REACT

MODULE:10 List and Hooks

Q-1)Explain Life cycle in Class Component and functional component with Hooks.

Ans: In React, components are the building blocks of user interfaces. They can be categorized into two types: class components and functional components. Both types have lifecycles, which are phases that a component goes through from its creation to its removal from the DOM **(Document Object Model)**.

1. Lifecycle in Class Components:

Mounting Phase:

- **constructor():** This is the first method called when a component is created. It's used for initializing state and binding event handlers.

- **componentDidMount()**: This method is invoked immediately after a component is mounted (inserted into the DOM). It's often used for fetching data from APIs or initializing any subscriptions.

Updating Phase:

- **componentDidUpdate():** This method is called after the component's updates are flushed to the DOM. It's useful for performing actions after the component re-renders due to changes in props or state.

- **shouldComponentUpdate():** This method allows you to optimize performance by controlling whether the component should re-render when its props or state change.

Unmounting Phase:

- **componentWillUnmount():** This method is invoked immediately before a component is unmounted and destroyed. It's used for

cleanup tasks such as cancelling network requests or removing event listeners.

2. Lifecycle in Functional Components with Hooks:

Mounting Phase:

useState(): This Hook allows functional components to have state variables. It initializes state and provides a function to update it.

useEffect() with an empty dependency array: This Hook replaces componentDidMount and componentDidUpdate. It runs after every render and is used for data fetching, setting up subscriptions, or manually changing the DOM.
Updating Phase:

Updating Phase:

useEffect() with dependencies: You can use useEffect with dependencies to perform actions based on changes to specific props or state variables.

useMemo()and useCallback(): These Hooks can be used to memoize expensive calculations or prevent unnecessary re-renders.

Unmounting Phase:

useEffect() with a cleanup function: By returning a cleanup function from useEffect, you can perform cleanup tasks when a component is unmounted. This is equivalent to componentWillUnmount in class components.

Both class components and functional components with Hooks provide mechanisms for managing component lifecycles. Hooks simplify the process by allowing functional components to use state and lifecycle methods, making code more concise and easier to understand. Additionally, Hooks encourage better code organization and reusability by separating concerns into smaller, composable functio