**Blogger App**

**REACT**

Vaishnavi

31/7/2025

Conditional rendering in React allows components to render different UI outputs Conditional rendering in React allows developers to dynamically display different UI components or elements based on conditions, similar to if-else in JavaScript. React provides multiple approaches to implement this, including **if-else statements, ternary operators, logical && operator, and element variables**.

In this lab, we will create a **React Application named** bloggerapp that displays one of three components (**Book Details**, **Blog Details**, or **Course Details**) based on a condition. We will demonstrate **various conditional rendering techniques** and the use of **map()** function for rendering lists with **keys**.

**Objective:**

Conditional rendering means showing different UI elements based on conditions like login status, user role, etc. React supports multiple methods to handle this.

**Common ways:**

1. **If-Else Statement**  
   Good for more structured conditions before the return.  
   *Example:*

if (isLoggedIn) {

return <Dashboard />;

} else {

return <LoginPage />;

}

1. **Ternary Operator**  
   Useful inside JSX to choose between two elements.  
   *Example:*

{isAdmin ? <AdminPanel /> : <UserPanel />}

1. **Logical AND (&&)**  
   Render something only if a condition is true.  
   *Example:*

{showNotification && <Notification />}

1. **IIFE (Immediately Invoked Function)**  
   When conditions are complex but you want to use them directly in JSX.  
   *Example:*

{(() => {

if (status === 'loading') return <Loading />;

if (status === 'done') return <Done />;

return <Error />;

})()}

1. **Switch Statement**  
   Clean for handling multiple distinct values.  
   *Example:*

switch (view) {

case 'home': return <Home />;

case 'about': return <About />;

case 'contact': return <Contact />;

default: return null;

}

Rendering multiple components in React means displaying more than one component together within a single parent. This can be done in the following ways:

* Wrap the components in a single enclosing element like a div, section, or React.Fragment (<>...</>).
* Example:

function App() {

return (

<>

<Header />

<MainContent />

<Footer />

</>

);

}

This helps maintain a clean structure and keeps the component tree readable.

A list component is used to display a list of items using the map() function. Instead of hardcoding elements, you loop through an array and render each item dynamically.

*Example:*

function CityList({ cities }) {

return (

<ul>

{cities.map((city, index) => (

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

))}

</ul>

);

}

In React, keys are identifiers used when rendering lists to help the framework keep track of individual elements. They are essential for efficient updating and correct behavior during re-renders

* A key tells React which items changed, were added, or removed.
* Each item in a list should have a unique key.
* The best choice for a key is a stable, unique ID from your data.
* Avoid using array indexes as keys unless there is no other option.

**Example:**

items.map(item => (

<Item key={item.id} value={item} />

))

Using keys correctly ensures that React updates only the changed elements instead of re-rendering the entire list.

When rendering a list, you can split each item into its own component (this is called **component extraction**). In such cases, the key must be assigned **where the list is being mapped**, not inside the child component. This ensures React can track and update each component efficiently.

Assign key in the parent component that calls map(), not in the extracted child component.

**Example:**

function UserCard({ user }) {

return <div>{user.name}</div>;

}

function UserList({ users }) {

return (

<div>

{users.map(user => (

<UserCard key={user.id} user={user} />

))}

</div>

);

}

The map() function is used in React to iterate over an array and return a list of JSX elements. It's one of the most commonly used methods for rendering UI from dynamic data.  
Always provide a unique key when rendering multiple elements using map().

**Different Example:**

const students = ['Arun', 'Beena', 'Chetan'];

const studentList = students.map((student, idx) => (

<div key={idx}>Student: {student}</div>

));

You can use this list inside a component like:

function Classroom() {

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

}

This will render each student’s name inside a div, and React will track each one using the provided key.

**Implementation:**

**Step 1: Create a React App**

****

**Step 2: Use the Component in App.js**

import React, { useState } from 'react';

import BookDetails from './components/BookDetails';

import BlogDetails from './components/BlogDetails';

import CourseDetails from './components/CourseDetails';

function App() {

  const [selected, setSelected] = useState('');

  return (

    <div style={{ textAlign: 'center', marginTop: '40px' }}>

      <h1>Welcome to Blogger App</h1>

      <div>

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

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

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

      </div>

      {selected === 'book' ? <BookDetails /> :

       selected === 'blog' ? <BlogDetails /> :

       selected === 'course' ? <CourseDetails /> :

       <p>Please select an option above.</p>}

    </div>

  );

}

export default App;

**BookDetails.jsx**

import React from 'react';

const BookDetails = () => {

  return (

    <div style={{

      background: '#fff0f5',

      border: '2px dashed #d63384',

      padding: '20px',

      borderRadius: '12px',

      fontFamily: 'Georgia',

    }}>

      <h2 style={{ color: '#c9184a' }}>📚 Featured Novel</h2>

      <p><strong>Title:</strong> The Mystic Garden</p>

      <p><strong>Author:</strong> Elira Moonstone</p>

      <p><strong>Genre:</strong> Fantasy / Romance</p>

    </div>

  );

};

export default BookDetails;

**BlogDetails.jsx**

import React from 'react';

const BlogDetails = () => {

  return (

    <div style={{

      background: '#e0f7fa',

      border: '2px solid #00bcd4',

      padding: '20px',

      borderRadius: '12px',

      fontFamily: 'Verdana',

    }}>

      <h2 style={{ color: '#00796b' }}>📝 Blog Spotlight</h2>

      <p><strong>Title:</strong> Life Between Coffee Breaks</p>

      <p><strong>Writer:</strong> Nia Kulkarni</p>

      <p><strong>Category:</strong> Lifestyle & Minimalism</p>

    </div>

  );

};

export default BlogDetails;

**CourseDetails.jsx**

import React from 'react';

const CourseDetails = () => {

  return (

    <div style={{

      background: '#f3f9d2',

      border: '2px dotted #8bc34a',

      padding: '20px',

      borderRadius: '12px',

      fontFamily: 'Tahoma',

    }}>

      <h2 style={{ color: '#558b2f' }}>🎓 Course Info</h2>

      <p><strong>Course:</strong> UI/UX with Figma</p>

      <p><strong>Instructor:</strong> Devika Sharma</p>

      <p><strong>Duration:</strong> 4 Weeks</p>

    </div>

  );

};

export default CourseDetails;

**App.css**

.body {

  font-family: Arial, sans-serif;

  background-color: #fff;

  padding: 20px;

  margin: 0;

}

.container {

  display: flex;

  justify-content: space-around;

  align-items: flex-start;

  text-align: left;

}

.box {

  border-left: 3px solid green;

  padding-left: 20px;

  width: 30%;

}

h2 {

  text-align: center;

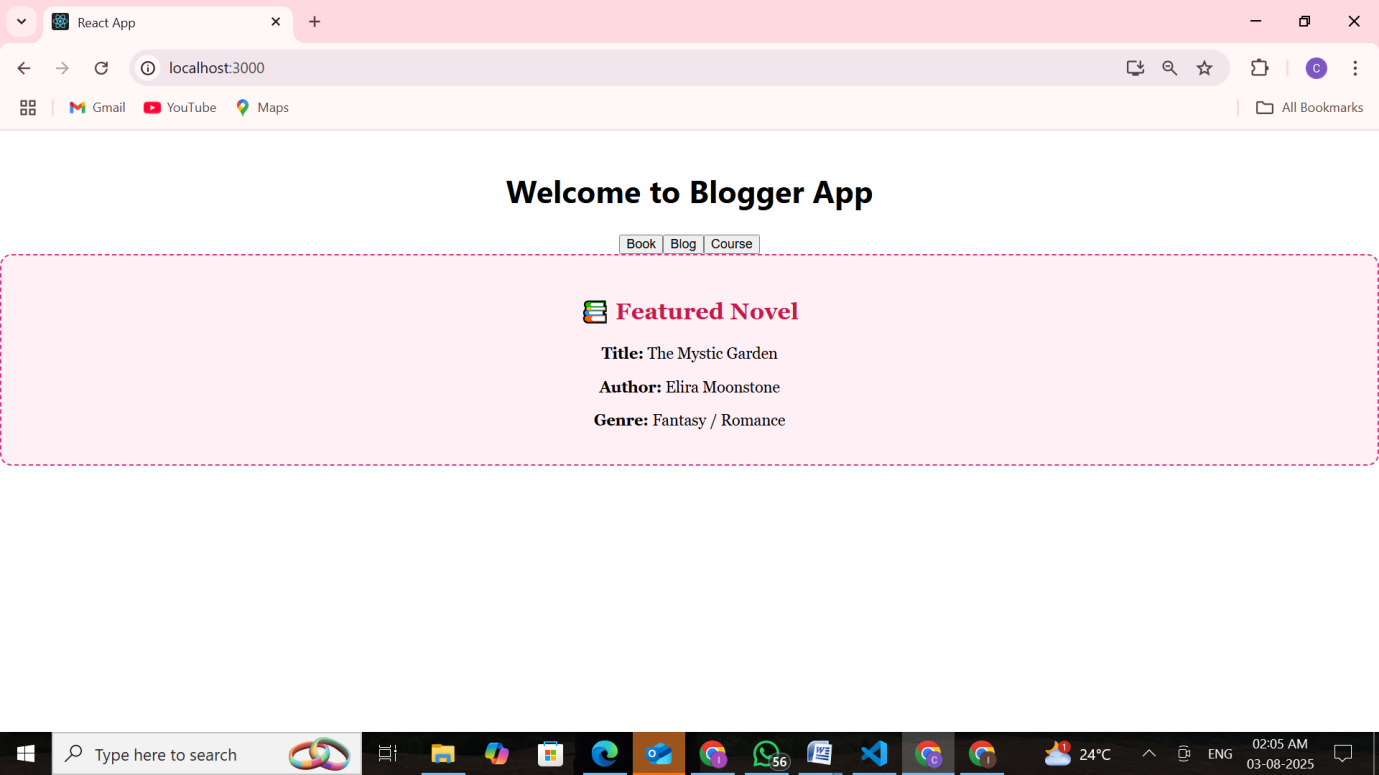
}

**Step 4: Run the App**

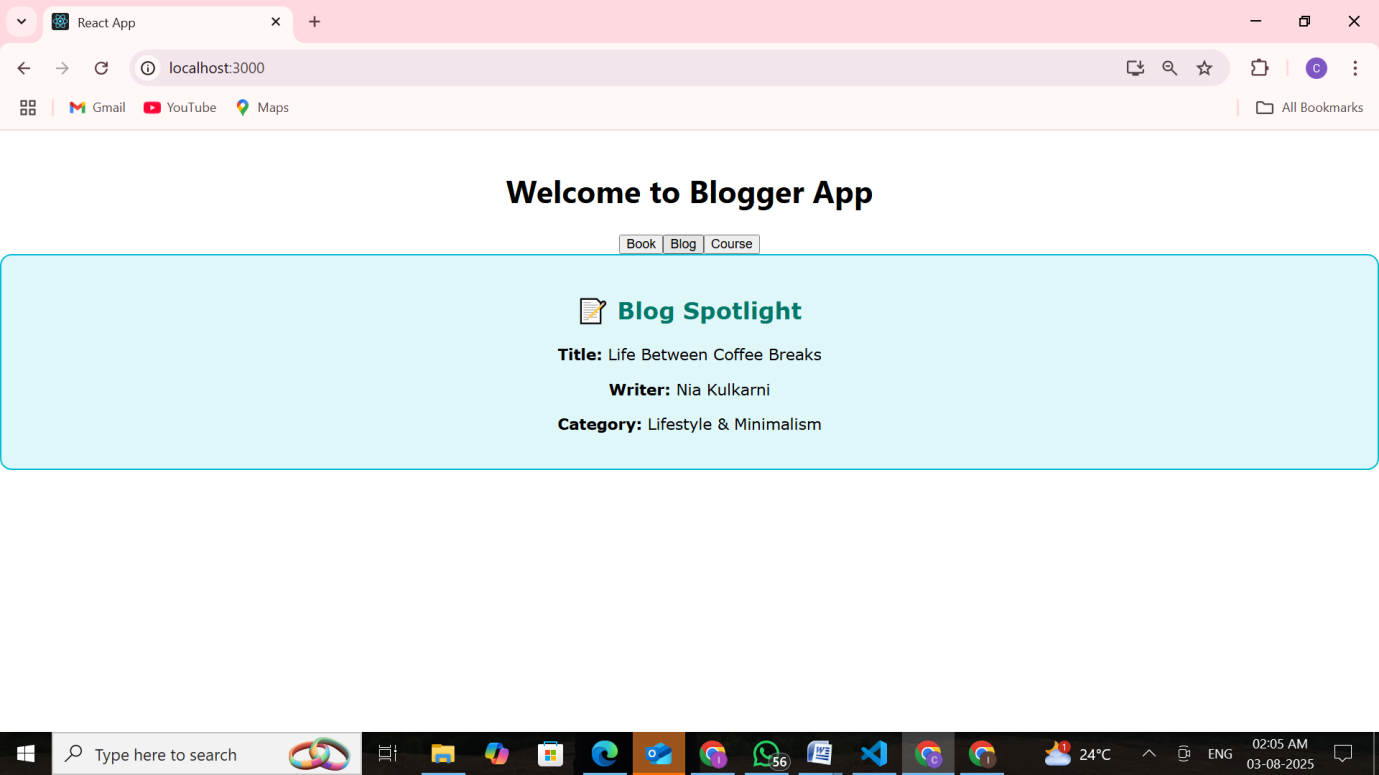
****

**Output:**

Book details:



Blog details:



Course details:

