Course Outcome 5 : Create single-page applications using React by employing components, hooks, props, and routing principles, following the MVC design pattern.

Experiment 08 : Program for making use of React Hooks that displays four buttons namely, "Red", "Blue", "Green", "Yellow". On clicking any of these buttons, the code displays the message that you have selected that particular color.

Problem Statement :

The problem at hand is to create a simple React application that demonstrates the use of React Hooks, specifically the useState hook. The application should display four buttons labeled "Red", "Blue", "Green", and "Yellow". When the user clicks on one of these buttons, the program should update the interface to display a message indicating which color was selected.

Solution :

The useState hook is one of the most commonly used React Hooks. It allows developers to add state variables to functional components. When the state changes, React re-renders the component, ensuring the user interface always reflects the latest state.

In this program, four buttons are displayed to the user: "Red", "Blue", "Green", and "Yellow". Each button has an event handler that updates the state with the selected color when clicked. The state is then used to dynamically display the message "You have selected <color>" on the screen. This demonstrates the concept of state management and reactivity in React using Hooks.

Procedure

1. Set up a React application using Create React App.
2. Create a functional component, for example, ColorSelector.
3. Import and use the useState hook to declare a state variable for storing the selected color.
4. Create four buttons labeled "Red", "Blue", "Green", and "Yellow".
5. Attach onClick event handlers to each button to update the state with the corresponding color.
6. Display a message below the buttons that reflects the current state, showing which color is selected.
7. Run the application in the browser to observe the output.

Conclusion

This experiment demonstrates the use of the useState hook in React for managing state in functional components. By clicking on different buttons, the state is updated, and React re-renders the component to reflect the change in the user interface. The implementation showcases the simplicity and efficiency of React Hooks in building interactive applications.