

}

.app-dark .tab-button.active {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

}

OUTPUT

