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

1. Explain the need and Benefits of component life cycle

The component lifecycle refers to the **stages** a React component goes through—from creation to removal. Understanding it helps you:

- **Control behavior** at specific stages (e.g. fetch data after render)
- **Optimize performance** by cleaning up resources (e.g. intervals, subscriptions)
- **Handle side effects** safely (like updating the DOM or interacting with external APIs)
- **Debug effectively** by knowing when things happen (and why!)

2. Identify various Lifecycle Hook Methods

React lifecycle hooks vary between **class components** and **function components**.

Class Component Lifecycle Methods:

Phase	Hook Method	Purpose
Mounting	constructor()	Initialize state & props
	componentDidMount()	Run code after component renders
Updating	componentDidUpdate()	Respond to prop or state changes
Unmounting	componentWillUnmount()	Cleanup before component is removed
Error Handling	componentDidCatch()	Handle errors in child components

Function Component Lifecycle (with Hooks):

Hook	Equivalent Purpose
useEffect()	Acts like componentDidMount, DidUpdate, & WillUnmount depending on how it's used

Hook	Equivalent Purpose
useLayoutEffect()	Similar to useEffect, but fires earlier for DOM reads/writes
useCallback()	Memoizes functions between renders
useMemo()	Optimizes expensive calculations

3. List the sequence of Steps in Rendering a Component

For **class components**, the order looks like this:

1. **constructor()** – Initial setup
2. **getDerivedStateFromProps()** *(optional)* – Sync state with props
3. **render()** – Describe the UI
4. **componentDidMount()** – Perform post-render logic

On updates:

1. **getDerivedStateFromProps()**
2. **shouldComponentUpdate()** *(optional)* – Skip re-render if needed
3. **render()**
4. **componentDidUpdate()**

On removal:

1. **componentWillUnmount()** – Cleanup

For **function components**, React handles most steps under the hood, but you control behavior with hooks like useEffect.

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

```
Post.js U
1   class Post {
2       constructor(id, title, body){
3           this.id=id;
4           this.title=title;
5           this.body=body;
6       }
7   }
8   export default Post;
```

Figure 1: Post class

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

```
Posts.js U
1  class Posts extends React.Component {
2      constructor(props){
3          super(props);
4      }
5  }
```

Figure 2: Posts Component

5. Initialize the component with a list of Post in state of the component using the constructor
6. 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)

```
JS Posts.js U
1  class Posts extends React.Component {
2      constructor(props){
3          super(props);
4          //code
5      }
6      loadPosts() {
7          //code
8      }
9  }
```

Figure 3: loadPosts() method

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

```
JS Posts.js U
1  class Posts extends React.Component {
2      constructor(props){
3          super(props);
4          //code
5      }
6      loadPosts() {
7          //code
8      }
9      componentDidMount() {
10         //code
11     }
12 }
```

Figure 4: componentDidMount() hook

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

```
JS Posts.js U
1   class Posts extends React.Component {
2   >     constructor(props) {···
5         }
6   >     loadPosts() {···
8         }
9   >     componentDidMount() {···
11        }
12        render() {
13            //code
14        }
15  }
```

Figure 5: render() method

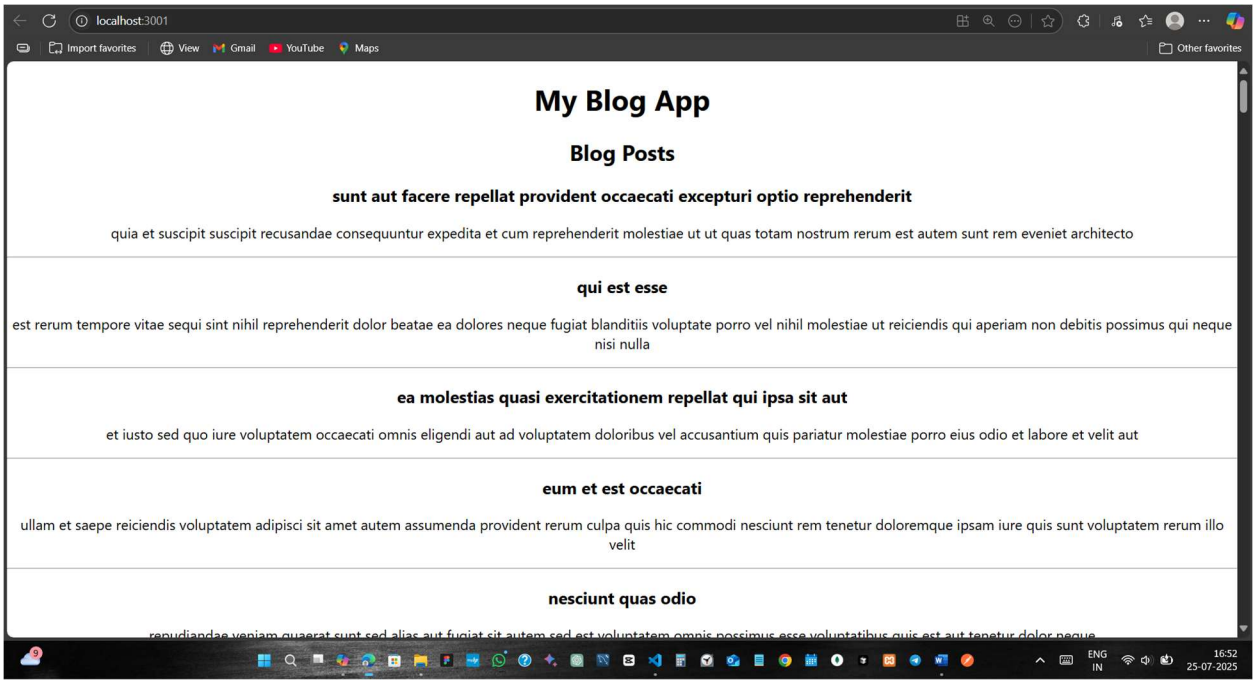
9. Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.

```
JS Posts.js U
1   class Posts extends React.Component {
2   >     constructor(props) {···
5         }
6   >     loadPosts() {···
8         }
9   >     componentDidMount() {···
11        }
12  >     render() {···
14        }
15        componentDidCatch(error, info) {
16            //code
17        }
18  }
```

Figure 6: componentDidCatch() hook

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

Result:


localhost:3001
Import favorites
View
Gmail
YouTube
Maps
Other favorites
My Blog App
Blog Posts
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto
qui est esse
est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit
nesciunt quas odio
ENG
IN
16:52
25-07-2025