**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

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

* Implement componentDidMount() hook
* Implementing componentDidCatch() life cycle 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.**

1. Create a new react application using *create-react-app* tool with the name as “blogapp”
2. Open the application using VS Code
3. Create a new file named as **Post.js** in **src folder** with following properties



Figure 2: Post class

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



Figure 3: Posts Component

1. Initialize the component with a list of Post in state of the component using the constructor
2. Create a new method in component with the name as **loadPosts()** which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (<https://jsonplaceholder.typicode.com/posts>)



Figure 4: loadPosts() method

1. Implement the **componentDidMount()** hook to make calls to **loadPosts()** which will fetch the posts



Figure 5: componentDidMount() hook

1. Implement the **render()** which will display the title and post of posts in html page using heading and paragraphs respectively.



Figure 6: render() method

1. Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.



Figure 7: componentDidCatch() hook

1. Add the Posts component to App component.
2. Build and Run the application using *npm start* command.

PROGRAM:

import React from 'react';

import Post from './Post';

import './Posts.css';

class Posts extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [],

      error: null

    };

  }

  loadPosts() {

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

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

      .then(data => {

        const posts = data.map(p => new Post(p.id, p.title, p.body));

        this.setState({ posts });

      })

      .catch(error => {

        this.setState({ error });

      });

  }

  componentDidMount() {

    this.loadPosts();

  }

  render() {

    if (this.state.error) {

      return <h2 className="error">Something went wrong!</h2>;

    }

    return (

      <div className="post-container">

        <h1 className="heading">Blog Posts</h1>

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

          <div key={post.id} className="post-card">

            <h2>{post.title}</h2>

            <p>{post.body}</p>

          </div>

        ))}

      </div>

    );

  }

  componentDidCatch(error, info) {

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

  }

}

export default Posts;

OUTPUT:

