**React Hands On 4:**

Q. Explain the need and Benefits of component life cycle

The component life cycle in React is the sequence of events a component goes through from creation to removal.  
It helps us control what happens when a component mounts, updates, or unmounts.  
The main benefit is that we can run code at specific times, like fetching data when the component appears or cleaning up resources when it’s removed.  
This makes our app more efficient, predictable, and easier to manage as it grows.

Q. Identify various life cycle hook methods

React class component life cycle hook methods:

1. constructor() — initializes state and binds methods when the component is created.
2. render() — returns the JSX to display the UI.
3. componentDidMount() — runs after the component is added to the DOM, good for fetching data or starting timers.
4. componentDidUpdate() — runs after the component’s updates are flushed to the DOM, useful for reacting to prop or state changes.
5. componentWillUnmount() — runs just before the component is removed, used to clean up timers, subscriptions, or listeners.

Q. List the sequence of steps in rendering a component

Steps in rendering a React class component:

1. constructor() runs to initialize state and props.
2. render() runs to return the JSX structure.
3. React updates the virtual DOM and compares it with the real DOM (diffing).
4. componentDidMount() runs after the component is inserted into the DOM.

Hands On ex:  
Post.js:

class Post {

  constructor(id, title, body) {

    this.id = id;

    this.title = title;

    this.body = body;

  }}

export default Post;

Posts.js:

import React from 'react';

import Post from './Post';

class Posts extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: []

    };}

  loadPosts() {

    fetch('https://jsonplaceholder.typicode.com/posts')

      .then(response => response.json())

      .then(data => {

        const postList = data.map(item => new Post(item.id, item.title, item.body));

        this.setState({ posts: postList });

      })

      .catch(error => {

        console.error('Error fetching posts:', error);

      });}

  componentDidMount() {

    this.loadPosts(); }

  componentDidCatch(error, info) {

    alert('An error occurred: ' + error); }

  render() {

    return (

      <div>

        <h2>Posts</h2>

        {this.state.posts.map(post => (

          <div key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.body}</p>

            <hr />

          </div>

        ))}

      </div>

    );}}

export default Posts;

Website screenshot:

