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

The component lifecycle in React is essential for managing how components are created, updated, and destroyed. It provides developers with methods (also called lifecycle hooks) that can be overridden to run code at specific points during a component's life.

* Need:
* React components go through multiple phases: mounting, updating, and unmounting.
* Each phase offers a unique opportunity to perform specific tasks such as fetching data, manipulating the DOM, or cleaning up resources.
* Benefits:
* Efficient Resource Management: Lifecycle methods help in managing memory and resources efficiently by allowing clean-up before a component is removed.
* Control over Rendering: Developers can control when a component should re-render, optimizing performance.
* Data Fetching and Initialization: Data can be fetched at the right time (e.g., right after the component is mounted).
* Error Handling: Lifecycle methods like componentDidCatch() allow catching and handling errors gracefully.

## 2. Identify various life cycle hook methods

Lifecycle hook methods in React class components include:

* Mounting Phase:
* constructor()  
  static getDerivedStateFromProps()  
  render()  
  componentDidMount()
* Updating Phase:
* static getDerivedStateFromProps()  
  shouldComponentUpdate()  
  render()  
  getSnapshotBeforeUpdate()  
  componentDidUpdate()
* Unmounting Phase:
* componentWillUnmount()
* Error Handling:
* componentDidCatch()  
  static getDerivedStateFromError()

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

When a React class component is rendered, it goes through the following sequence:

* Mounting (when the component is created and inserted into the DOM):
* 1. constructor()  
  2. static getDerivedStateFromProps()  
  3. render()  
  4. componentDidMount()
* Updating (when props or state changes):
* 1. static getDerivedStateFromProps()  
  2. shouldComponentUpdate()  
  3. render()  
  4. getSnapshotBeforeUpdate()  
  5. componentDidUpdate()
* Unmounting (when the component is removed from the DOM):
* 1. componentWillUnmount()
* Error Handling (if an error occurs during rendering or in a lifecycle method):
* 1. static getDerivedStateFromError()  
  2. componentDidCatch()