### ****Objective Answers – Week6\_HandsOn4****

#### 1. Explain the need and Benefits of component life cycle

In React, the **component lifecycle** represents the different phases a component goes through from creation to removal from the DOM. Understanding these phases is crucial because it allows developers to execute code at specific points in a component's life — such as when the component is mounted, updated, or unmounted.

**Benefits of lifecycle methods:**

* Allows fetching data at the right moment (componentDidMount)
* Enables cleanup of resources like timers or subscriptions (componentWillUnmount)
* Helps in handling errors gracefully (componentDidCatch)
* Gives developers fine-grained control over rendering behavior
* Improves performance and efficiency by avoiding unnecessary operations

Lifecycle methods are especially important in class components, where they provide structured points to plug in logic tied to the component's behavior.

#### 2. Identify various life cycle hook methods

React class components are structured around **three major lifecycle phases**, each with specific methods:

##### ****Mounting Phase**** (component is created and inserted into the DOM)

* constructor() – Initializes the component’s state and binds methods
* static getDerivedStateFromProps() – Syncs props with state before rendering
* render() – Returns the JSX to be displayed
* componentDidMount() – Invoked once after the component is rendered; used for API calls, timers, subscriptions

##### ****Updating Phase**** (component re-renders due to state/props change)

* static getDerivedStateFromProps() – Called again before each re-render
* shouldComponentUpdate() – Optimizes performance by controlling re-rendering
* render() – Updates the UI
* getSnapshotBeforeUpdate() – Captures info before DOM updates
* componentDidUpdate() – Acts after updates are flushed to the DOM

##### ****Unmounting Phase**** (component is removed from DOM)

* componentWillUnmount() – Cleanup logic goes here (timers, listeners, etc.)

##### ****Error Handling****

* componentDidCatch(error, info) – Catches rendering errors in child components and prevents app crashes.

#### 3. List the sequence of steps in rendering a component

The **rendering process** of a class component in React follows a specific sequence, particularly during mounting and updating:

### Mounting (when the component is being added to the DOM):

* constructor() – Initializes state and props
* getDerivedStateFromProps() (optional)
* render() – JSX is generated
* **React updates the DOM**
* componentDidMount() – Ideal for API calls and side effects

### Updating (when state or props change):

* getDerivedStateFromProps() (optional)
* shouldComponentUpdate() (optional, for optimization)
* render() – Re-renders the component
* getSnapshotBeforeUpdate() (optional)
* componentDidUpdate() – Called after DOM update

### Unmounting (when the component is removed):

* componentWillUnmount() – Used for cleanup logic

These steps ensure React components behave predictably and allow developers to inject logic at the appropriate lifecycle stages for clean, efficient UI updates.