WEEK 6

Mandatory Hands-On

**REACT**

**Create a new react application using *create-react-app* tool with the name as “blogapp”**

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

The component lifecycle helps manage the different phases a component goes through—from creation to removal. It allows developers to perform actions at specific stages like fetching data, updating the DOM, or cleaning up. Enables better memory management by freeing up unused resources. Improves application performance by controlling re-renders and updates efficiently. Enhances debugging and maintenance by knowing when and why components behave a certain way.

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

constructor()– Initializes state and binds methods.

componentDidMount()– Invoked once after the component is mounted; used for API calls or DOM operations.

shouldComponentUpdate()– Determines if a component should re-render on state/prop changes.

componentDidUpdate() – Runs after component updates; used to react to prop/state changes.

componentWillUnmount() – Cleanup tasks before the component is destroyed (like removing event listeners).

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

**Constructor**: Initializes state and props.

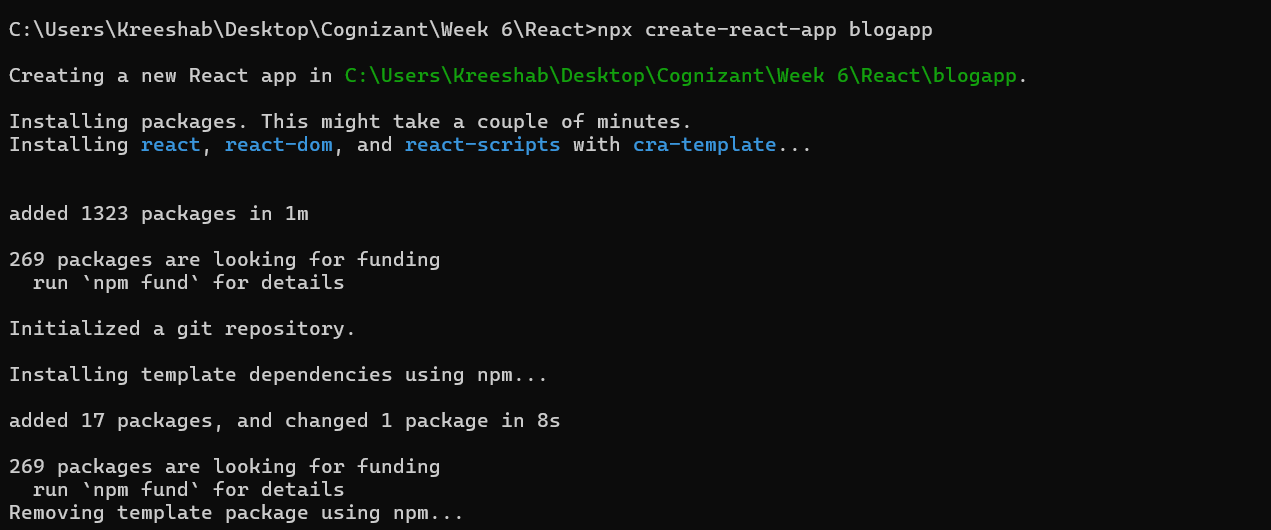
**Render**: Returns the JSX to describe the UI.

**React Updates the DOM** based on JSX.

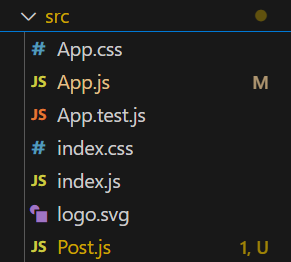
**componentDidMount**: Runs after the component is added to the DOM.

**On updates**, React runs shouldComponentUpdate, then render, then componentDidUpdate.

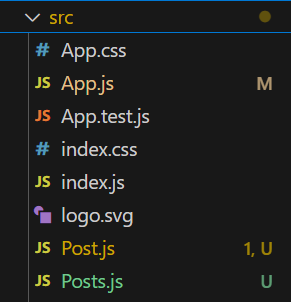
* *Blogapp* application is created in the directory Week6\ React\ blogapp using command *npx create-react-app blogapp.*



* Navigated to React app and created Post.js file inside the src.

****

* Navigated to React app and created Posts.js file inside the src.



**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: [

        new Post(1, 'Post Title 1', 'This is the body of post 1'),

        new Post(2, 'Post Title 2', 'This is the body of post 2'),

        new Post(3, 'Post Title 3', 'This is the body of post 3')

      ]

    };

  }

  loadPosts = async () => {

    try {

      const response = await fetch('https://jsonplaceholder.typicode.com/posts');

      if (!response.ok) {

        throw new Error('Network response was not ok');

      }

      const postsData = await response.json();

      this.setState({ posts: postsData });

    } catch (error) {

      console.error('Failed to load 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}>

            <h2>{post.title}</h2>

            <p>{post.body}</p>

          </div>

        ))}

      </div>

    );

  }

}

export default Posts;

**App.js:**

import React from 'react';

import Posts from './Posts';

function App() {

return (

<div>

<Posts />

</div>

);

}

export default App;

**Sample Output:**



