React Hands-on Lab – BlogApp: Component Lifecycle Hooks

# Objective:

Create a React application named "blogapp" to demonstrate the use of component lifecycle methods such as componentDidMount() and componentDidCatch() using class-based components.

# Learning Objectives:

* • Explain the need and Benefits of component life cycle
* • Identify various life cycle hook methods
* • List the sequence of steps in rendering a component
* • Implement componentDidMount() hook
* • Implement componentDidCatch() life cycle hook

# Prerequisites:

- Node.js  
- npm  
- Visual Studio Code

# Steps to Complete the Hands-on Lab:

## Step 1: Create React Application

Run the following command in terminal:  
  
npx create-react-app blogapp

## Step 2: Open Project in VS Code

Navigate into the project folder and open with VS Code:  
  
cd blogapp  
code .

## Step 3: Create Post.js

In the "src" folder, create a new file named "Post.js" and define the Post class as a simple data holder:  
  
class Post {  
 constructor(id, title, body) {  
 this.id = id;  
 this.title = title;  
 this.body = body;  
 }  
}  
  
export default Post;

## Step 4: Create Posts Component

Create "Posts.js" inside the "src" folder. This will be a class component:  
  
import React, { Component } from 'react';  
import Post from './Post';  
  
class Posts extends Component {  
 constructor(props) {  
 super(props);  
 this.state = { posts: [] };  
 }  
  
 loadPosts = () => {  
 fetch("https://jsonplaceholder.typicode.com/posts")  
 .then(response => response.json())  
 .then(data => this.setState({ posts: data }))  
 .catch(error => console.error("Error:", error));  
 }  
  
 componentDidMount() {  
 this.loadPosts();  
 }  
  
 componentDidCatch(error, info) {  
 alert("An error occurred: " + error);  
 }  
  
 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;

## Step 5: Update App.js

Import and use the Posts component:  
  
import React from 'react';  
import Posts from './Posts';  
  
function App() {  
 return (  
 <div>  
 <Posts />  
 </div>  
 );  
}  
  
export default App;

## Step 6: Run the Application

Open the terminal and execute the following command:  
  
npm start  
  
The browser will automatically open at http://localhost:3000 and display the list of blog posts.

# Output:

Once the application is successfully running, you will see the heading "Posts" followed by a list of post titles and their content.  
  
Each post will be rendered with:  
- Title as a heading  
- Body as a paragraph  
  
If any error occurs during rendering, it will be caught and displayed using an alert message via componentDidCatch().