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

```
C:\WINDOWS\system32\cmd.

C:\Users\HP>npx create-react-app blogapp

Creating a new React app in C:\Users\HP\blogapp.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1323 packages in 2m

269 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.

Installing template dependencies using npm...

added 18 packages, and changed 1 package in 14s

269 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...


removed 1 package, and audited 1341 packages in 9s

269 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Created git commit.

Success! Created blogapp at C:\Users\HP\blogapp
Inside that directory, you can run several commands:

  npm start
    Starts the development server.
```

2. Open the application using VS Code
3. Create a new file named as **Post.js** in **src folder** with following properties

```
JS Post.js  U

class Post {
    constructor(id, title, body){
        this.id=id;
        this.title=title;
        this.body=body;
    }
}
export default Post;
```

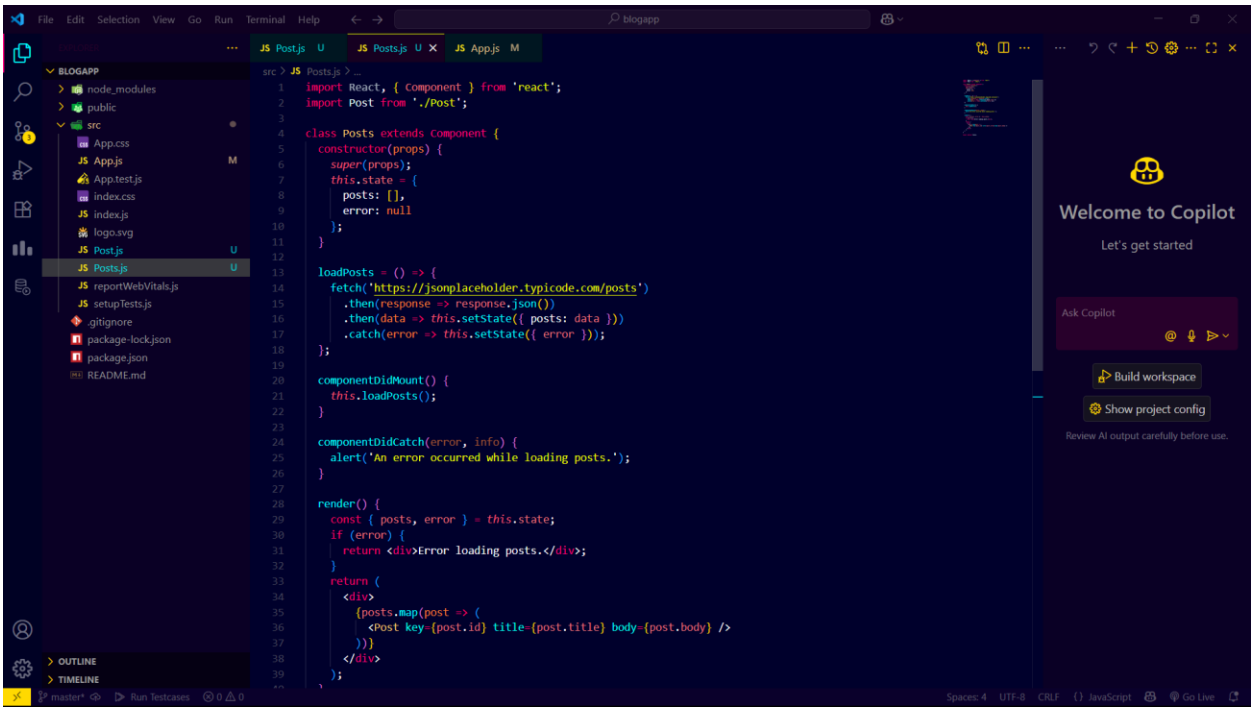
Figure 1: Post class

```
JS Post.js U ×   JS Posts.js U   JS App.js M
src > JS Post.js > ...
1   import React from 'react';
2
3   function Post({ title, body }) {
4     return (
5       <div>
6         <h2>{title}</h2>
7         <p>{body}</p>
8       </div>
9     );
10  }
11
12  export default Post;
13
```

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

```
JS 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)

```
Posts.js U
class Posts extends React.Component {
    constructor(props){
        super(props);
        //code
    }
    loadPosts() {
        //code
    }
}
```

Figure 3: loadPosts() method

```javascript
loadPosts = () => {
  fetch('https://jsonplaceholder.typicode.com/posts')
    .then(response => response.json())
    .then(data => this.setState({ posts: data }))
    .catch(error => this.setState({ error }));
};

componentDidMount() {
  this.loadPosts();
}
```

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

```javascript
JS Posts.js U ×
class Posts extends React.Component {
    constructor(props){
        super(props);
        //code
    }
    loadPosts() {
        //code
    }
    componentDidMount() {
        //code
    }
}
```

Figure 4: componentDidMount() hook

```javascript
componentDidMount() {
  this.loadPosts();
}
```

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

```
  render() {
    const { posts, error } = this.state;
    if (error) {
      return <div>Error loading posts.</div>;
    }
    return (
      <div>
        {posts.map(post => (
          <Post key={post.id} title={post.title} body={post.body} />
        ))}
      </div>
    );
  }
}
```

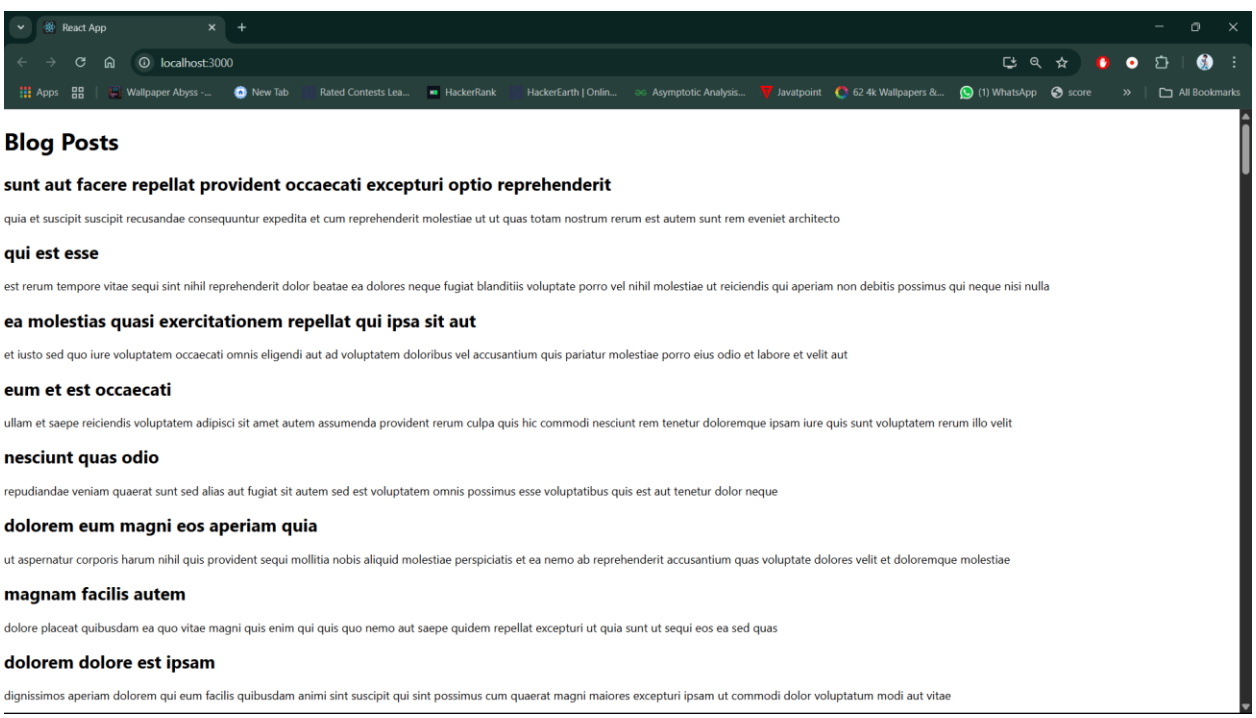
9. Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.

```
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

```
componentDidCatch(error, info) {
  alert('An error occurred while loading posts.');
}
```

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



Submitted By:

Name : Lingaraj Nayak

Superset ID : 6387607