1**.Objectives**

* List the features of ES6

1. l**et** and **const** for block-scoped variables
2. **Arrow functions** (=>)
3. **Classes and Inheritance**
4. **Template literals** (e.g., `${name}`)
5. **Default parameters**
6. **Destructuring assignment**
7. **Spread and rest operators** (...)
8. **Modules** (import/export)
9. **Promises**
10. **Map and Set data structures**

* Explain JavaScript let

1. let allows you to declare block-scoped variables.
2. It prevents issues like variable hoisting and accidental redeclarations

* Identify the differences between var and let

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Hoisting | Yes, initialized as undefined | Yes, but not initialized |
| Redeclaration | Allowed | Not allowed in same scope |

* Explain JavaScript const

1. Declares **block-scoped, read-only** variables.
2. Must be **initialized** during declaration.
3. You **can mutate** objects or arrays assigned to const.

* Explain ES6 class fundamentals

class Person {

constructor(name) {

this.name = name;

}

greet() {

console.log(`Hello, ${this.name}`);

}

}

const p = new Person("Alice");

p.greet();

* Explain ES6 class inheritance

In JavaScript (ES6), class inheritance means creating a new class (called a child class or subclass) that inherits properties and methods from an existing class.

* Define ES6 arrow functions

1. Shorter syntax
2. No binding of this
3. Cannot be used as constructors

// Traditional

function add(a, b) {

return a + b;

}

// Arrow function

const add = (a, b) => a + b;

* Identify set(), map()

**Set**

Stores unique values

const mySet = new Set([1, 2, 2, 3])

console.log(mySet); // Set {1, 2, 3}

**Map**

Stores key-value pairs

Keys can be any type

const myMap = new Map();

myMap.set('a', 1);

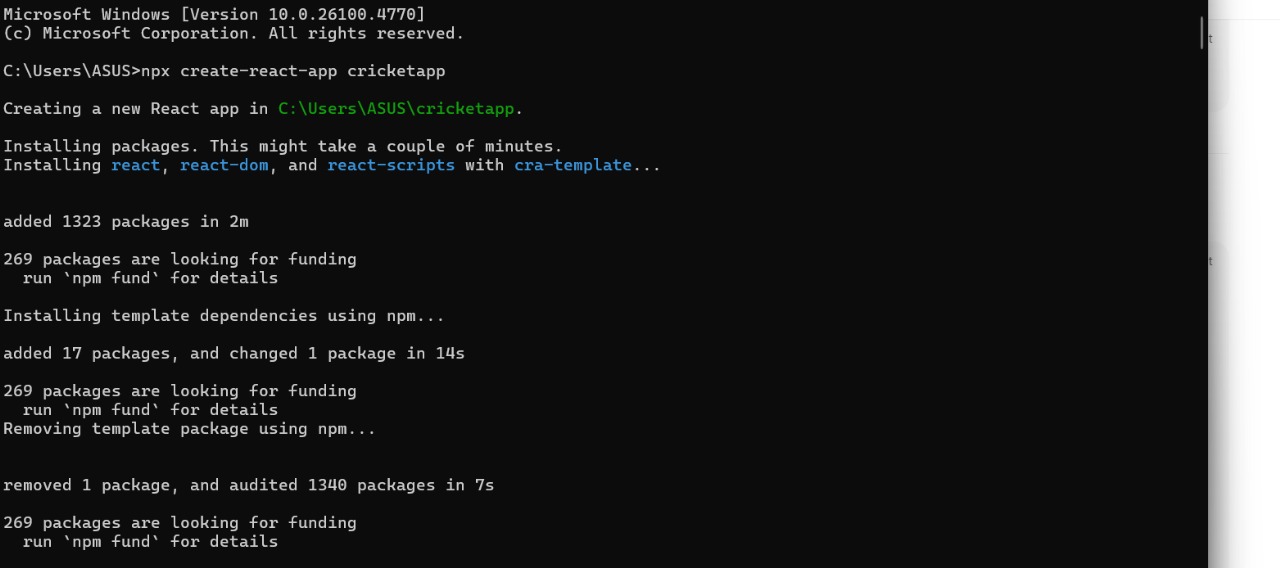
myMap.set(42, 'answer');

console.log(myMap.get(42));

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



**AllPlayers.js**

import React from 'react';

function AllPlayers({ players }) {

  return (

    <div>

      <h2>List of Players</h2>

      <ul>

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

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

        ))}

      </ul>

    </div>

  );

}

export default AllPlayers;

* Filter the players with scores below 70 using arrow functions of ES6.

1. IndianPlayers
   1. Display the Odd Team Player and Even Team players using the Destructuring features of ES6
   2. 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.

**ListBelow70.js**

import React from 'react';

function ListBelow70({ players }) {

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

  return (

    <div>

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

      <ul>

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

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

        ))}

      </ul>

    </div>

  );

}

export default ListBelow70;

**OddPlayers.js**

import React from 'react';

function OddPlayers({ players }) {

  const [first, , third, , fifth] = players;

  return (

    <div>

      <h2>Odd Players</h2>

      <ul>

        <li>First : {first}</li>

        <li>Third : {third}</li>

        <li>Fifth : {fifth}</li>

      </ul>

    </div>

  );

}

export default OddPlayers;

**EvenPlayers.js**

import React from 'react';

function EvenPlayers({ players }) {

  const [, second, , fourth, , sixth] = players;

  return (

    <div>

      <h2>Even Players</h2>

      <ul>

        <li>Second : {second}</li>

        <li>Fourth : {fourth}</li>

        <li>Sixth : {sixth}</li>

      </ul>

    </div>

  );

}

export default EvenPlayers;

**IndianPlayersData.js**

export const T20Players = ['First Player', 'Second Player', 'Third Player'];

export const RanjiTrophyPlayers = ['Fourth Player', 'Fifth Player', 'Sixth Player'];

export const IndianPlayers = [...T20Players, ...RanjiTrophyPlayers];

**MergedIndianPlayers.js**

import React from 'react';

function MergedIndianPlayers({ players }) {

  return (

    <div>

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

      <ul>

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

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

        ))}

      </ul>

    </div>

  );

}

export default MergedIndianPlayers;

**App.js**

import React from 'react';

import './App.css';

import AllPlayers from './AllPlayers';

import ListBelow70 from './ListBelow70';

import OddPlayers from './OddPlayers';

import EvenPlayers from './EvenPlayers';

import MergedIndianPlayers from './MergedIndianPlayers';

import { IndianPlayers } from './IndianPlayersData';

function App() {

  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">

      <AllPlayers players={players} />

      <ListBelow70 players={players} />

      <OddPlayers players={['Sachin1', 'Dhoni2', 'Virat3', 'Rohit4', 'Yuvaraj5', 'Raina6']} />

      <EvenPlayers players={['Sachin1', 'Dhoni2', 'Virat3', 'Rohit4', 'Yuvaraj5', 'Raina6']} />

      <MergedIndianPlayers players={IndianPlayers} />

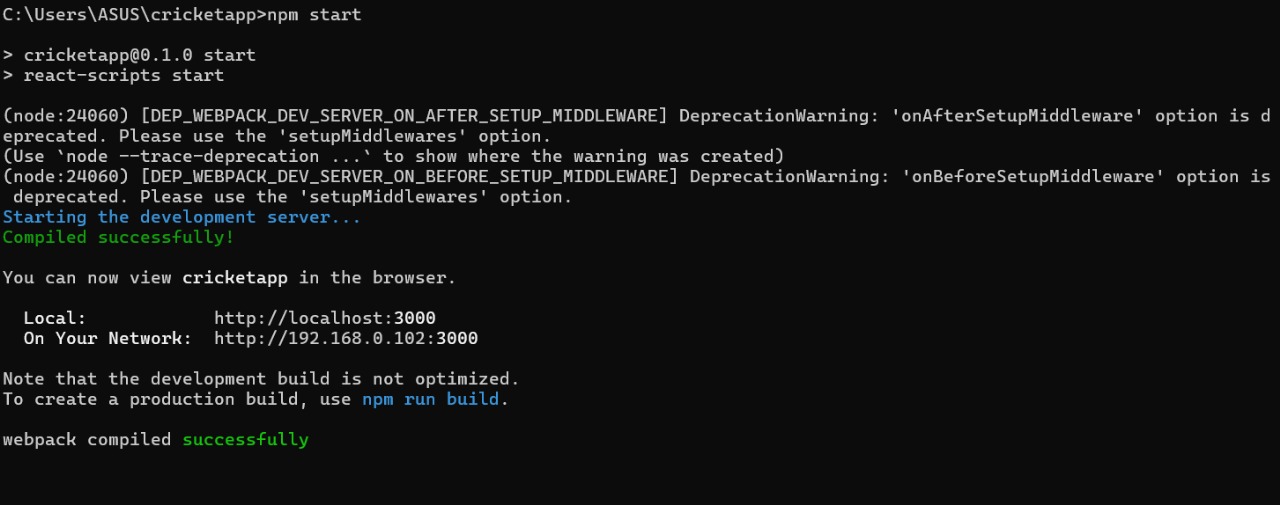
    </div>

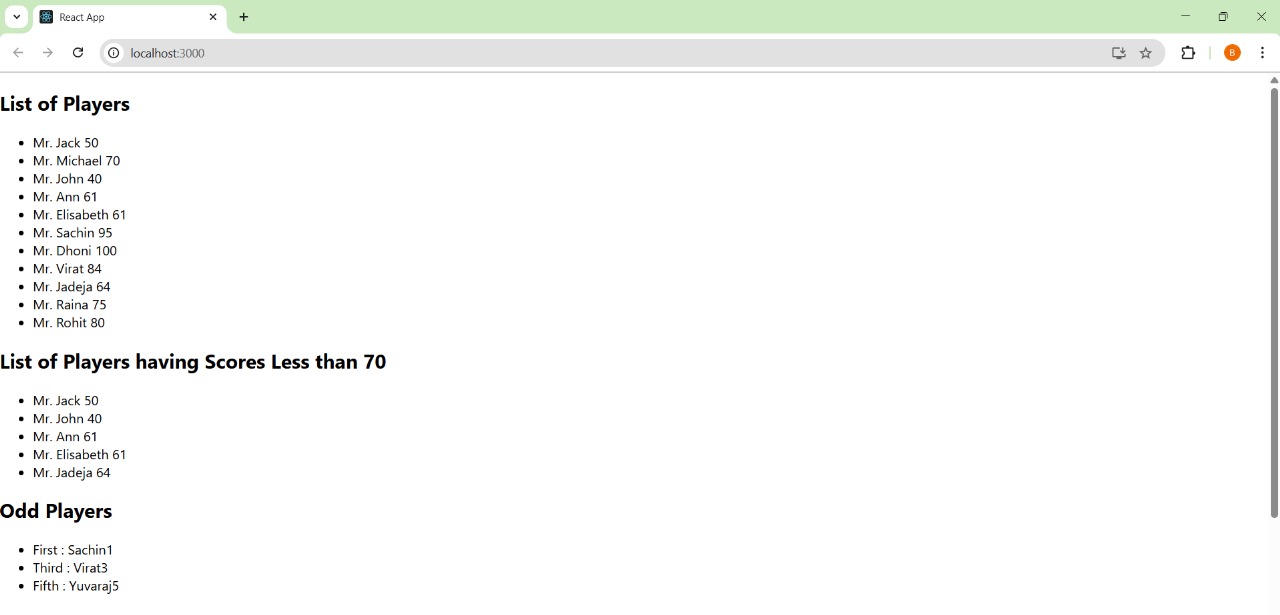
  );

}

export default App;

OUTPUT:







**2.Objectives**

**Define JSX**

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

Example:

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

JSX is easier to read and write than traditional React.createElement() calls**.**

**Explain ECMA Script**

ECMAScript (ES) is the standard specification on which JavaScript is based.  
Different versions like ES5, ES6 (ES2015), etc., define new features of the JavaScript language.

**ES6 introduced features like:**

* let / const
* Arrow functions
* Classes
* Template literals
* Modules (import / export)
* Destructuring
* Promises

**Explain React.createElement()**

React.createElement() is a method used to manually create React elements.

Syntax:

React.createElement(type, props, children)

Example:

const element = React.createElement('h1', { className: 'greeting' }, 'Hello!');

This returns a React element object that gets rendered to the DOM.

**Explain how to create React nodes with JSX**

const heading = <h1>Welcome to React!</h1>;

**Define how to render JSX to DOM**

import React from 'react';

import ReactDOM from 'react-dom';

const element = <h1>Hello React</h1>;

ReactDOM.render(element, document.getElementById('root'));

It inserts the element into the specified DOM node.

**Explain how to use JavaScript expressions in JSX**

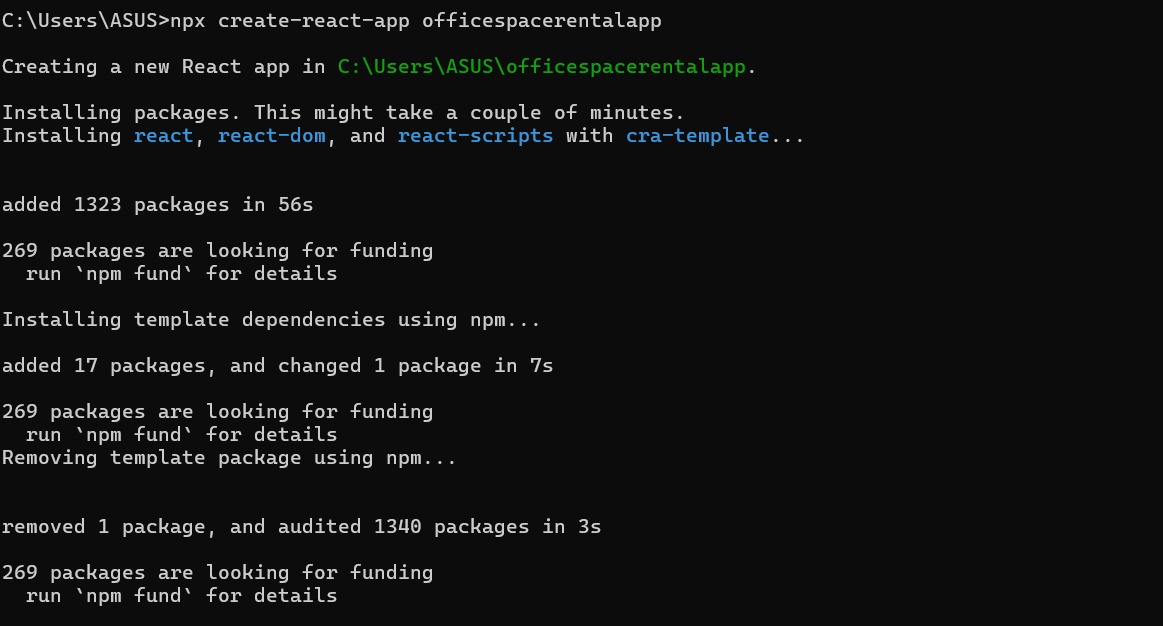
const name = "Alice";

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

**Explain how to use inline CSS in JSX**

const heading = <h1 style={{ color: 'blue', fontSize: '24px' }}>Styled Text</h1>;

* The style keys must be in camelCase (backgroundColor, not background-color).
* The values should be strings or numbers.
* 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.



**App.js**

import React from 'react';

import './App.css';

function App() {

  const element = "Office Space";

  const sr = "data:image/jpeg;base64,/9j/4AAQSkZJRgQABAAD/+77/VoJbQD+ykBk8PBuCK2BV/ Q==";

  const jsxatt = <img src={sr} width="35%" height="35%" alt="Office Space" />;

  const ItemName = { Name: "DBS", Rent: 80000, Address: "Chennai" };

  let colors = [];

  if (ItemName.Rent <= 60000) {

    colors.push("textRed");

  } else {

    colors.push("textGreen");

  }

  return (

    <div className="container">

      <h1>{element} , at Affordable Range</h1>

      {jsxatt}

      <h1>Name: {ItemName.Name}</h1>

      <h3 className={colors.join(' ')}>Rent: Rs. {ItemName.Rent}</h3>

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

    </div>

  );

}

export default App;

**App.css**

body {

  margin: 0;

  padding: 0;

  font-family: Arial, sans-serif;

}

.container {

  max-width: 900px;

  margin: 50px auto;

  text-align: center;

  padding: 30px;

}

img {

  margin: 20px 0;

}

.textRed {

  color: red;

  font-weight: bold;

}

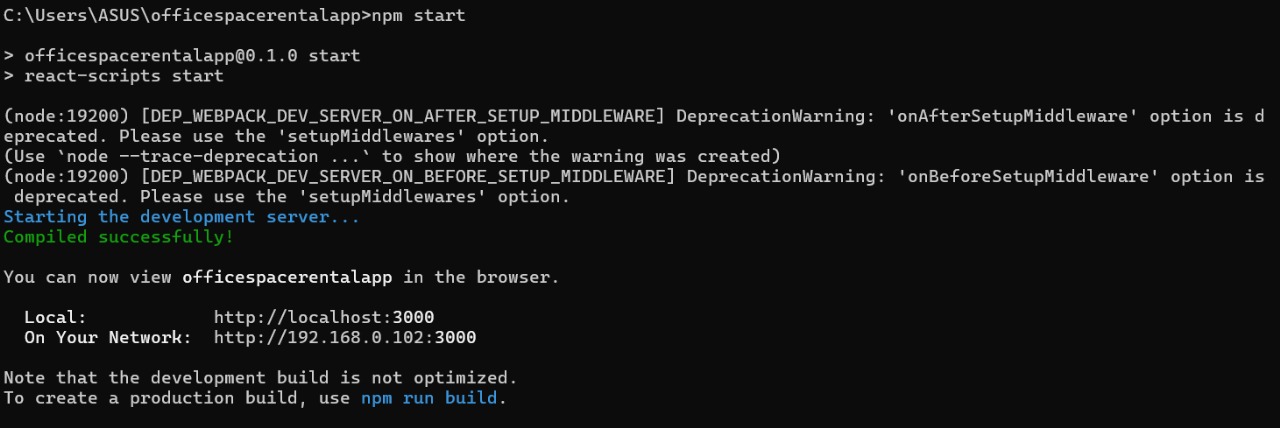
.textGreen {

  color: green;

  font-weight: bold;

}

OUTPUT:





**3,Objectives**

**Explain React Events**

In React, events are how your app responds to user interactions, like clicks, key presses, mouse movements, etc.  
React handles events similarly to DOM events but uses a cross-browser wrapper called Synthetic Events for consistency.

**Example:**

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

**Explain Event Handlers**

An event handler is a function that is called when an event occurs.

For example:

function handleClick() {

alert('Button was clicked!');

}

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

<button onClick={() => console.log('Clicked!')}>Click</button>

**Define Synthetic Event**

A SyntheticEvent is React’s cross-browser wrapper around the native browser event.  
It behaves like the native event but works identically across all browsers for consistency.

React creates a synthetic event for every event:

function handleInput(event) {

console.log(event.target.value); // SyntheticEvent

}

These synthetic events wrap native events, so they include all standard event methods like preventDefault() and stopPropagation().

**Identify React Event Naming Convention**

React follows a specific naming convention for event handlers:

| **Feature** | **React** | **HTML** |
| --- | --- | --- |
| Event name | **camelCase** | lowercase |
| Function | Pass a function | String or inline JS |

Examples:

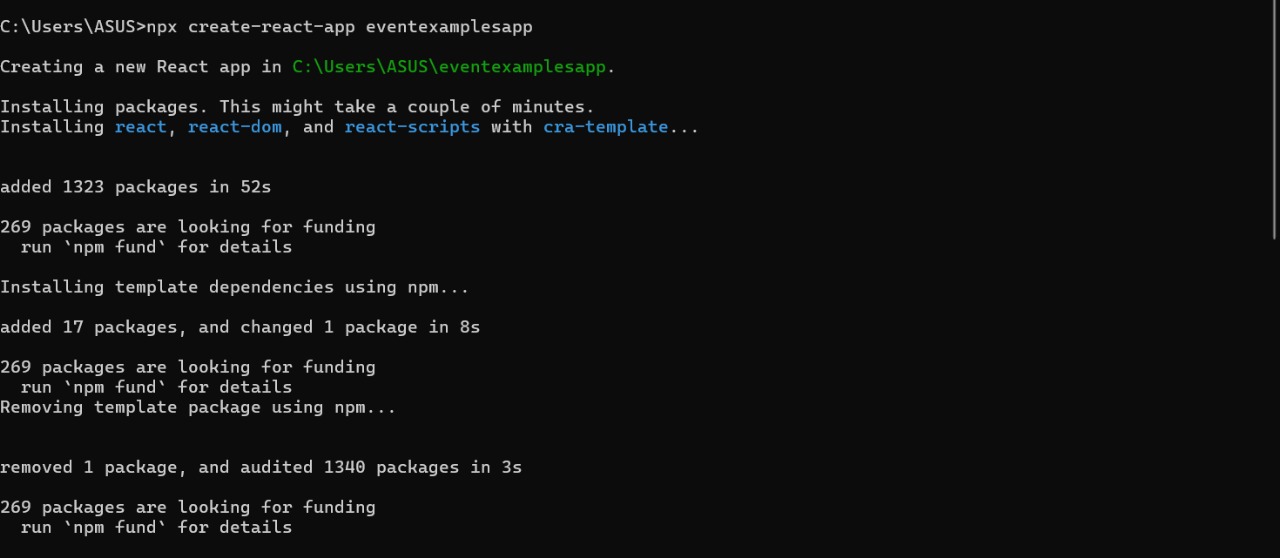
| **Task** | **HTML** | **React** |
| --- | --- | --- |
| Click | <button onclick="doSomething()"> | <button onClick={doSomething}> |
| Submit | <form onsubmit="submitForm()"> | <form onSubmit={submitForm}> |

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.



**CurrencyConvertor.js**

import React, { useState } from 'react';

function CurrencyConvertor() {

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

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

  const handleSubmit = (e) => {

    e.preventDefault();

    const euroRate = 0.011;

    const converted = (parseFloat(amount) \* euroRate).toFixed(2);

    setCurrency(converted);

    alert(`Converted Amount in Euro: €${converted}`);

  };

  return (

    <div>

      <h1 style={{ color: 'green' }}>Currency Convertor!!!</h1>

      <form onSubmit={handleSubmit}>

        <label>Amount:</label>

        <input

          type="number"

          value={amount}

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

        /><br /><br />

        <label>Currency:</label>

        <textarea value={currency} readOnly /><br /><br />

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

      </form>

    </div>

  );

}

export default CurrencyConvertor;

**App.js**

import React, { useState } from 'react';

import './App.css';

import CurrencyConvertor from './components/CurrencyConvertor';

function App() {

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

  const incrementValue = () => {

    setCount(prev => prev + 1);

    sayHello();

  };

  const sayHello = () => {

    alert("Hello! Member1");

  };

  const decrementValue = () => {

    setCount(prev => prev - 1);

  };

  const sayWelcome = (message) => {

    alert(message);

  };

  const handleClick = () => {

    alert("I was clicked");

  };

  return (

    <div className="container">

      <h2>{count}</h2>

      <button onClick={incrementValue}>Increment</button><br /><br />

      <button onClick={decrementValue}>Decrement</button><br /><br />

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

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

      <CurrencyConvertor />

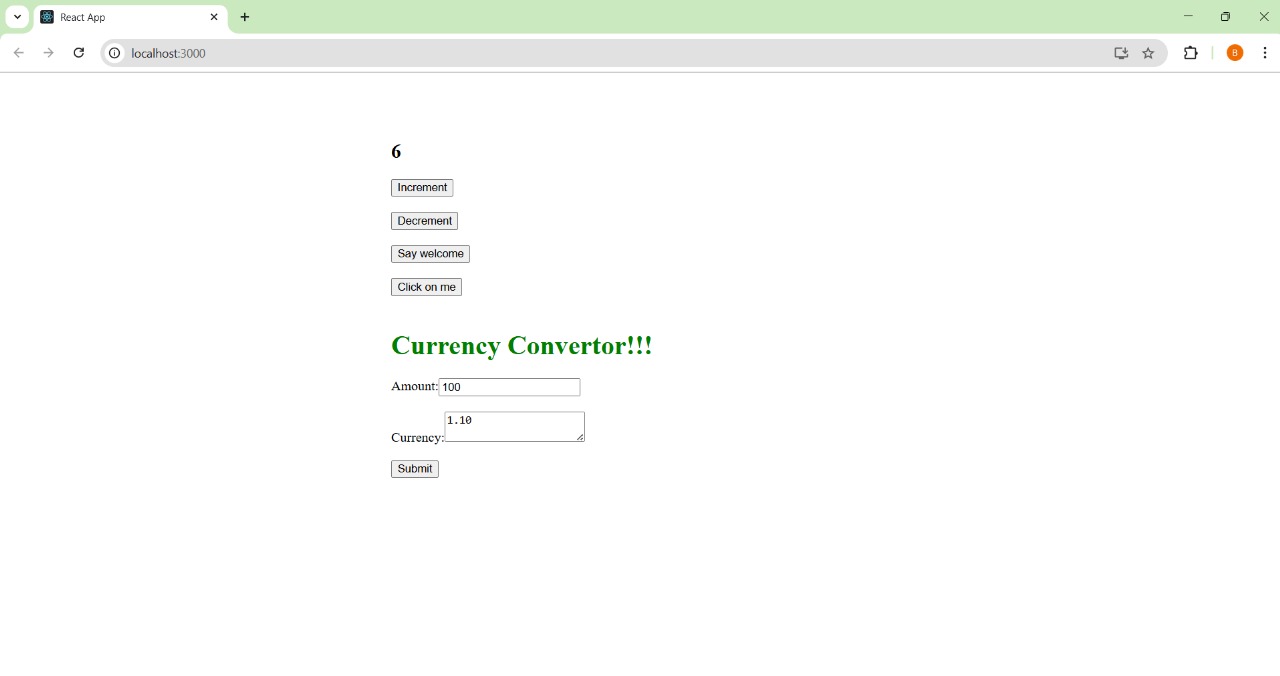
    </div>

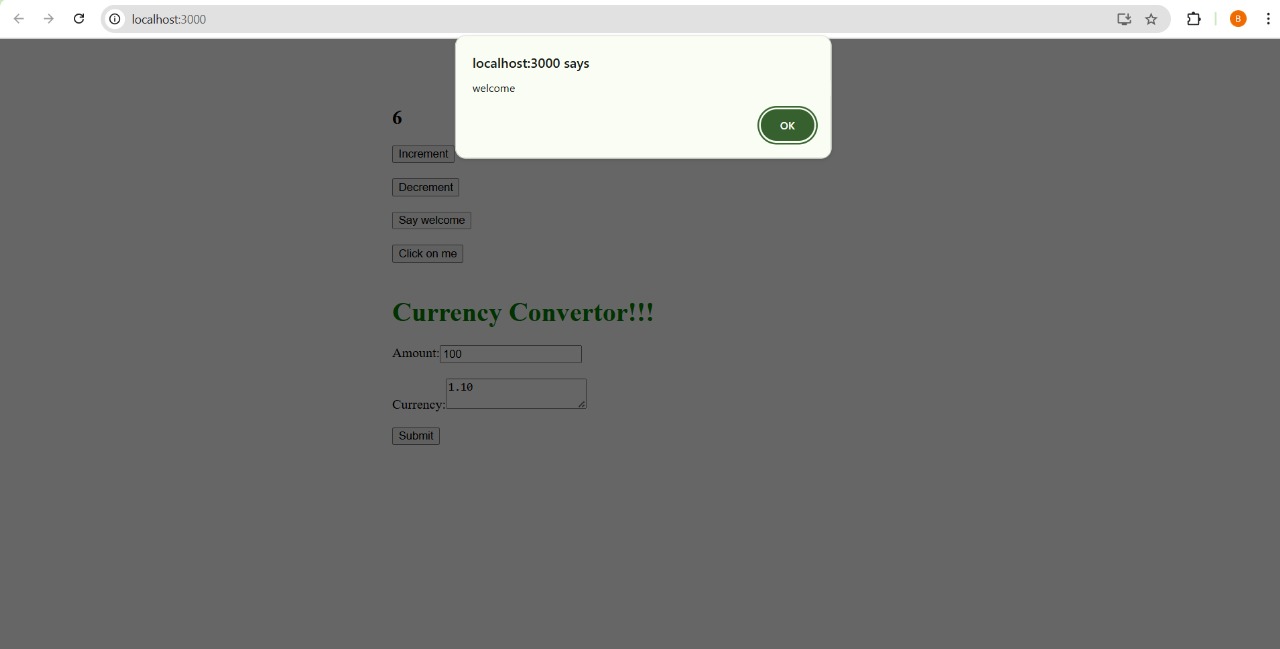
  );

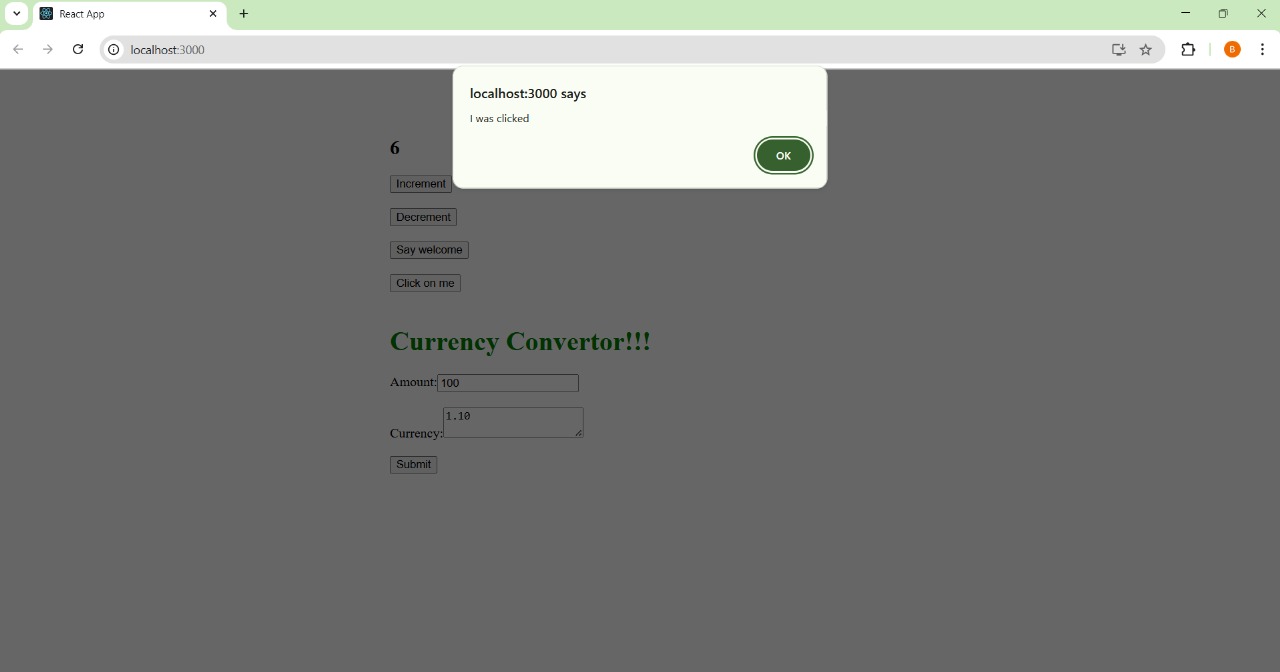
}

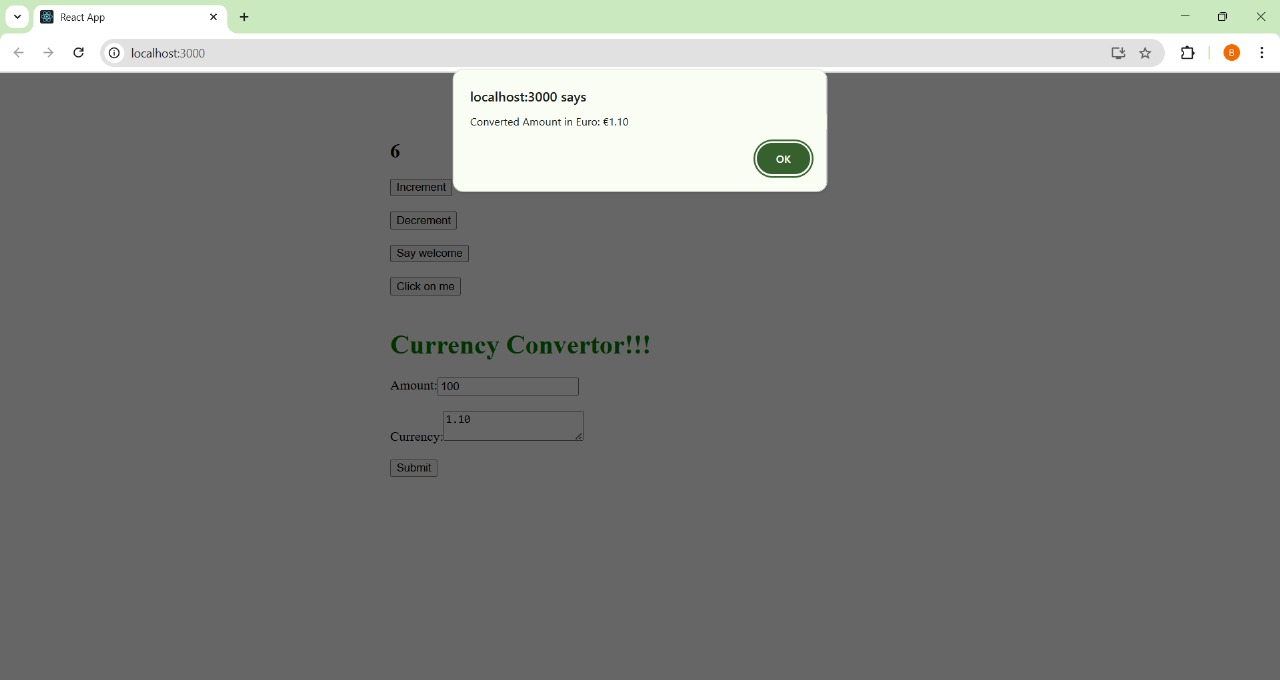
export default App;

OUTPUT:









**4.Objectives**

**Explain Conditional Rendering in React**

Conditional rendering means showing or hiding UI elements based on conditions, like a user’s login status or data availability.

React uses JavaScript conditions inside JSX to control what gets displayed.

Example using if-else:

function Greeting(props) {

if (props.isLoggedIn) {

return <h1>Welcome back!</h1>;

}

return <h1>Please log in.</h1>;

}

Example using ternary operator:

<h1>{isLoggedIn ? 'Welcome!' : 'Login required'}</h1>

Using logical AND (&&) operator:

{isAdmin && <button>Delete User</button>}

**Define Element Variables**

Element variables are used to store JSX elements in a variable.  
This allows you to conditionally assign components or elements before rendering.

Example:

let message;

if (isLoggedIn) {

message = <h1>Hello, User</h1>;

} else {

message = <h1>Please Log In</h1>;

}

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

**Explain How to Prevent Components from Rendering**

1. Returning null

A component that returns null renders nothing but does not break the app.

Example:

function Warning(props) {

if (!props.show) {

return null; // Prevent rendering

}

return <p>Warning Message!</p>;

}

2. Using short-circuit conditions

{shouldShow && <Component />}

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.

**App.js**

import React, { useState } from 'react';

import Greeting from './components/Greeting';

import LoginButton from './components/LoginButton';

import LogoutButton from './components/LogoutButton';

import FlightDetails from './components/FlightDetails';

import BookingSection from './components/BookingSection';

import './App.css';

function App() {

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

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

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

  return (

    <div className="container">

      <Greeting isLoggedIn={isLoggedIn} />

      {isLoggedIn ? (

        <>

          <LogoutButton onClick={handleLogoutClick} />

          <FlightDetails />

          <BookingSection />

        </>

      ) : (

        <LoginButton onClick={handleLoginClick} />

      )}

    </div>

  );

}

export default App;

**LoginButton.js**

export default function LoginButton(props) {

  return <button onClick={props.onClick}>Login</button>;

}

**LogoutButton.js**

export default function LogoutButton(props) {

  return <button onClick={props.onClick}>Logout</button>;

}

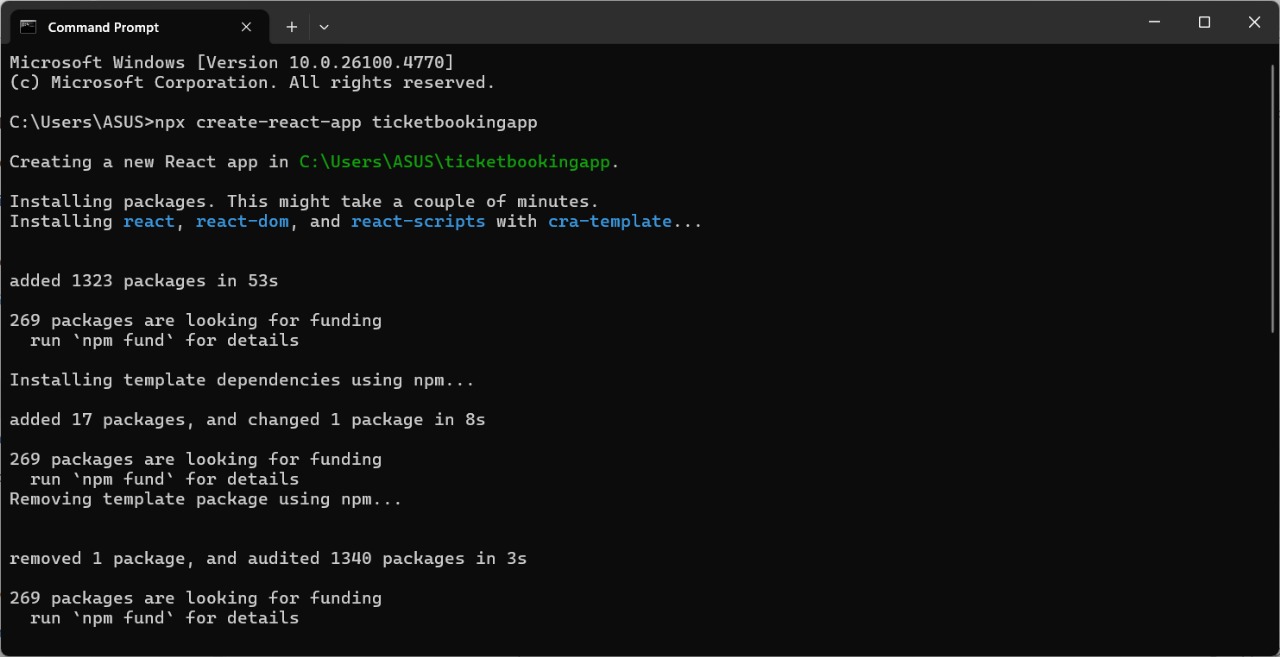
**UserGreeting.js**

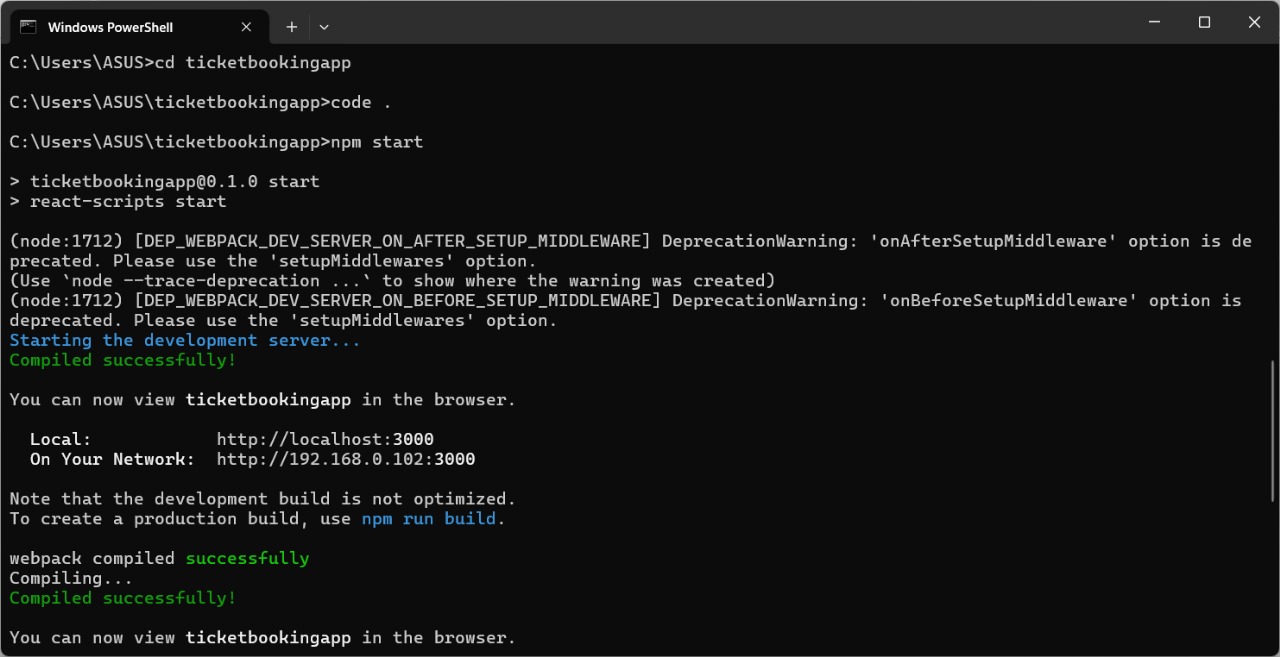
export default function UserGreeting() {

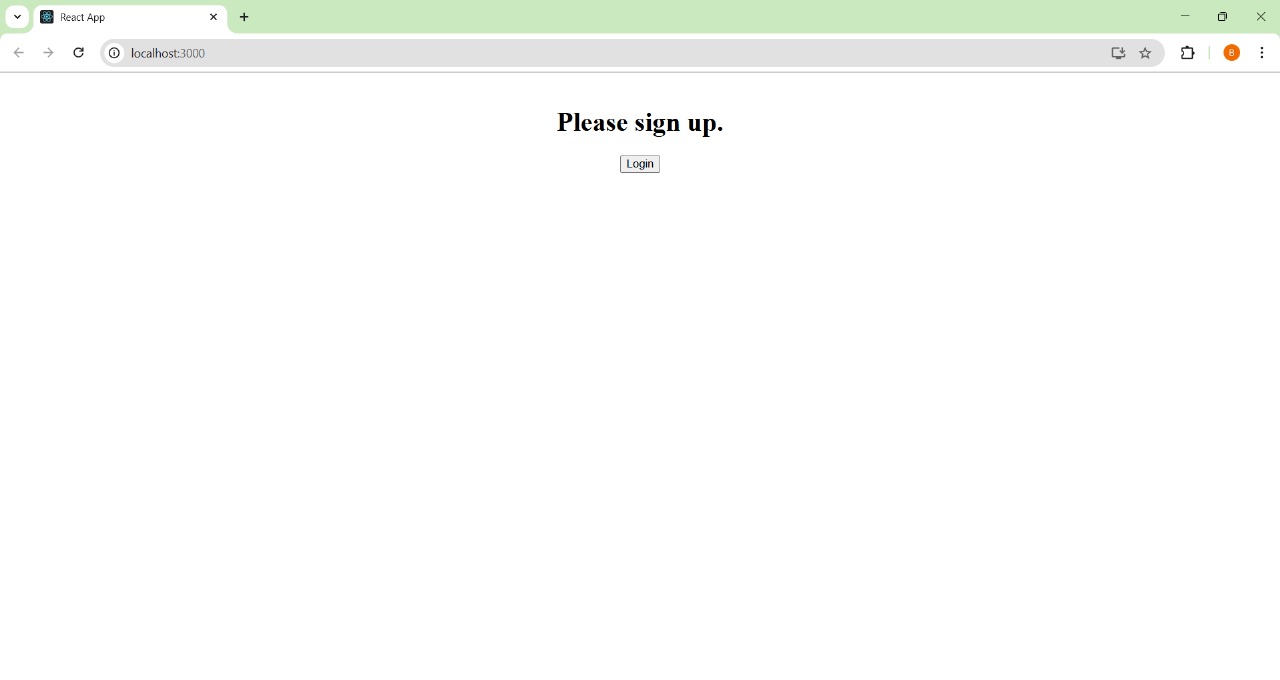
  return <h1>Welcome back</h1>;

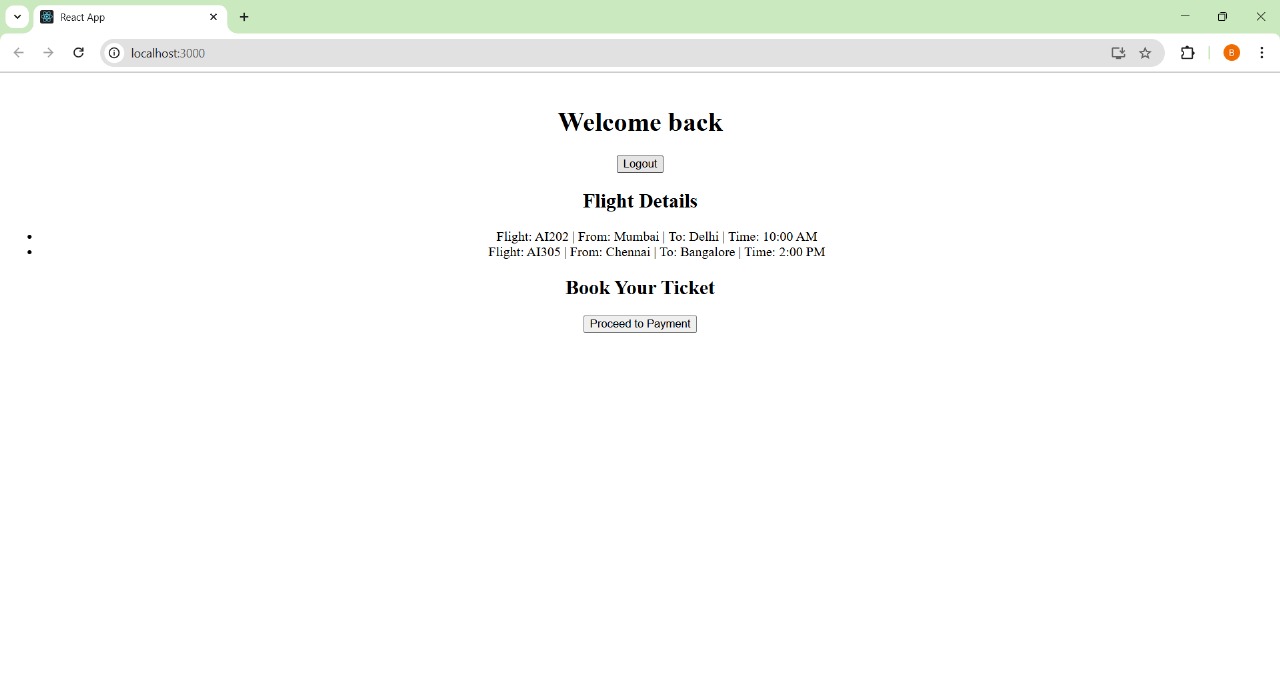
}

OUTPUT:









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

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

**App.js**

import React from "react";

import BookDetails from "./components/BookDetails";

import BlogDetails from "./components/BlogDetails";

import CourseDetails from "./components/CourseDetails";

import { books, blogs, courses } from "./components/data";

import "./App.css";

function App() {

  const view = "all";

  return (

    <div className="container">

      <h1>Blogger App</h1>

      <div className="details-wrapper">

        {(view === "course" || view === "all") && (

          <div className="detail-box">

            <CourseDetails courses={courses} />

          </div>

        )}

        {(view === "book" || view === "all") && (

          <div className="detail-box">

            <BookDetails books={books} />

          </div>

        )}

        {(view === "blog" || view === "all") && (

          <div className="detail-box">

            <BlogDetails blogs={blogs} />

          </div>

        )}

      </div>

    </div>

  );

}

export default App;

**App.css**

.container {

  padding: 20px;

  font-family: Arial, sans-serif;

  text-align: center;

}

.details-wrapper {

  display: flex;

  justify-content: space-around;

  margin-top: 30px;

}

.detail-box {

  border: 2px solid green;

  border-radius: 10px;

  padding: 20px;

  width: 25%;

  box-shadow: 0 0 10px rgba(0, 128, 0, 0.1);

}

**Data.js**

export const books = [

  { name: "Master React", price: 780 },

  { name: "Deep Dive into Angular 11", price: 860 },

  { name: "Mongo Essentials", price: 750 }

];

export const blogs = [

  { title: "React Learning", author: "Stephen Biz", content: "Welcome to learning React!" },

  { title: "Installation", author: "Schewzdenier", content: "You can install React from npm." }

];

export const courses = [

  { name: "Angular", date: "4/7/2025" },

  { name: "React", date: "6/7/2025" }

];

**BookDetails.js**

import React from "react";

function BookDetails({ books }) {

  return (

    <div className="section">

      <h2>Book Details</h2>

      {books.map((book, index) => (

        <div key={index}>

          <h3>{book.name}</h3>

          <p>{book.price}</p>

        </div>

      ))}

    </div>

  );

}

export default BookDetails;

**BlogDetails.js**

import React from "react";

function BlogDetails({ blogs }) {

  return (

    <div className="section">

      <h2>Blog Details</h2>

      {blogs.map((blog, index) => (

        <div key={index}>

          <h3>{blog.title}</h3>

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

          <p>{blog.content}</p>

        </div>

      ))}

    </div>

  );

}

export default BlogDetails;

**CourseDetails.js**

import React from "react";

function CourseDetails({ courses }) {

  return (

    <div className="section">

      <h2>Course Details</h2>

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

        <div key={index}>

          <h3>{course.name}</h3>

          <p>{course.date}</p>

        </div>

      ))}

    </div>

  );

}

export default CourseDetails;

OUTPUT:

