**4. ReactJS – HOL**

**1. Need and Benefits of Component Lifecycle**

**Why it is needed?**

* To control what happens when a component is created, updated, or removed.

**Benefits:**

* Load data (like from an API).
* Update UI when data changes.
* Clean up memory (like stopping a timer).
* Keeps the app efficient and organized.

**2. Lifecycle Hook Methods (Class Components)**

React has **3 main phases** with these methods:

**🔹 Mounting (Component is created):**

* constructor() – Setup things.
* render() – Show the UI.
* componentDidMount() – Runs after the UI appears (good for API calls).

**🔹 Updating (Component changes):**

* shouldComponentUpdate() – Should it update?
* render() – Show updated UI.
* componentDidUpdate() – After update, do something.

**🔹 Unmounting (Component removed):**

* componentWillUnmount() – Cleanup before it's gone (like clearing intervals).

**3. Steps in Rendering a Component**

**When a component is first shown (mounting):**

1. constructor()
2. render()
3. componentDidMount()

**When it updates (state or props change):**

1. shouldComponentUpdate()
2. render()
3. componentDidUpdate()

**When it is removed (unmounting):**

* componentWillUnmount()

**React Application : blogapp**

1. Create a new react application using *create-react-app* tool with the name as “blogapp”
2. Open the application using VS Code
3. Create a new file named as **Post.js** in **src folder** with following properties



*Figure 2: Post class*

1. Create a new class based component named as **Posts** inside **Posts.js** file



*Figure 3: Posts Component*

1. Initialize the component with a list of Post in state of the component using the constructor
2. Create a new method in component with the name as **loadPosts()** which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (<https://jsonplaceholder.typicode.com/posts>)



*Figure 4: loadPosts() method*

1. Implement the **componentDidMount()** hook to make calls to **loadPosts()** which will fetch the posts



*Figure 5: componentDidMount() hook*

1. Implement the **render()** which will display the title and post of posts in html page using heading and paragraphs respectively.



*Figure 6: render() method*

1. Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.



*Figure 7: componentDidCatch() hook*

1. Add the Posts component to App component.
2. Build and Run the application using *npm start* command.

**Final Output:**

