**Exercise 4 REACTJS-HOL**

**OBJECTIVES**

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

The component lifecycle helps manage the different phases a component goes through—from creation to removal. It allows developers to perform actions at specific stages like fetching data, updating the DOM, or cleaning up. Enables better memory management by freeing up unused resources. Improves application performance by controlling re-renders and updates efficiently. Enhances debugging and maintenance by knowing when and why components behave a certain way.

1. **Identify various life cycle hook methods**

**constructor()** – Initializes state and binds methods.

**componentDidMount()** – Invoked once after the component is mounted; used for API calls or DOM operations.

**shouldComponentUpdate()** – Determines if a component should re-render on state/prop changes.

**componentDidUpdate()** – Runs after component updates; used to react to prop/state changes.

**componentWillUnmount()** – Cleanup tasks before the component is destroyed (like removing event listeners).

1. **List the sequence of steps in rendering a component**

**Constructor**: Initializes state and props.

**Render**: Returns the JSX to describe the UI.

**React Updates the DOM** based on JSX.

**componentDidMount**: Runs after the component is added to the DOM.

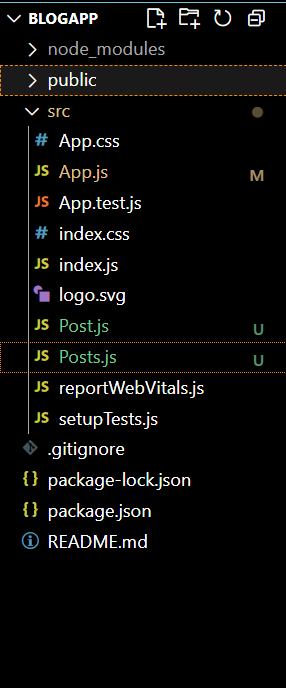
**On updates**, React runs shouldComponentUpdate, then render, then componentDidUpdate.

**HANDS ON PRACTICE**

1. **Create a new react application using *create-react-app* tool with the name as “blogapp”**

npx create-react-app blogapp

**CODE STRUCTURE:**



1. **Create a new file named as Post.js in src folder with following properties**

**Post.js**

class Post{

    constructor(id,title,body){

        this.id=id;

        this.title=title;

        this.body=body;

    }

}

export default Post;

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

**Posts.js**

import React from 'react';

import PostComponent from './Post'; // PostComponent renders each post

class Posts extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [],

      hasError: false

    };

  }

  // Method to load posts from API

  loadPosts() {

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

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

      .then((data) => {

        const postObjects = data.slice(0, 10).map(post => ({

          id: post.id,

          title: post.title,

          body: post.body

        }));

        this.setState({ posts: postObjects });

      })

      .catch((error) => {

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

        this.setState({ hasError: true });

      });

  }

  // Lifecycle hook

  componentDidMount() {

    this.loadPosts();

  }

  // Error boundary

  componentDidCatch(error, info) {

    alert("An error occurred in Posts component.");

    console.error("Caught error:", error, info);

    this.setState({ hasError: true });

  }

  // Render posts

  render() {

    if (this.state.hasError) {

      return <h2>Something went wrong.</h2>;

    }

    return (

      <div>

        <h1>Blog Posts</h1>

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

          <PostComponent key={post.id} title={post.title} body={post.body} />

        ))}

      </div>

    );

  }

}

export default Posts;

1. **Output**

