**4. ReactJS-HOL**

**Objectives:**

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

The component life cycle in React defines the different stages a component goes through from its creation to its destruction. These stages include mounting (insertion into the DOM), updating (re-rendering), and unmounting (removal from the DOM). Understanding these life cycle phases allows developers to execute code at specific points, such as fetching data, setting timers, or cleaning up resources.  
The benefits include:

* Improved code control
* Optimized performance
* Predictable behavior of components
* Proper resource management using cleanup techniques

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

React class components provide several lifecycle hook methods grouped into three phases:

**Mounting** (when the component is created and inserted into the DOM):

* constructor()
* static getDerivedStateFromProps()
* render()
* componentDidMount()

**Updating** (when the component re-renders due to changes in props or state):

* static getDerivedStateFromProps()
* shouldComponentUpdate()
* render()
* getSnapshotBeforeUpdate()
* componentDidUpdate()

**Unmounting** (when the component is removed from the DOM):

* componentWillUnmount()

**Error Handling**:

* componentDidCatch()
* static getDerivedStateFromError()

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

During the mounting phase, the sequence of lifecycle method calls is:

1. constructor()
2. static getDerivedStateFromProps()
3. render()
4. componentDidMount()

During the updating phase (when props or state changes), the order is:

1. static getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

During unmounting:

1. componentWillUnmount()

**Code:  
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(post => new Post(post.id, post.title, post.body));

        this.setState({ posts: postList });

      })

      .catch(error => {

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

      });

  }

  componentDidMount() {

    this.loadPosts();

  }

  render() {

    return (

      <div>

        <h1>Posts</h1>

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

          <div key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.body}</p>

          </div>

        ))}

      </div>

    );

  }

  componentDidCatch(error, info) {

    alert("An error occurred: " + error.toString());

  }

}

export default Posts;

**App.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(post => new Post(post.id, post.title, post.body));

        this.setState({ posts: postList });

      })

      .catch(error => {

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

      });

  }

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

    alert("An error occurred: " + error.toString());

  }

  render() {

    return (

      <div>

        <h1>Posts</h1>

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

          <div key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.body}</p>

          </div>

        ))}

      </div>

    );

  }

}

export default Posts;

**Output:** 

