Assignment: Module – 4 (List & Hook)

Que. **Life cycle in Class Component and functional component with Hooks**

Ans. Each component in React has a lifecycle which you can monitor and manipulate during its three main phases.
The three phases are: Mounting, Updating, and Unmounting.

A React component undergoes three different phases in its lifecycle, including mounting, updating, and unmounting. Each phase has specific methods responsible for a particular stage in a component's lifecycle. These methods are technically particular to class-based components and not intended for functional components.

However, since the concept of Hooks was released in React, you can now use abstracted versions of these lifecycle methods when you're working with functional component state. Simply put, React Hooks are functions that allow you to "hook into" a React state and the lifecycle features within function components.

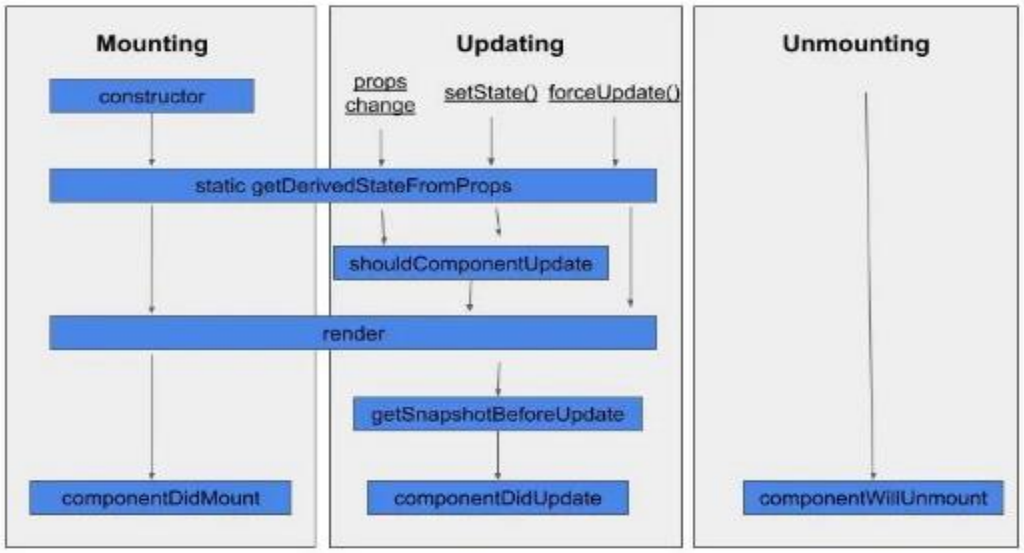
Phases of a React component's lifecycle

A React component undergoes three phases in its lifecycle: mounting, updating, and unmounting.

1. The **mounting** phase is when a new component is created and inserted into the DOM or, in other words, when the life of a component begins. This can only happen once, and is often called "initial render."
2. The **updating** phase is when the component updates or re-renders. This reaction is triggered when the props are updated or when the state is updated. This phase can occur multiple times, which is kind of the point of React.
3. The last phase within a component's lifecycle is the **unmounting** phase, when the component is removed from the DOM.

In a class-based component, you can call different methods for each phase of the lifecycle. These lifecycle methods are of course not applicable to functional components because they can only be written/contained within a class. However, react hooks give functional components the ability to use states. Hooks have gaining popularity because they make working with React cleaner and often less verbose.

React Component Life cycle



The Mounting Phase:

In the mounting phase, a component is prepared for and actually inserted into the DOM. To get through this phase, four lifecycle methods are called: constructor, static getDerivedStateFromProps, render, and componentDidMount.

Constructor Method: The constructor method is the very first method called during the mounting phase.
This method is mostly used for initializing the state of the component and binding event-handler methods within the component. The constructor method is not necessarily required. If you don't intend to make your component stateful (or if that state doesn’t need to be initialized) or bind any method, then it’s not necessary to implement.
The constructor method is called when the component is initiated, but before it’s rendered. It’s also called with props as an argument. It's important you call the super(props) function with the props argument passed onto it within the constructor before any other steps are taken.
This will then initiate the constructor of React.Component (the parent of this class-based component) and it inherits

the constructor method and other methods of a typical React component.

getDerivedStateFromProps: Props and state are completely different concepts, and part of building your app intelligently is deciding which data goes where.
In many cases though, your component's state will be *derivative* of its props. This is where the static getDerivedStateFromProps method comes in. This method allows you to modify the state value with any props value. It's most useful for changes in props over time, and we'll learn later that it's also useful in the update phase.
The method static getDerivedStateFromProps accepts two arguments: props and state, and returns an object, or null if no change is needed. These values are passed directly to the method, so there's no need for it to have access to the instance of the class (or any other part of the class) and thus is considered a static method.

Render Method: The render method is the only required method for a class-based React component. It's called after the getDerivedStateFromProps method and actually renders or inserts the HTML to the DOM.

componentDidMount Method: componentDidMount is the last lifecycle method called in the mounting phase. It's called right after the component is rendered or mounted to the DOM.
With this method, you're allowed to add side effects like sending network requests or updating the component's state. Additionally, the componentDidMount method allows you to make subscriptions like subscribing to the Redux store. You can also call the this.setState method right away; however this will cause a re-render as it kicks in the update phase, since the state has changed.

The Updating Phase

The Updating phase is triggered when component props or state change, and consists of the following methods: static getDerivedFromProps, shouldComponentUpdate, render, getSna pshotBeforeUpdate, and componentDidUpdate.

The methods getDerivedFromProps and render are also part of the mounting phase. Since they've been covered previously, this section focuses on the other three methods.

static getDerivedFromProps: In the update phase, the first lifecycle method called is getDerivedStateFromProps. This method is useful if you have updated props and you want to reflect that in the component's state.
For instance, a component's state may depend on the value of its props. With getDerivedStateFromProps, before the component was even re-rendered, its state can reflect those changes and can be shown (if applicable) to the newly updated component.

shouldComponentUpdate: shouldComponentUpdate is another rarely used lifecycle method. It's specifically intended for performance optimization, and basically lets you tell React when you don't need to re-render when a new state or props comes in. While it can help avoid re-renders, you shouldn't rely on it to prevent re-renders since you might skip a necessary update and encounter bugs.
This method can accept nextProps and nextState as arguments, however, they're optional, and you can declare it without the arguments. This method then returns a Boolean value. The Boolean value defines whether a re-render happens. The default value is true, where re-render happens in all cases whenever state or props changes.

getSnapshotBeforeUpdate:The getSnapshotBeforeUpdate meth od gives you access to the previous props and state of the component before it's updated. This allows you to work or check on the previous values of the state or props. It's another method that's rarely used.
A good use case for this method is handling scroll positions in a chat app. When a new message comes in as the user is viewing old messages, it shouldn't push the old ones out of view.

componentDidUpdate: The componentDidUpdate method is the last lifecycle method invoked in the update phase. It allows you to create side effects like sending network requests or calling the this.setState method. It’s important to remember that there should always be a way to avoid the setState (like some sort of logic), or it will result in an infinite loop of re-rendering. This method can accept up to three parameters: prevProps, prevState, and snapshot (if you implement the getSnapshotBeforeUpdate method).

<u>Unmounting Phase:</u>

The unmounting phase is the third and final phase of a React component. At this phase, the component is removed from the DOM. Unmounting only has one lifecycle method involved: componentWillUnmount.

componentWillUnmount: componentWillUnmount is invoked right before the component is unmounted or removed from the DOM. It’s meant for any necessary clean-up of the component, like unsubscribing to any subscriptions or canceling any network requests. Once this method is done executing, the component will be destroyed.