

What is React?

React is a popular open-source JavaScript library for building user interfaces (UI).

It was created and is maintained by Facebook (now Meta).

React allows developers to create reusable UI components and efficiently update and render them when data changes.

Here are some key features and concepts of React:

1. **Component-Based Architecture:** React follows a component-based architecture, where the UI is divided into reusable and encapsulated components. These components manage their own state and can be composed together to build complex UIs.
2. **Virtual DOM:** React uses a virtual DOM (Document Object Model) to efficiently update the browser's DOM. The virtual DOM is a lightweight in-memory representation of the actual DOM. When state changes, react calculates the minimal DOM operations required to update the UI, reducing the performance cost of frequent DOM manipulations.
3. **JSX:** React uses JSX (JavaScript XML) syntax, which allows developers to write HTML-like code in JavaScript. JSX makes it easier to visualize and reason about the structure of the UI components.
4. **One-Way Data Binding:** React follows a one-way data binding approach, where state flows in a single direction, from parent components to child components. This makes the application more predictable and easier to reason about.
5. **React Hooks:** Introduced in React 16.8, Hooks allow developers to use state and other React features in functional components, without the need for class components. Hooks provide a more straightforward and lightweight way to manage component logic and state.
6. **Rich Ecosystem:** React has a vast ecosystem of third-party libraries and tools, such as React Router for handling routing, Redux for state management, and various UI component libraries like Material-UI and Ant Design.

React is widely used for building single-page applications (SPAs), mobile applications (with React Native), and even server-side rendered applications (with libraries like Next.js). Its component-based architecture, efficient updates, and comprehensive ecosystem make it a popular choice for building modern, interactive web applications.

Folder Structure of React Application

In a typical React application, the folder structure is designed to promote organization, modularity, and scalability. While there's no strict rule on how to structure a React project, a common and recommended structure looks something like this:

Here's what each folder typically contains:

node_modules/: This directory contains all the installed npm packages and their dependencies.

public/: This directory contains the main HTML file (index.html) and other static assets like images, fonts, etc.

src/:

components/: This directory contains reusable UI components, organized into subdirectories based on their functionality or feature.

pages/: This directory contains the main page components that represent different routes or views in the application.

styles/: This directory contains global styles or styles shared across multiple components.

utils/: This directory contains utility functions or helper modules used across the application.

assets/: This directory contains static assets like images, icons, or other media files.

services/: This directory contains modules for interacting with external services or APIs.

App.js: This is the root component that serves as the entry point for the React application.

index.js: This file is the entry point for the bundler (e.g., Webpack) and is responsible for rendering the root React component.

.gitignore: This file specifies which files or directories should be ignored by the Git version control system.

package.json: This file contains metadata about the project, including dependencies and scripts.

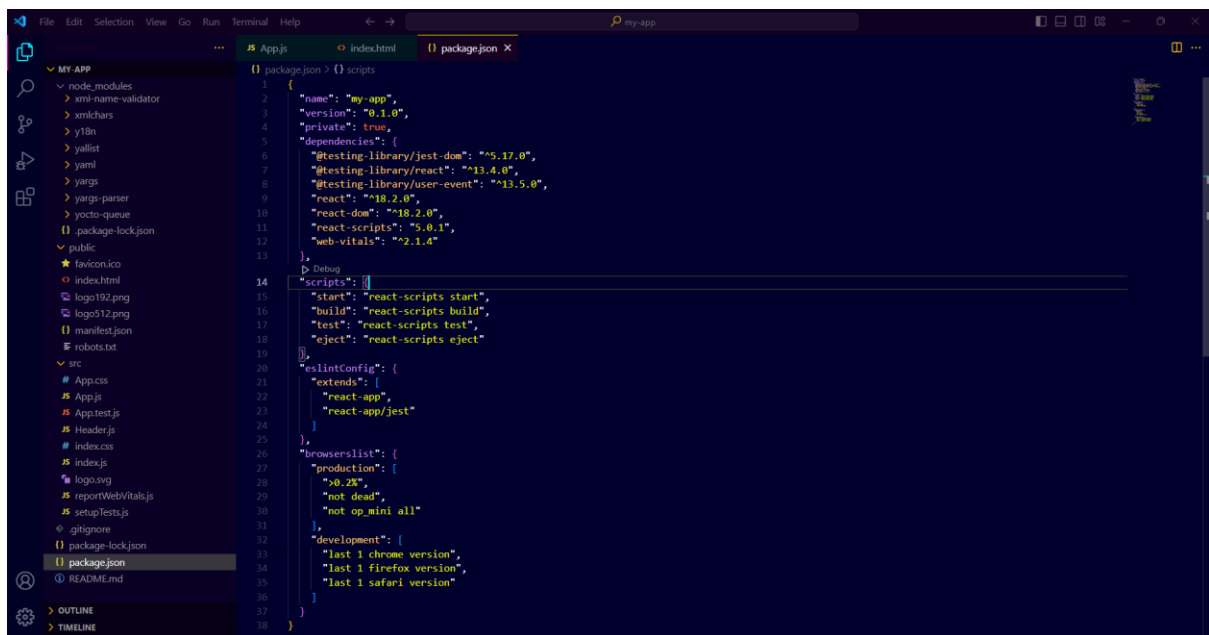
package-lock.json: This file is automatically generated and contains the exact versions of installed dependencies to ensure consistent builds.

README.md: This file provides documentation about the project, its setup, and usage instructions.

This structure is just a common convention and can be adjusted based on the needs and preferences of your project. The key idea is to separate concerns, promote reusability, and maintain a logical organization of files and directories.

1.Package.json-

Store required scripts and dependencies (versions and other information).



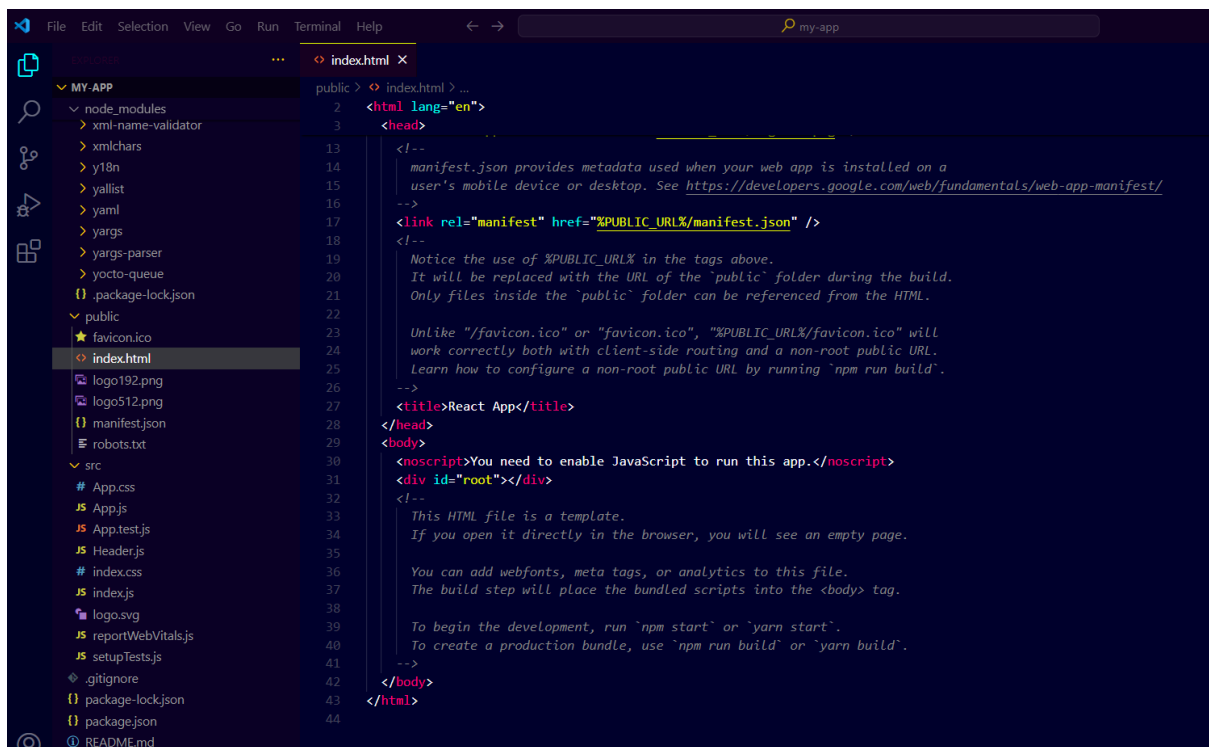
The screenshot shows the VS Code editor with the 'package.json' file open. The file contains the following content:

```
1 {
2   "name": "my-app",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@testing-library/jest-dom": "^5.17.0",
7     "@testing-library/react": "^13.4.0",
8     "@testing-library/user-event": "^13.5.0",
9     "react": "^18.2.0",
10    "react-dom": "^18.2.0",
11    "react-scripts": "5.0.1",
12    "web-vitals": "^2.1.4"
13  },
14  "scripts": {
15    "start": "react-scripts start",
16    "build": "react-scripts build",
17    "test": "react-scripts test",
18    "eject": "react-scripts eject"
19  },
20  "eslintConfig": {
21    "extends": [
22      "react-app",
23      "react-app/jest"
24    ]
25  },
26  "browserslist": {
27    "production": [
28      ">0.2%",
29      "not dead",
30      "not op_mini all"
31    ],
32    "development": [
33      "last 1 chrome version",
34      "last 1 firefox version",
35      "last 1 safari version"
36    ]
37  }
38 }
```

2.Index.html-Only one html file is there,

used for single page application,using this file we can do add,update,delete but we don't add any extra code here.

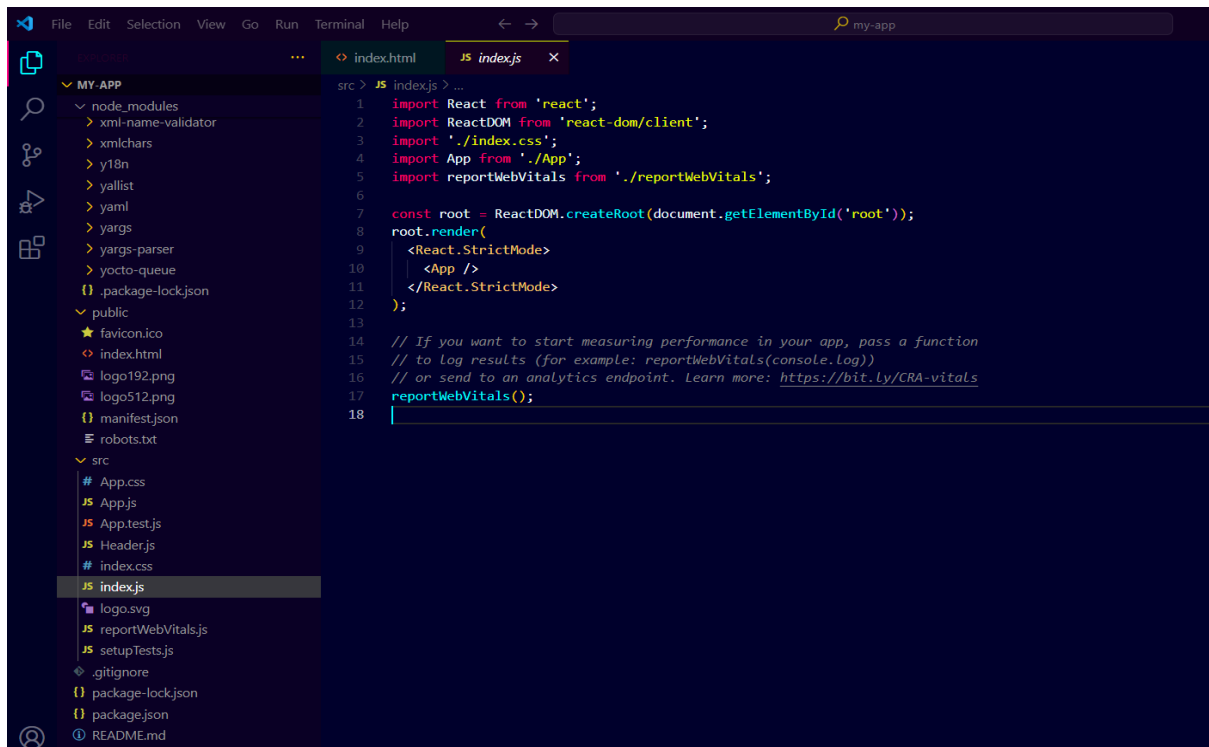
Defined here- `<div id="root"></div>`



The screenshot shows the VS Code editor with the 'index.html' file open. The file contains the following content:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4
5   <!--
6     manifest.json provides metadata used when your web app is installed on a
7     user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
8   -->
9   <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
10
11   <!--
12     Notice the use of %PUBLIC_URL% in the tags above.
13     It will be replaced with the URL of the 'public' folder during the build.
14     Only files inside the 'public' folder can be referenced from the HTML.
15
16     Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
17     work correctly both with client-side routing and a non-root public URL.
18     Learn how to configure a non-root public URL by running `npm run build`.
19   -->
20   <title>React App</title>
21 </head>
22 <body>
23
24   <noscript>You need to enable JavaScript to run this app.</noscript>
25
26   <div id="root"></div>
27
28   <!--
29     This HTML file is a template.
30     If you open it directly in the browser, you will see an empty page.
31
32     You can add webfonts, meta tags, or analytics to this file.
33     The build step will place the bundled scripts into the <body> tag.
34
35     To begin the development, run `npm start` or `yarn start`.
36     To create a production bundle, use `npm run build` or `yarn build`.
37   -->
38 </body>
39 </html>
```

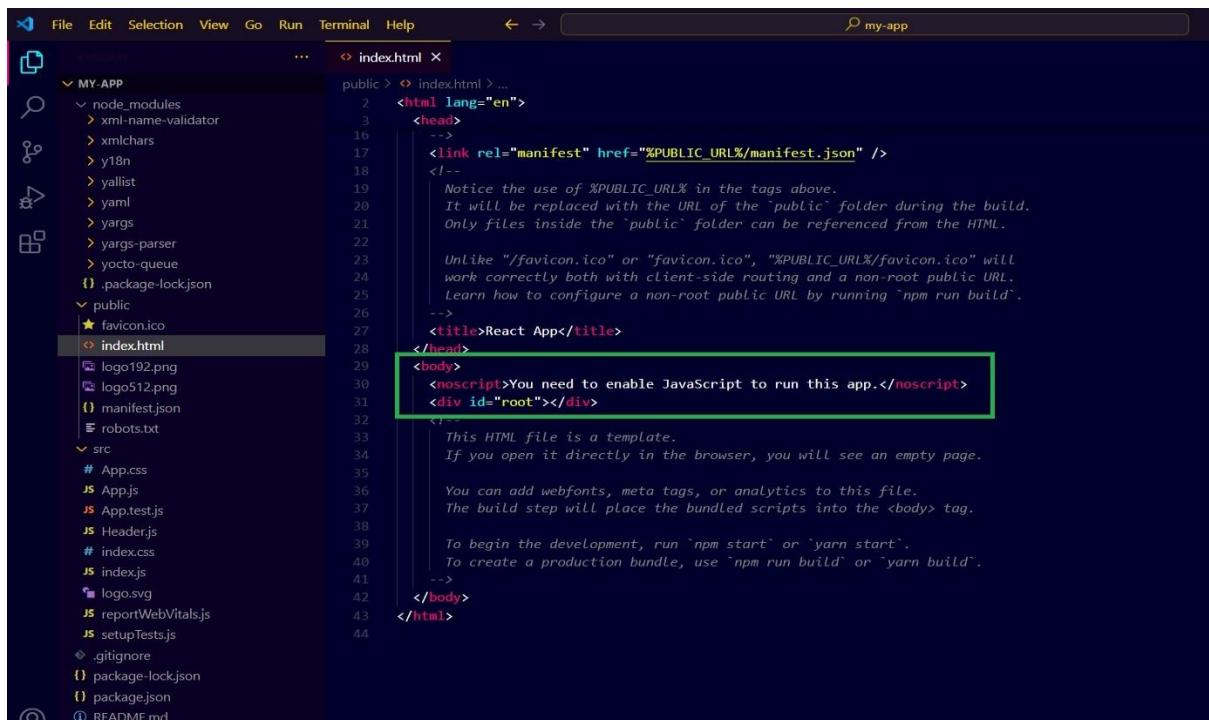
3.Index.js- starting point of react and App component code render in html file which Id is “root”.



The screenshot shows the VS Code editor with the 'index.js' file open. The file contains the following code:

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 const root = ReactDOM.createRoot(document.getElementById('root'));
8 root.render(
9   <React.StrictMode>
10     <App />
11   </React.StrictMode>
12 );
13
14 // If you want to start measuring performance in your app, pass a function
15 // to log results (for example: reportWebVitals(console.log))
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17 reportWebVitals();
18
```

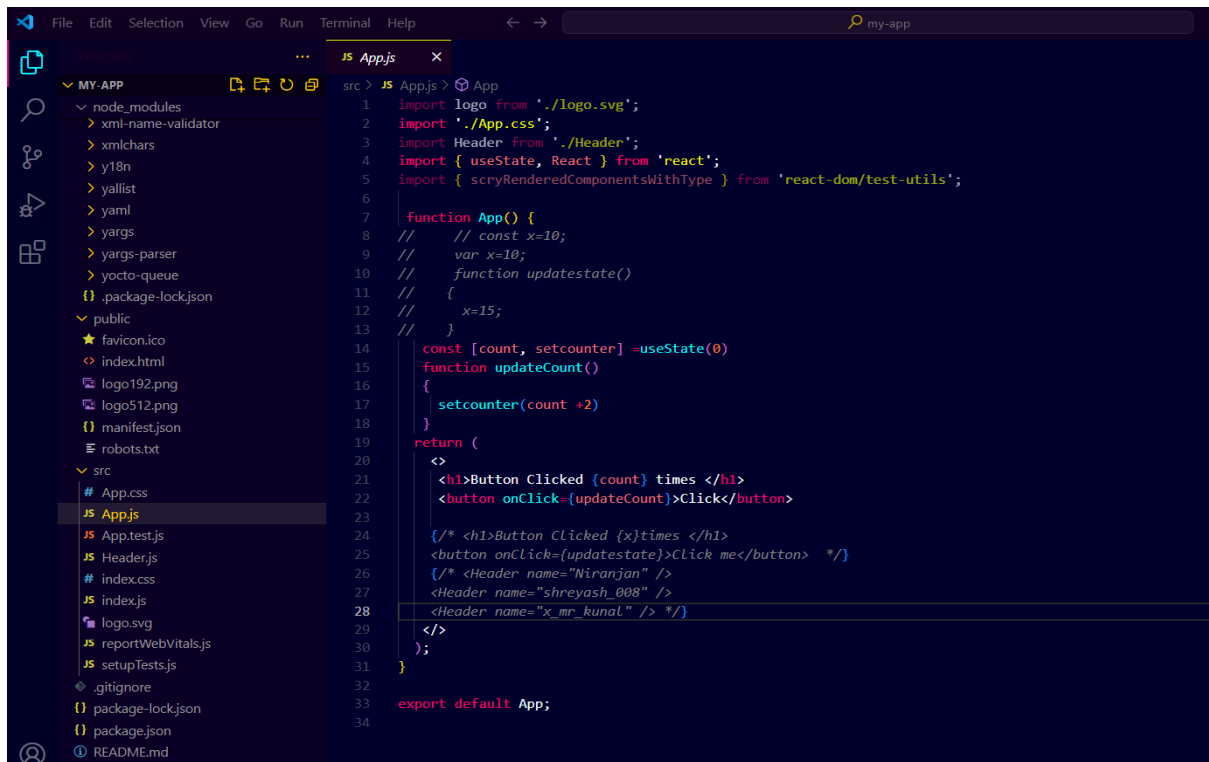
3.1 We already check in **index.html** there is **<div id=“ root”></div>**



The screenshot shows the VS Code editor with the 'index.html' file open. The file contains the following code:

```
2 <html lang="en">
3   <head>
4     <!--
5       <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
6     -->
7     <!--
8       Notice the use of %PUBLIC_URL% in the tags above.
9       It will be replaced with the URL of the 'public' folder during the build.
10       Only files inside the 'public' folder can be referenced from the HTML.
11
12       Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
13       work correctly both with client-side routing and a non-root public URL.
14       Learn how to configure a non-root public URL by running `npm run build`.
15     -->
16     <title>React App</title>
17   </head>
18   <body>
19     <noscript>You need to enable JavaScript to run this app.</noscript>
20     <div id="root"></div>
21   </body>
22 </html>
```

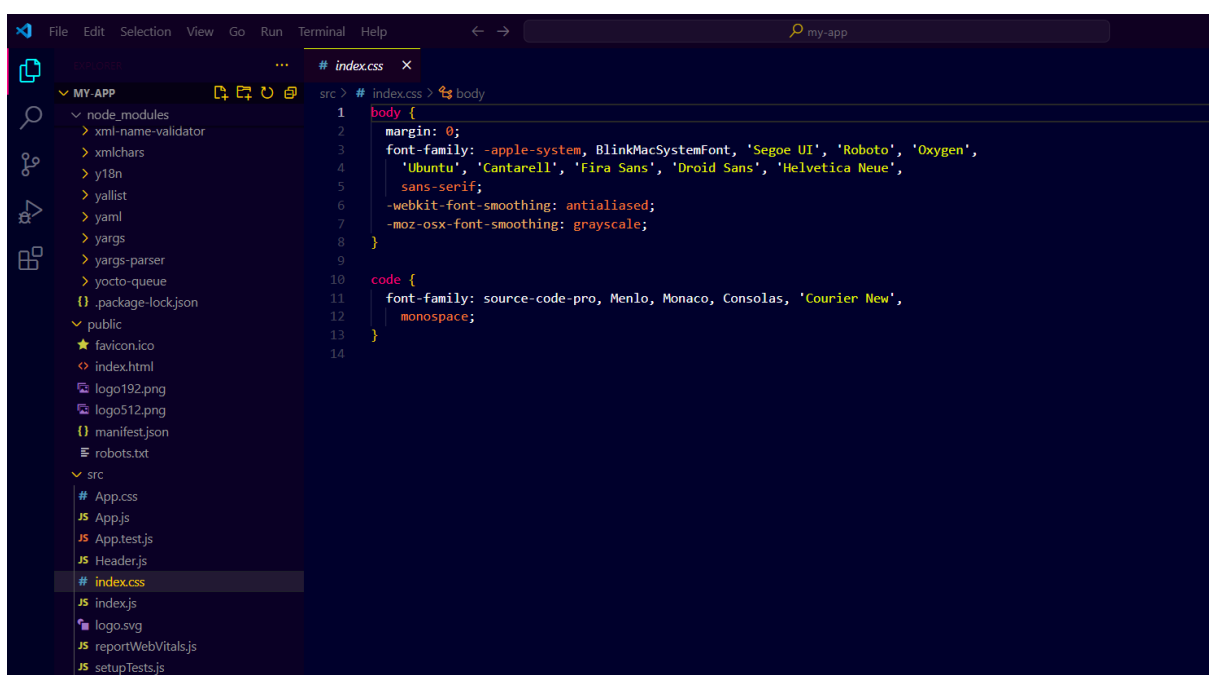
4.App.js- (Actual code is written here) All code executes here, this code render in index.js file which id is “root”



The screenshot shows the VS Code editor with the 'my-app' project open. The file explorer on the left shows the project structure, including 'node_modules', 'public', and 'src'. The 'App.js' file is selected in the 'src' folder. The code in 'App.js' is as follows:

```
src > JS App.js > App
1  import logo from './logo.svg';
2  import './App.css';
3  import Header from './Header';
4  import { useState, React } from 'react';
5  import { scryRenderedComponentsWithType } from 'react-dom/test-utils';
6
7  function App() {
8    // // const x=10;
9    // var x=10;
10   // function updatestate()
11   // {
12   //   x=15;
13   // }
14   const [count, setcounter] =useState(0)
15   function updateCount()
16   {
17     setcounter(count +2)
18   }
19   return (
20     <>
21     <h1>Button Clicked {count} times </h1>
22     <button onClick={updateCount}>Click</button>
23
24     /* <h1>Button Clicked {x}times </h1>
25     <button onClick={updatestate}>Click me</button> */
26     /* <Header name="Niranjan" />
27     <Header name="shreyash_008" />
28     <Header name="x_mr_kunal" /> */
29   </>
30   );
31 }
32
33 export default App;
34
```

5. Index.css- This file used for Styling



The screenshot shows the VS Code editor with the 'my-app' project open. The file explorer on the left shows the project structure, including 'node_modules', 'public', and 'src'. The 'index.css' file is selected in the 'src' folder. The code in 'index.css' is as follows:

```
src > # index.css > body
1  body {
2    margin: 0;
3    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
4    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
5    sans-serif;
6    -webkit-font-smoothing: antialiased;
7    -moz-osx-font-smoothing: grayscale;
8  }
9
10 code {
11   font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
12   monospace;
13 }
14
```

7. App.test.js- used for writing Testcases

