**DN 4.0 WEEK 7 MANDATORY PROBLEMS**

**MODULE 9: REACT JS**

1. **OBJECTIVES:**
   1. **List the features of ES6 (ECMAScript 2025)**

ES6 introduced major improvements to JavaScript to make it more powerful and readable

Key features are:

* Let and const
* Arrow functions
* Template literals
* Destructuring
* Spread and Rest
* Default Parameters
* Classes and Inheritance
* Map and Set
  1. **Explain JavaScript let**

Let allows us to declare block scoped variables.

Unlike var, which is function scoped.

* 1. **Identify difference between var and let**

Var is Function Scoped and redeclaration is allowed in same scope.

Let is block scoped and redeclaration is not allowed in same scope.

* 1. **Explain javascript const**

Used for declaring constants values that are cannot be reassigned.

It is also block scoped.

You can change the content but you cannot reassign

Ex:

Const player={name: ‘Virat’};

Player.name= ‘Raina’; // This is allowed

Player={name:’yuvraj’}; // this is not allowed

* 1. **Explain ES6 Class fundamentals**

ES6 has introduced class as syntactic sugar over JavaScript’s prototype base inheritance

Uses constructor for initialization

Methods are declared inside the class so there is no need to use function keyword.

* 1. **Explain ES6 Class Inheritance**

We use extends keyword to inherit from another class and super() to call the parent constructor same as how we use in OOPS concept

Ex:

Class Cricketer extends Player{

Constructor(name,score,team){

Super(name,score); // this calls player constructor

This.team=team;

}

}

So this program inherits all the properties and methods from player class.

* 1. **Define ES6 Arrow functions**

Arrow functions provide a shorter syntax for functions and don’t bind their own this

Ex:

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

Const greet=name=>’Hello,${name}’;

* 1. **Identify Set() and Map()**

Set stores unique values same as in python sets

Ex:

Const nums=new Set([1,2,3,3]);

Console.log(nums); // this prints {1,2,3} duplicates are excluded

Map stores key value pairs where key can be any data type.

Ex:

Const scores= new map();

Score.set(‘Dhoni’,100);

Scores.set(‘Raina’,90);

Console.log(scores.get(‘Raina’)); // This gives output as 90

**PROGRAM:**

**Create a react app named cricketapp and add 2 files in src folder named IndianPlayers.js and ListofPlayers.js in that**

**And modify app.js and index.js**

**IndianPlayers.js**

import React from 'react';

function IndianPlayers() {

  const finalXI = [

    'Sachin Tendulkar', 'Virender Sehwag', 'Gautam Gambhir',

    'Yuvraj Singh', 'MS Dhoni', 'Suresh Raina',

    'Harbhajan Singh', 'Zaheer Khan', 'Munaf Patel',

    'Sreesanth', 'Yusuf Pathan'

  ];

  const oddTeam = finalXI.filter((\_, index) => index % 2 === 0);

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

  const [odd1, odd2, ...restOdd] = oddTeam;

  const [even1, even2, ...restEven] = evenTeam;

  const T20players = ['Suresh Raina', 'Yusuf Pathan', 'Yuvraj Singh'];

  const RanjiTrophyPlayers = ['Rahane', 'Pujara', 'Wasim Jaffer'];

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

  return (

    <div>

      <h2>Destructuring Odd and Even Team Members</h2>

      <p><strong>Odd Team First Two:</strong> {odd1}, {odd2}</p>

      <p><strong>Even Team First Two:</strong> {even1}, {even2}</p>

      <p><strong>Remaining Odd Team:</strong> {restOdd.join(', ')}</p>

      <p><strong>Remaining Even Team:</strong> {restEven.join(', ')}</p>

      <hr />

      <h2>Merged Player List (T20 + Ranji Trophy)</h2>

      <ul>

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

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

        ))}

      </ul>

    </div>

  );

}

export default IndianPlayers;

**ListofPlayers.js**

import React from 'react';

function ListofPlayers() {

  const players = [

    { name: 'Sachin Tendulkar', score: 85 },

    { name: 'Virender Sehwag', score: 45 },

    { name: 'Gautam Gambhir', score: 97 },

    { name: 'Yuvraj Singh', score: 74 },

    { name: 'MS Dhoni', score: 91 },

    { name: 'Suresh Raina', score: 56 },

    { name: 'Harbhajan Singh', score: 35 },

    { name: 'Zaheer Khan', score: 44 },

    { name: 'Munaf Patel', score: 25 },

    { name: 'Sreesanth', score: 22 },

    { name: 'Yusuf Pathan', score: 65 },

  ];

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

  return (

    <div>

      <h2>2011 World Cup Squad (Using map method):</h2>

      <ul>

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

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

        ))}

      </ul>

      <h2>Players with Score &lt; 70 (Using arrow + filter):</h2>

      <ul>

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

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

        ))}

      </ul>

    </div>

  );

}

export default ListofPlayers;

**App.js**

import React from 'react';

import ListofPlayers from './ListofPlayers';

import IndianPlayers from './IndianPlayers';

function App() {

  const flag = true; *// Change to false to see IndianPlayers component*

  return (

    <div style={{ padding: '30px', fontFamily: 'Arial' }}>

      <h1>Cricket Player App</h1>

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

    </div>

  );

}

export default App;

**Index.js**

import React from 'react';

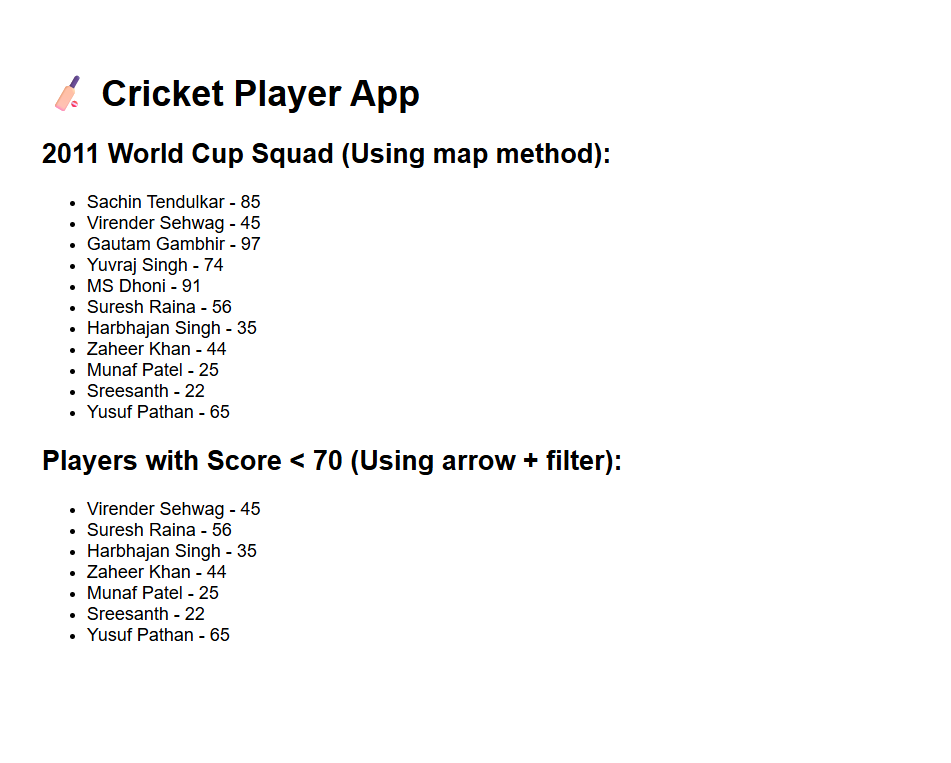
import ReactDOM from 'react-dom/client';

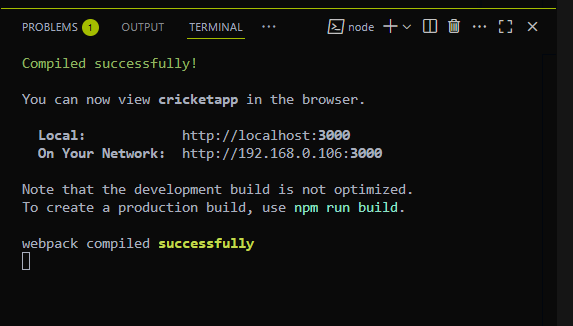
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**OUTPUT:**



****

1. **OBJECTIVES:**
   1. **Define JSX**

JSX means JavaScript XML is a syntax extension for JavaScript used in React to describe what the UI should look like. It allows you to write HTML lke code directly in JavaScript files

Ex:

Const heading=<h1> Cognizant digital Nuture</h1>;

JSX make code more readable and easier to write.

* 1. **Explain ECMA Script**

ECMAScript (often abbreviated as ES) is the standard specification that JavaScript follows. React apps commonly use features from ES6 and later.

Ex:

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

* 1. **Explain React.createElement()**

This method is used by React internally to create elements. JSX is syntactic sugar for this.

Ex:

const element = React.createElement('h1', null, 'Hello World');

in jsx we can write like this also

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

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

React nodes (elements) can be created using JSX syntax

Ex:

const title = <h2>Cognizant digital nurture</h2>;

const img = <img src="office.jpg" alt="Office space" />;

* 1. **Define how to render JSX to the DOM**

JSX needs to be rendered into the DOM using the ReactDOM.render()

Ex:

import ReactDOM from 'react-dom/client';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

* 1. **Explain how to use JavaScript expression in JSX**

You can use JavaScript expression inside JSX by enclosing them in curly braces {}

Ex:

const name = "WeWork";

const element = <h1>{name} Office Space</h1>;

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

You can apply CSS styles inline using a JavaScript object with camelCased property names.

Ex:

const style = {

color: "red",

backgroundColor: "lightgray"

};

const heading = <h2 style={style}>Cognizant Digital Nurture</h2>;

**PROGRAM:**

**Here we need change the app.js and index.js**

**App.js**

import React from 'react';

import './App.css';

function App() {

  const officeList = [

    {

      id: 1,

      name: 'Tech Park A',

      rent: 55000,

      address: 'Madhapur, Hyderabad',

      image: 'https://via.placeholder.com/250x150?text=Tech+Park+A',

    },

    {

      id: 2,

      name: 'Workspace B',

      rent: 75000,

      address: 'Indiranagar, Bangalore',

      image: 'https://via.placeholder.com/250x150?text=Workspace+B',

    },

    {

      id: 3,

      name: 'Office Hub C',

      rent: 60000,

      address: 'Kondapur, Hyderabad',

      image: 'https://via.placeholder.com/250x150?text=Office+Hub+C',

    },

  ];

  const getRentColor = (rent) => {

    return {

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

      fontWeight: 'bold',

    };

  };

  return (

    <div style={{ padding: '30px', fontFamily: 'Arial' }}>

      <h1>Office Space Rental Listings</h1>

      {officeList.map((office) => (

        <div key={office.id} style={{ marginBottom: '20px', border: '1px solid gray', padding: '10px', borderRadius: '10px' }}>

          <img src={office.image} alt={office.name} width="250" height="150" />

          <h2>{office.name}</h2>

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

          <p><strong>Rent:</strong> <span style={getRentColor(office.rent)}>₹{office.rent}</span></p>

        </div>

      ))}

    </div>

  );

}

export default App;

**Index.js**

import React from 'react';

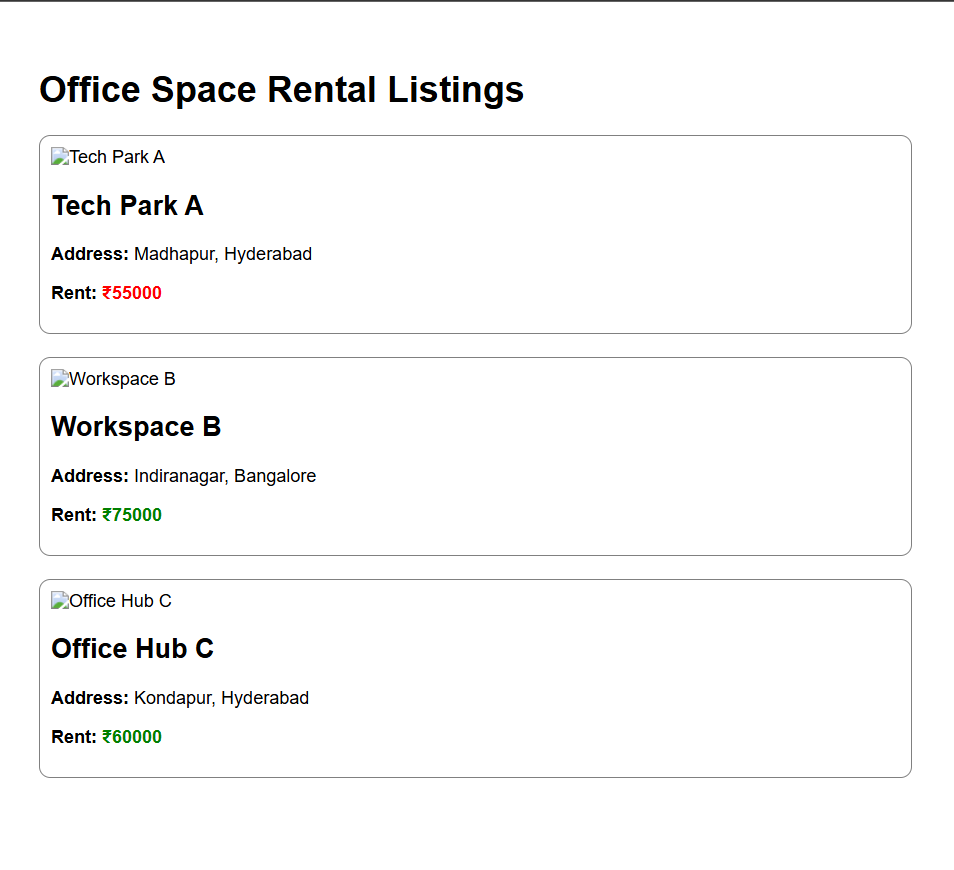
import ReactDOM from 'react-dom/client';

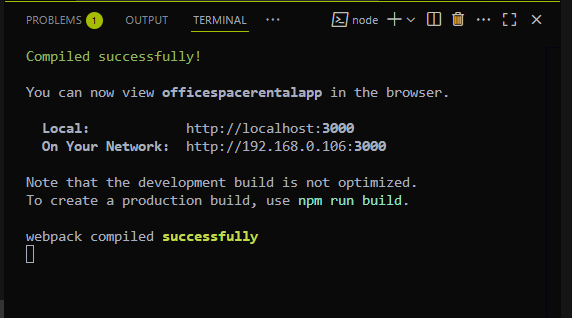
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**OUTPUT:**





1. **OBJECTIVES:**
   1. **Explain React Events**

In React, events are actions triggered by users

These events are similar to HTML DOM events like onclick and onchange, etc

Ex

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

* 1. **Explain Event handlers**

An Event handlers is a function that gets executedin response to an event.

It tells the react what to do when a user event occurs

Ex:

function handleClick() {

alert("Button was clicked");

}

* 1. **Define Synthetic Event**

React creates a Synthetic Event a wrapper around the browsers native event.

It makes react events consistent across all browsers

It has all the normal methods.

Ex:

function handleClick(e) {

e.preventDefault(); // this Prevents page reload

console.log("SyntheticEvent triggered:", e);

}

* 1. **Identify React Event Naming Convention**

React follows strict camelCase naming for events

Like in html we use onclick and onchange but in react we have to use only camecase naming where we use onChange, onClick instead of those conventions.

**PROGRAM:**

**Create a react app named eventexamplesapp**

**And in src create theses files**

**App.js**

import React, { useState } from 'react';

import CurrencyConvertor from './CurrencyConvertor';

function App() {

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

  function increment() {

    setCount(prev => prev + 1);

    sayHello();

  }

  function decrement() {

    setCount(prev => prev - 1);

  }

  function sayHello() {

    console.log("Hello! Counter has been incremented.");

  }

  function sayMessage(msg) {

    alert("Message: " + msg);

  }

  function handleSyntheticEvent(e) {

    alert("I was clicked");

    console.log(e);

  }

  return (

    <div style={{ padding: '30px', fontFamily: 'Arial' }}>

      <h1>React Event Handling Lab</h1>

      <h2>Counter: {count}</h2>

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

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

      <hr />

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

      <hr />

      <button onClick={handleSyntheticEvent}>Synthetic Event Button</button>

      <hr />

      <CurrencyConvertor />

    </div>

  );

}

export default App;

**CurrencyConverter.js**

import React, { useState } from 'react';

function CurrencyConvertor() {

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

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

  const handleSubmit = (e) => {

    e.preventDefault(); *// Prevent form reload*

    if (rupees && !isNaN(rupees)) {

      const converted = (parseFloat(rupees) / 90).toFixed(2);

      setEuros(converted);

    } else {

      alert("Please enter a valid number");

    }

  };

  return (

    <div>

      <h2>Currency Convertor (INR ➝ EUR)</h2>

      <form onSubmit={handleSubmit}>

        <input

          type="number"

          placeholder="Enter amount in INR"

          value={rupees}

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

        />

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

      </form>

      {euros && <p>Equivalent in Euros: €{euros}</p>}

    </div>

  );

}

export default CurrencyConvertor;

**Index.js**

import React from 'react';

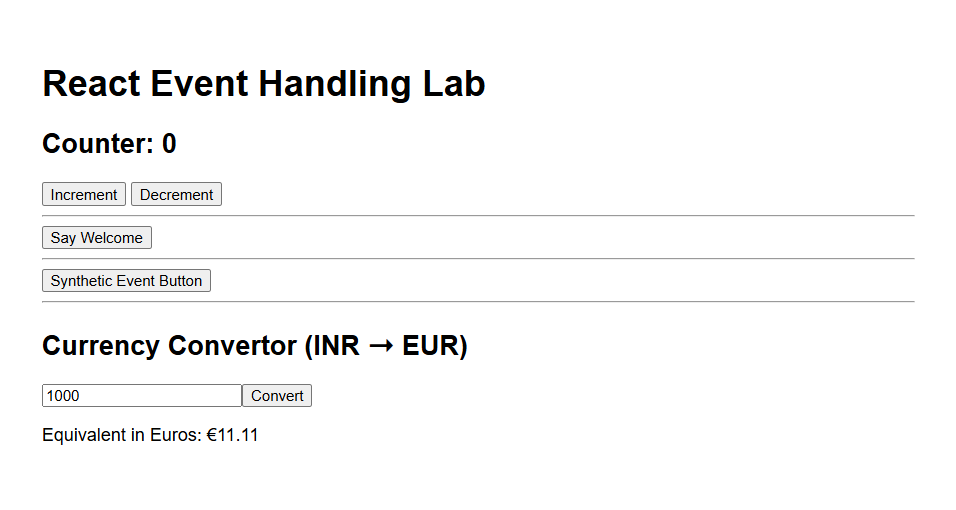
import ReactDOM from 'react-dom/client';

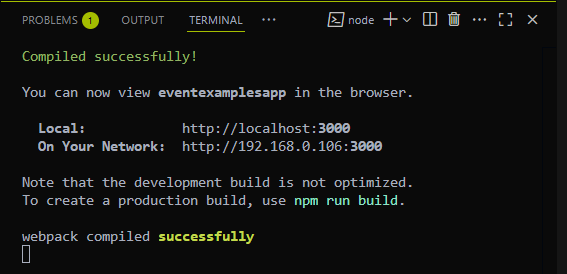
import App from './App';

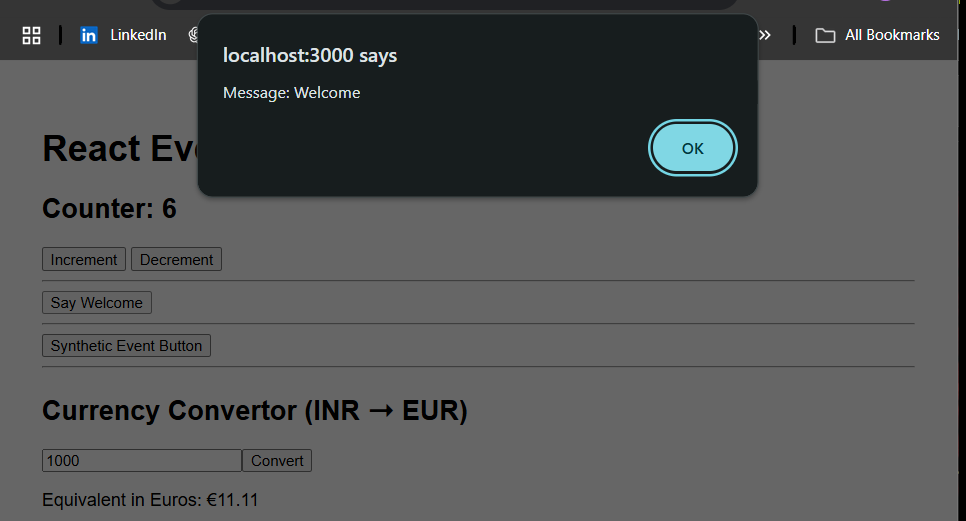
const root = ReactDOM.createRoot(document.getElementById('root'));

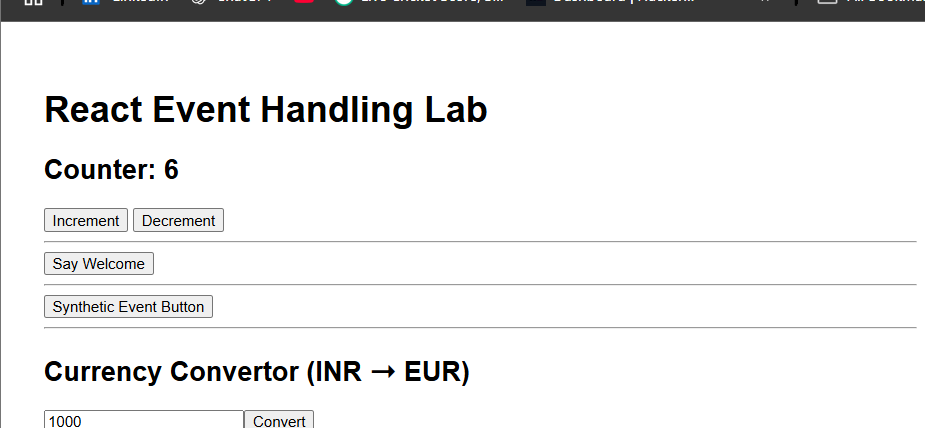
root.render(<App />);

**OUTPUT:**









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

Conditional rendering in React means Displaying different components or elements based on a condition

Ex:

{isLoggedIn ? <UserPage /> : <GuestPage />}

* 1. **Define element variables**

You can assign components or JSX elements to variables and return them from render methods.

Ex:

let content;

if (isLoggedIn) {

content = <UserPage />;

} else {

content = <GuestPage />;

}

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

* 1. **Explain how to prevent components from rendering**

To prevent a component from rendering you can return null to stop rendering

Ex:

if (!isLoggedIn) return null;

**PROGRAM:**

**Now create the react app named ticketbookingapp**

**And create a folder under src named components inside that write these files**

**FlightDetails.js**

const FlightDetails = () => {

  return (

    <div>

      <h2>Flight Details</h2>

      <p>Destination: Visakhapatnam</p>

      <p>Departure: 10:00 AM</p>

    </div>

  );

};

export default FlightDetails;

**GuestPage.js**

import FlightDetails from './FlightDetails';

const GuestPage = () => {

  return (

    <div>

      <h1>Welcome, Guest!</h1>

      <FlightDetails />

      <p>Please log in to book tickets.</p>

    </div>

  );

};

export default GuestPage;

**UserPage.js**

import FlightDetails from './FlightDetails';

const UserPage = () => {

  return (

    <div>

      <h1>Welcome, User!</h1>

      <FlightDetails />

      <button>Book Ticket</button>

    </div>

  );

};

export default UserPage;

Now modify the app.js file

**App.js**

import React, { useState } from 'react';

import GuestPage from './components/GuestPage';

import UserPage from './components/UserPage';

function App() {

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

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

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

  let pageContent;

  if (isLoggedIn) {

    pageContent = <UserPage />;

  } else {

    pageContent = <GuestPage />;

  }

  return (

    <div className="App">

      <h1>Ticket Booking App</h1>

      {isLoggedIn ? (

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

      ) : (

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

      )}

      {pageContent}

    </div>

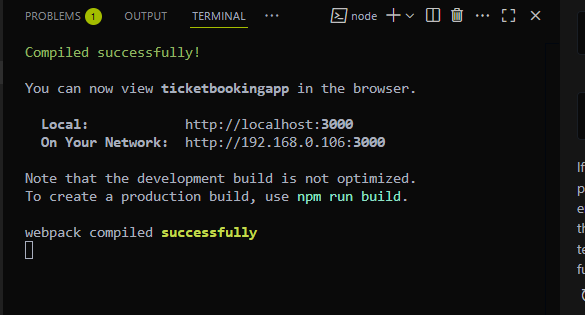
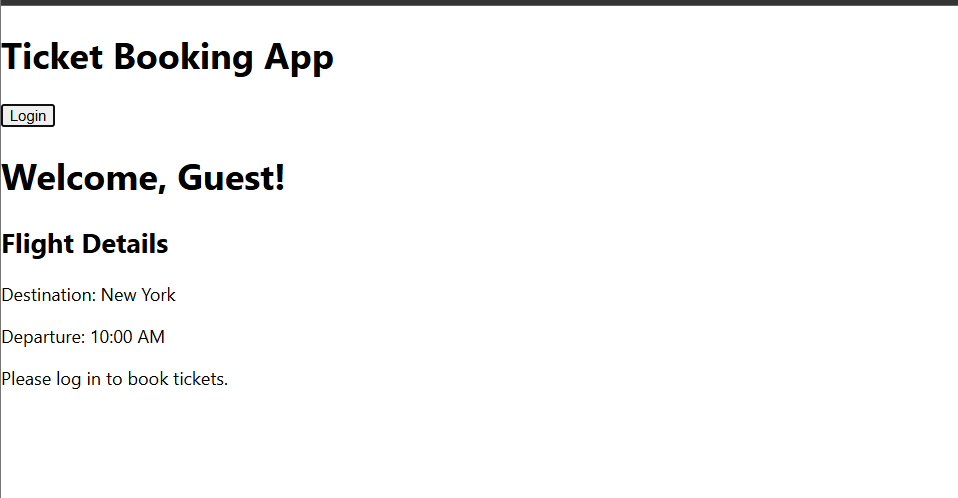
  );

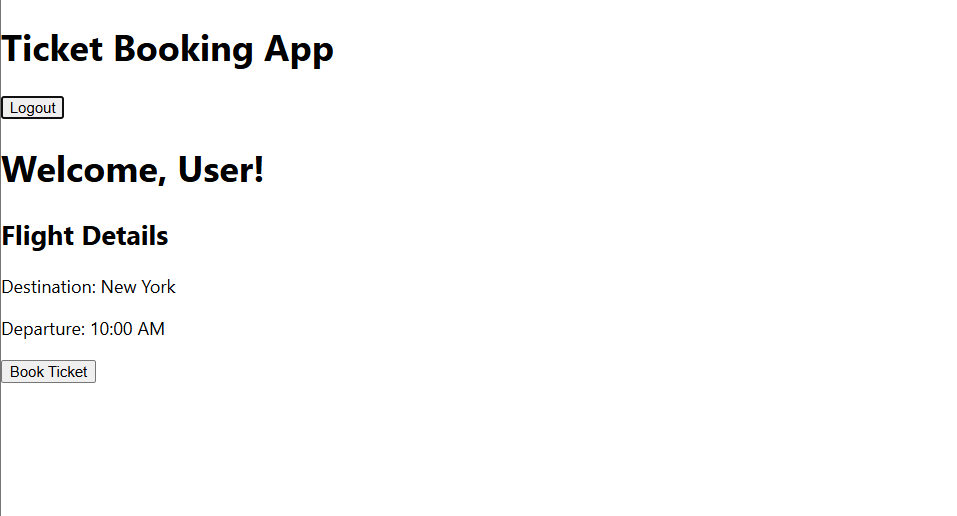
}

export default App;

Now perform npm start to get the result

**OUTPUT:**

****

****

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

Conditional rendering in react means displaying different UI elements based on certain condition like user input

We can use these:

* If statement
* Ternary Operator
* Logical AND
* Switch case
  1. **Explain how to render multiple components**

React allows you to render more than one component at the same time. You can do this by wrapping them in a parent container like <div> or a react Fragment(<> </>)

Ex:

<>

<Header />

<Sidebar />

<Footer />

</>

* 1. **Define list component**

A list component is a component that renders a list of items, usually from an array using the map() function.

Ex:

function ItemList({ items }) {

return (

<ul>

{items.map(item => <li key={item.id}>{item.name}</li>)}

</ul>

);

}

* 1. **Explain about keys in React applications**

Keys are unique identifiers assigned to list elements to help react track which items change, get added or removed.

Advantages:

They help react efficiently update the DOM

Without keys, performance can suffer and UI bugs can appear

* 1. **Explain how to extract components with keys**

When you use map() to render a list, it’s a good practice to extract the list item into its own component and pass the key prop to the outer element.

Ex:

Item.js

function Item({ name }) {

return <li>{name}</li>;

}

List.js

import Item from './Item';

function List({ items }) {

return (

<ul>

{items.map(item => <Item key={item.id} name={item.name} />)}

</ul>

);

}

* 1. **Explain react map, map() function**

The map() function is a built-in JavaScript method used to transform or render each element of an array.

In react, map() is often used to create a list of components from an array of data.

Ex:

const fruits = ["Apple", "Banana", "Cherry"];

return (

<ul>

{fruits.map((fruit, index) => (

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

))}

</ul>

);

**PROGRAM:**

First create a react app then npm install and change the path cd bloggerapp

Now create a folder inside the src named components and those files are

**BlogDetails.js**

import React from 'react';

const BlogDetails = () => {

  return (

    <div>

      <h2>Blog Details</h2>

      <p>Topic: Conditional Rendering</p>

      <p>Author: Suresh Raina</p>

    </div>

  );

};

export default BlogDetails;

**BookDetails.js**

import React from 'react';

const BookDetails = () => {

  return (

    <div>

      <h2>Book Details</h2>

      <p>Title: React Mastery</p>

      <p>Author: Suriya Shiva Kumar</p>

    </div>

  );

};

export default BookDetails;

**CourseDetails.js**

import React from 'react';

const CourseDetails = () => {

  return (

    <div>

      <h2>Course Details</h2>

      <p>Course: React for Beginners</p>

      <p>Instructor: Mahendra Singh Dhoni</p>

    </div>

  );

};

export default CourseDetails;

Now edit the app.js file

**App.js**

import React, { useState } from 'react';

import BookDetails from './components/BookDetails';   *// ✅ default import*

import BlogDetails from './components/BlogDetails';   *// ✅ default import*

import CourseDetails from './components/CourseDetails'; *// ✅ default import*

function App() {

  const [view, setView] = useState('book');

  const renderComponent = () => {

    switch (view) {

      case 'book':

        return <BookDetails />;

      case 'blog':

        return <BlogDetails />;

      case 'course':

        return <CourseDetails />;

      default:

        return <p>Select a valid view.</p>;

    }

  };

  return (

    <div>

      <h1>Blogger App</h1>

      <button onClick={() => setView('book')}>Book</button>

      <button onClick={() => setView('blog')}>Blog</button>

      <button onClick={() => setView('course')}>Course</button>

      {renderComponent()}

      {view === 'book' ? <p>You're viewing books.</p> : null}

      {view === 'course' && <p>This is a course section.</p>}

    </div>

  );

}

export default App;

**Output:**

