Assignment 7b

Roll No: 71 T14 Batch

Aim: WAP to implement the concept of React Hooks.

LO Attained: LO5

Theory:

React Hooks are a fundamental addition to the React library that revolutionized the way developers manage state and side effects in functional components. Allow developers to use state and other React features without writing class components.

Advantages of React Hooks:

1. **State Management:** React Hooks enable functional components to manage state. The useState hook, for instance, allows developers to add state variables to their components easily. This is a significant departure from the previous class-based approach, making code more concise and readable. It eliminates the need for constructor functions and the this keyword, simplifying the mental model for handling component state.

2. **Side Effects:** React Hooks also address the management of side effects. The useEffect hook is used to perform tasks like data fetching, DOM manipulation, and subscription management. By encapsulating side effects within functional components, Hooks promote a more declarative and predictable approach to handling asynchronous operations. This contrasts with the lifecycle methods used in class components, which can be more error-prone and harder to reason about.

3. **Custom Hooks**: One of the most powerful aspects of React Hooks is the ability to create custom hooks. Custom hooks allow developers to extract and share stateful logic across different components, enhancing code reusability. This promotes the separation of concerns and the creation of more modular and maintainable codebases. Custom hooks can encapsulate complex logic, making it easier to test and reason about.

4. **Simplification of Component Logic**: React Hooks simplify component logic by allowing developers to use multiple state variables and side effects within a single functional component. This reduces the need for higher-order components (HOCs) or render props, which were previously used to share logic between components. Hooks promote a more straightforward and flexible architecture, resulting in cleaner and more maintainable code.

5. **Backward Compatibility**: React Hooks were introduced as additions to React and do not break backward compatibility. This means that developers can gradually adopt Hooks in their existing codebases without rewriting large portions of their application. This smooth transition path has encouraged many developers to embrace Hooks and modernize their React applications.

Features :

Simplicity: Hooks are much simpler to use than class-based components. You don't need to worry about writing constructors, lifecycle methods, or this binding.

Reusability: Hooks can be easily reused across multiple components. This is because they are just functions, and you can pass them as arguments to other functions.

Testability: Hooks are easier to test than class-based components. This is because they are just functions, and you can easily mock them out in your tests.

Some of the most common React Hooks are:

useState: This hook allows you to add state to a function component.

useEffect: This hook allows you to perform side effects, such as making API calls or setting up subscriptions.

useContext: This hook allows you to consume context values from a context provider.

useRef: This hook allows you to get a reference to a DOM element.

Conclusion:

LO5 was achieved. Implemented React Hooks (useState) in a program. React Hooks have fundamentally transformed state and side effect management in React by simplifying component logic, enhancing code reusability with custom hooks, and providing a more declarative approach for handling asynchronous operations.