## **Objectives**

* Explain about conditional rendering in React

Conditional Rendering in React refers to the technique of rendering different UI elements or components based on specific conditions. Just like JavaScript, React allows the use of conditional statements (such as if, else, ternary operators, and logical &&) within components to determine what should be displayed.

* It improves user experience by showing relevant content based on application state.
* React doesn't have built-in if/else templates like some other frameworks; instead, it uses JavaScript expressions directly in JSX.
* **Ternary Operator**:

{isLoggedIn ? <LogoutButton /> : <LoginButton />}

* **Logical AND (&&)**:

{hasPermission && <AdminPanel />}

* **Function or Variable Handling**:  
  Assign JSX to a variable and use logic to conditionally assign content
* Define element variables

**Element Variables** are variables that store JSX elements. They are used in React to manage complex UI logic in a clean and organized way. By defining a JSX element in a variable, you can assign different elements based on logic and then render that variable.

* To improve code readability and structure.
* To simplify conditional rendering logic.
* To prepare content dynamically before returning JSX.

**Example:**

let message;

if (isLoggedIn) {

message = <h1>Welcome back!</h1>;

} else {

message = <h1>Please sign in.</h1>;

}

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

* Explain how to prevent components from rendering

Sometimes in React, you may want to intentionally stop a component or part of a component from rendering. This can help improve performance or handle conditional visibility.

**Techniques to Prevent Rendering:**

1. **Return null**:

Returning null from a component means nothing will be rendered to the DOM.

function Notification({ show }) {

if (!show) return null;

return <div>You have a new message.</div>;

}

1. **Using Conditional Logic**:

Render components only when a condition is true.

{isVisible && <Sidebar />}

1. **shouldComponentUpdate() (Class Components)**:

This lifecycle method can prevent re-rendering based on conditions.

shouldComponentUpdate(nextProps, nextState) {

return false; // Stops the component from updating

}

1. **Using React.memo() (Functional Components)**:

This higher-order component prevents re-rendering unless props change.

const MyComponent = React.memo((props) => {

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

});

## **Notes**

Estimated time to complete this lab: **60 minutes.**

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.





**Hint:**







HandsOn:



