React Component Lifecycle – Blog Posts Lab

# Defined Objectives

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

Component lifecycle methods in React allow developers to run code at particular times in the component's life. This improves performance, handles errors, and supports data loading.

## 2. Identify various life cycle hook methods

Key Lifecycle Methods:  
- componentDidMount(): Runs after component is mounted.  
- componentDidUpdate(): Runs after component updates.  
- componentWillUnmount(): Runs before component is removed.  
- componentDidCatch(): Catches errors in rendering or lifecycle methods.

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

1. Constructor  
2. getDerivedStateFromProps (optional)  
3. Render  
4. componentDidMount  
5. Updates with componentDidUpdate  
6. Unmount with componentWillUnmount

# Hands-on Lab Instructions

In this hands-on lab, you will learn how to:

· Implement componentDidMount() hook

· Implement componentDidCatch() lifecycle hook

# Prerequisites

The following is required to complete this hands-on lab:

· Node.js

· NPM

· Visual Studio Code

# Notes

Estimated time to complete this lab: 60 minutes.

# Steps to Create Blog App with Lifecycle Methods

1. Create a new react application using create-react-app tool with the name as 'blogapp'  
  
npx create-react-app blogapp

2. Open the application using Visual Studio Code.

3. Create a new file named as Post.js in src folder with following properties (title and body).

4. Create a new class-based component named Posts inside Posts.js file.

5. Initialize the component with a list of posts in the state using the constructor.

6. Create a new method in the component with the name loadPosts() which will fetch posts using Fetch API from:  
  
https://jsonplaceholder.typicode.com/posts

7. Implement the componentDidMount() hook to call loadPosts() and fetch the post data.

8. Implement the render() method to display the post titles and body using headings and paragraphs.

9. Define a componentDidCatch() method to catch and display any component errors as alert messages.

10. Add the Posts component to App.js and render it inside the App component.

11. Build and run the application using the following command:  
  
npm start

# Screenshot of Output

