**React – JSON-server and Firebase Real Time Database**

Q1: What do you mean by RESTful web services?

Ans: - RESTful web services are web APIs that follow the principles of REST (Representational State Transfer), an architectural style that uses standard HTTP methods like GET, POST, PUT, and DELETE to perform CRUD operations on resources, which are typically represented in JSON or XML format. These services are stateless, scalable, and commonly used to allow communication between client and server in modern web applications.

Q2: What is Json-Server? How we use in React ?

Ans: - JSON-Server is a simple, lightweight fake REST API server that lets developers mock backend data quickly using a db.json file. In React, it is commonly used for prototyping or testing by running json-server --watch db.json, which provides endpoints like a real backend, and React apps can then fetch or manipulate data via HTTP methods.

Q3: How do you fetch data from a Json-server API in React? Explain the role of fetch() or axios() in making API requests.

Ans: To fetch data from a JSON-server API in React, we typically use fetch() or axios() inside a useEffect() hook to make asynchronous API calls. fetch() is a built-in JavaScript method to make HTTP requests, while axios() is a third-party library that provides a simpler syntax and automatic JSON parsing. Both allow React apps to retrieve or modify data from the API and update the component state accordingly.

Q4: What is Firebase? What features does Firebase offer?

Ans: Firebase is a Backend-as-a-Service (BaaS) platform by Google that provides various tools for building and scaling web and mobile applications. Its key features include real-time NoSQL database (Firestore), authentication, cloud functions, hosting, storage, and analytics, making it easy for developers to implement backend functionalities without managing servers.

Q5: Discuss the importance of handling errors and loading states when working with APIs in React.

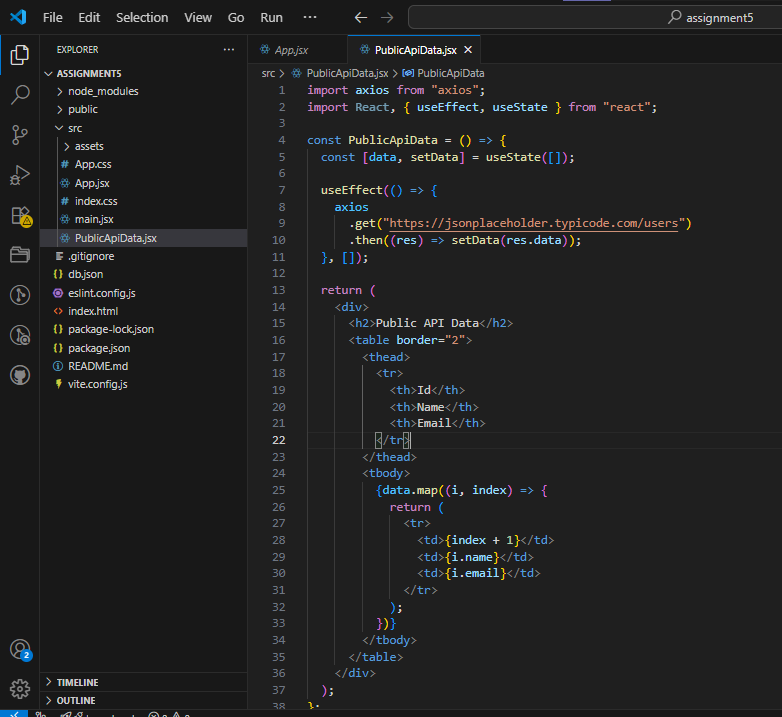
Ans: Handling errors and loading states is crucial when working with APIs in React to provide a good user experience and ensure app stability. Without proper loading indicators, users may be confused during network delays, and unhandled errors (like failed requests) can break the UI or result in incorrect data being shown. Using conditional rendering and try-catch or .catch() blocks helps manage these states effectively.

**Lab Tasks**

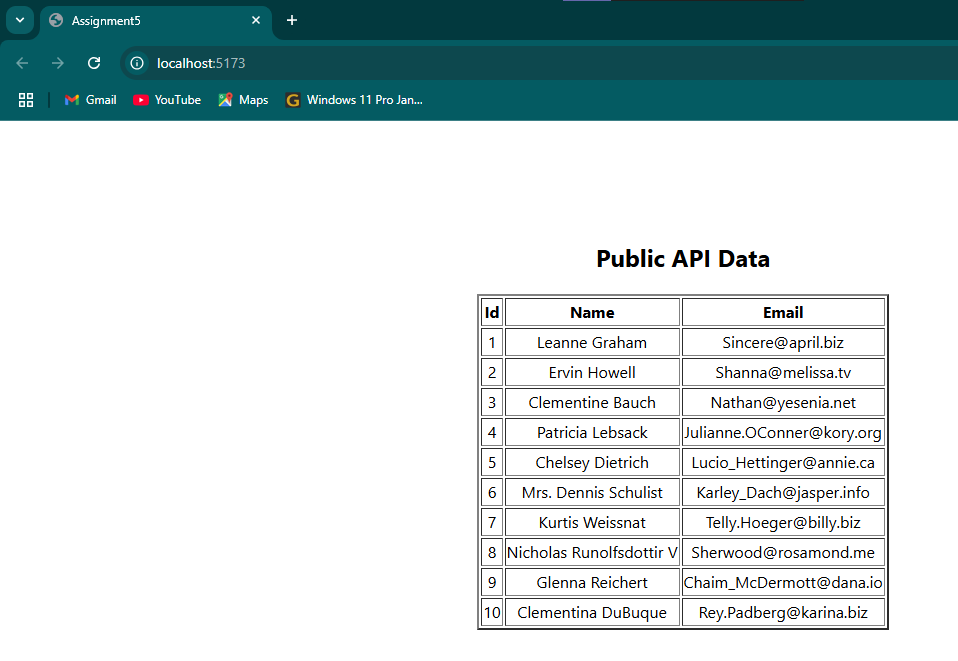
**Task 1 :**

* **Create a React component that fetches data from a public API (e.g., a list of users) and displays it in a table format.**

**Code:**

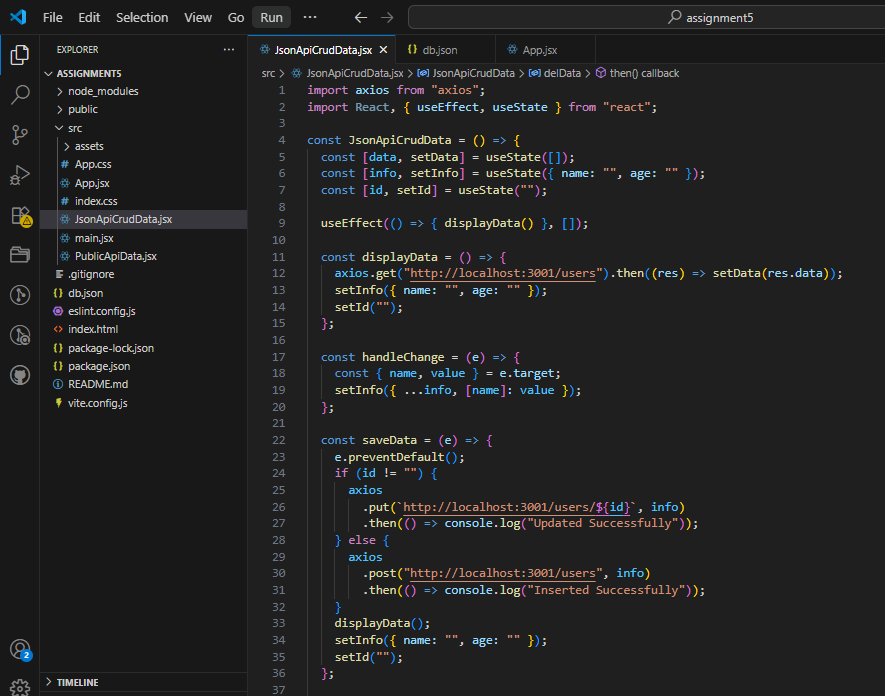
****

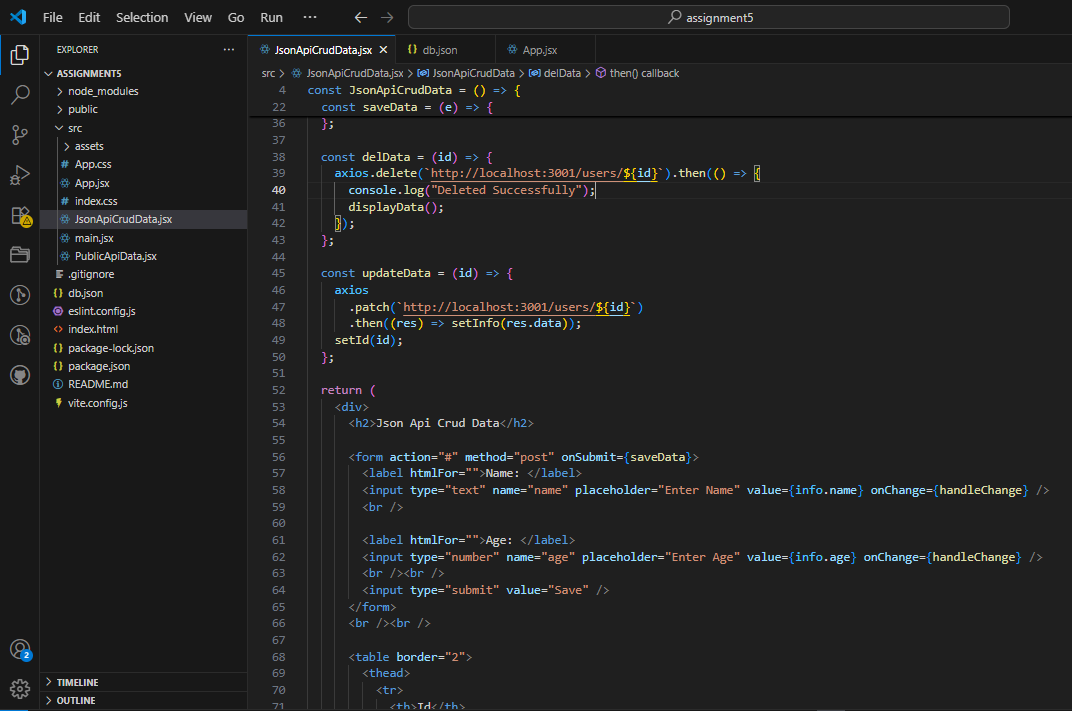
**Output:**

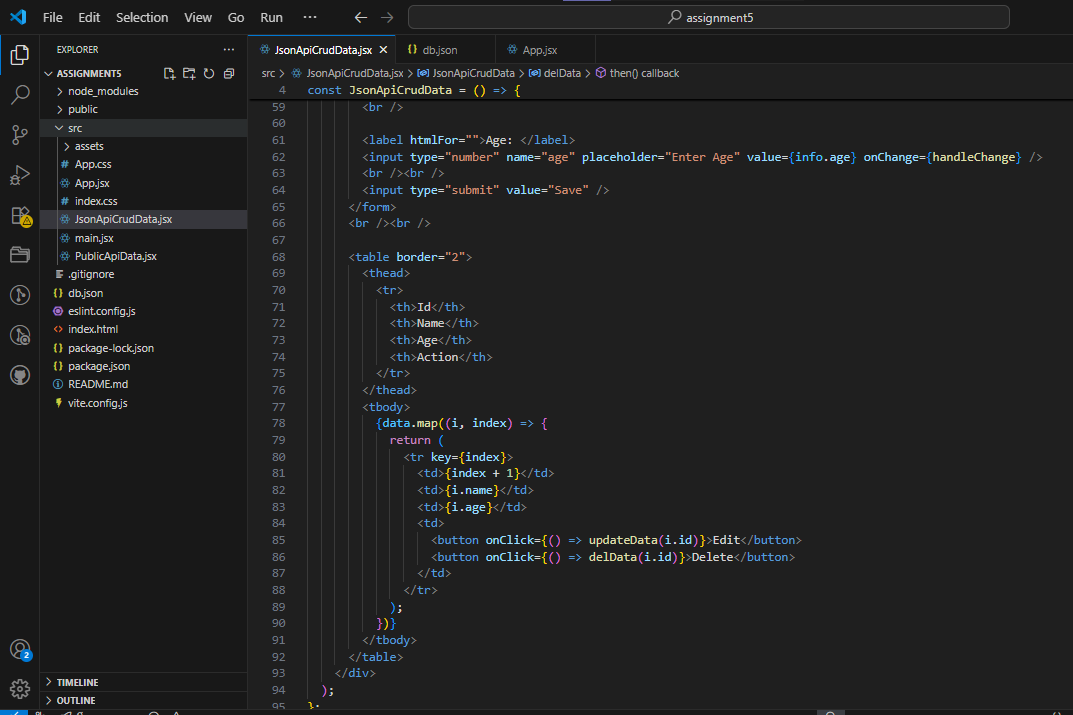
****

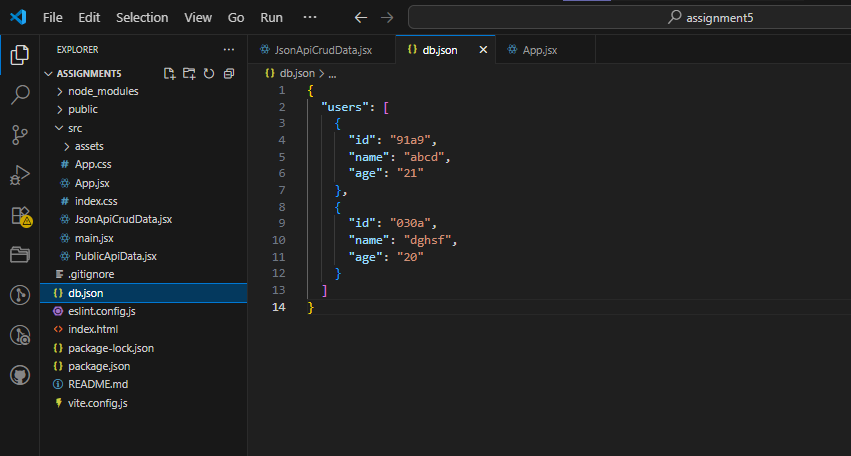
* **Create a React app with Json-server and use Get , Post , Put , Delete & patch method on Json-server API.**

**Code:**

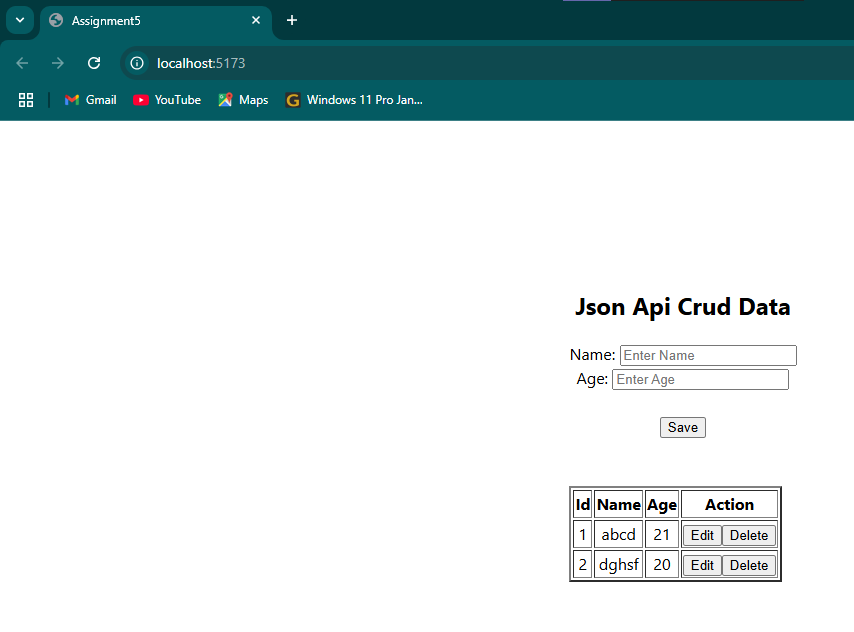
****

****

****

****

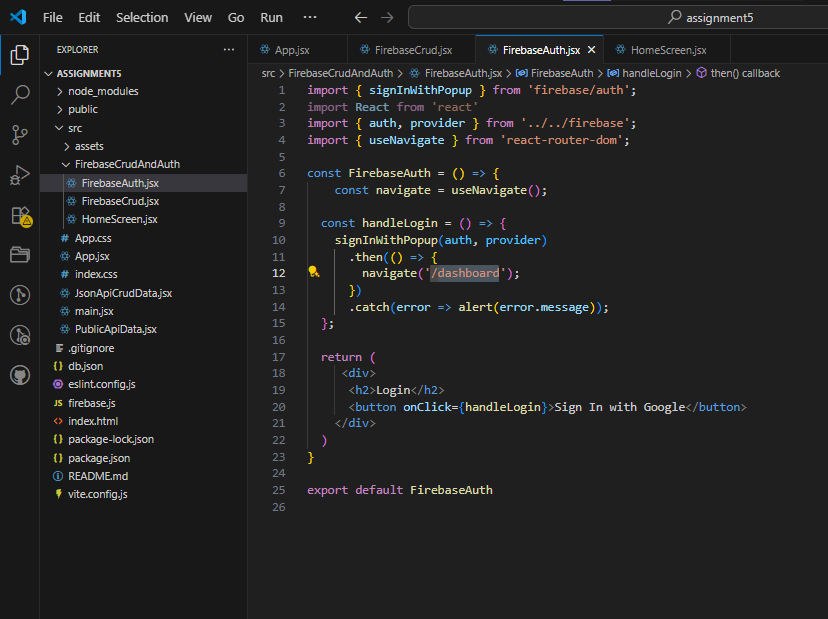
**Output:**

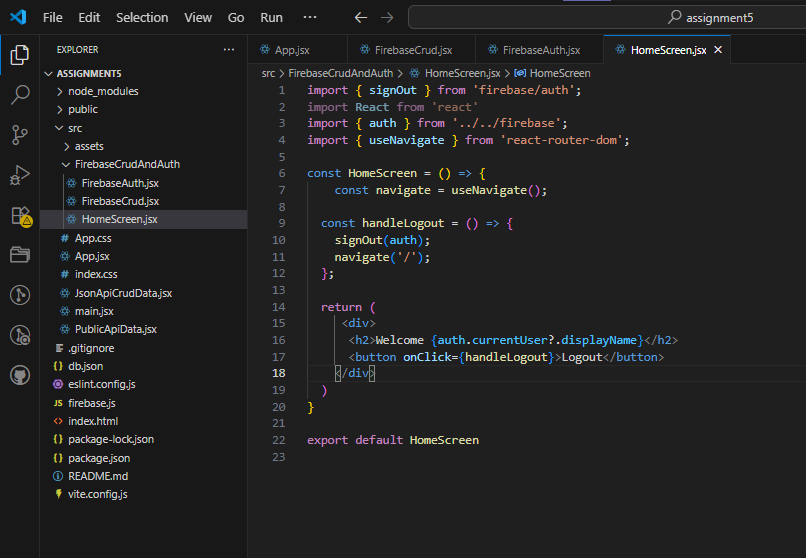
****

**Task 2 :**

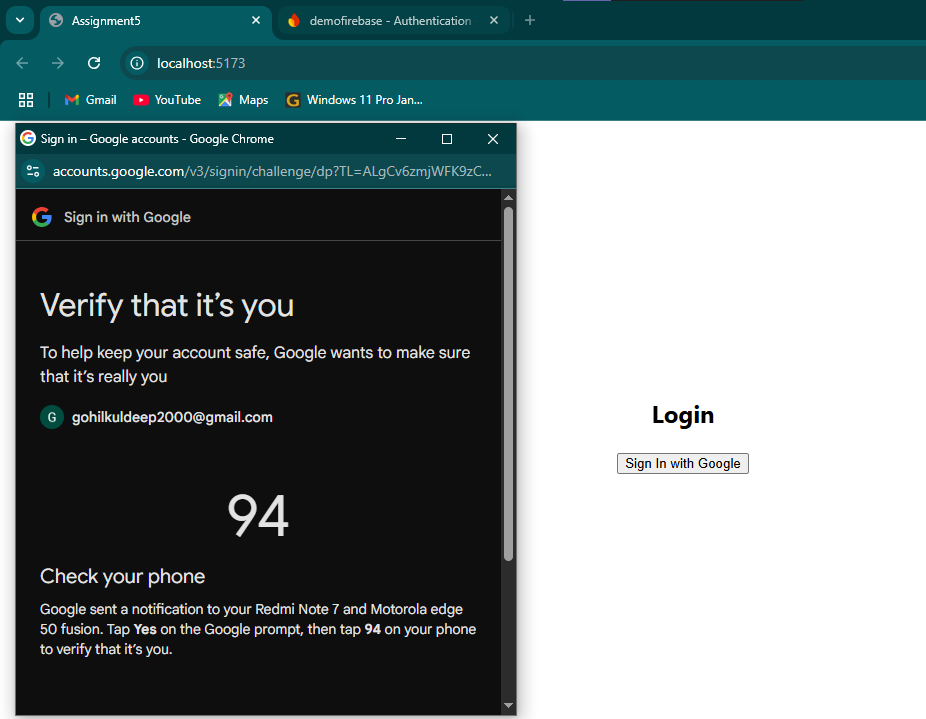
* **Create a React app crud and Authentication with firebase API.**
* **Implement google Authentication with firebase API.**

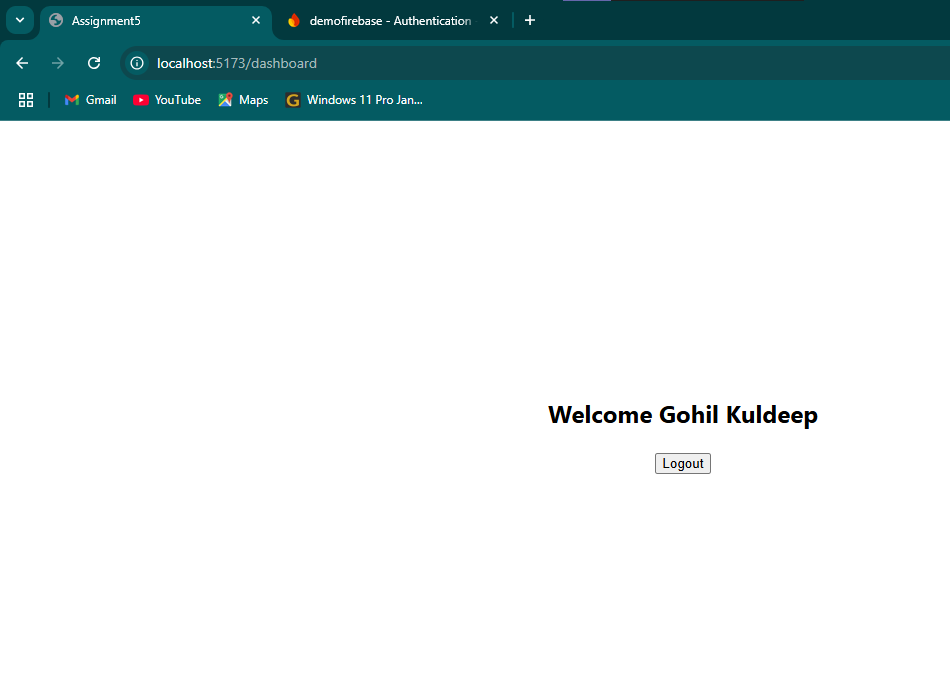
**Code:**

****

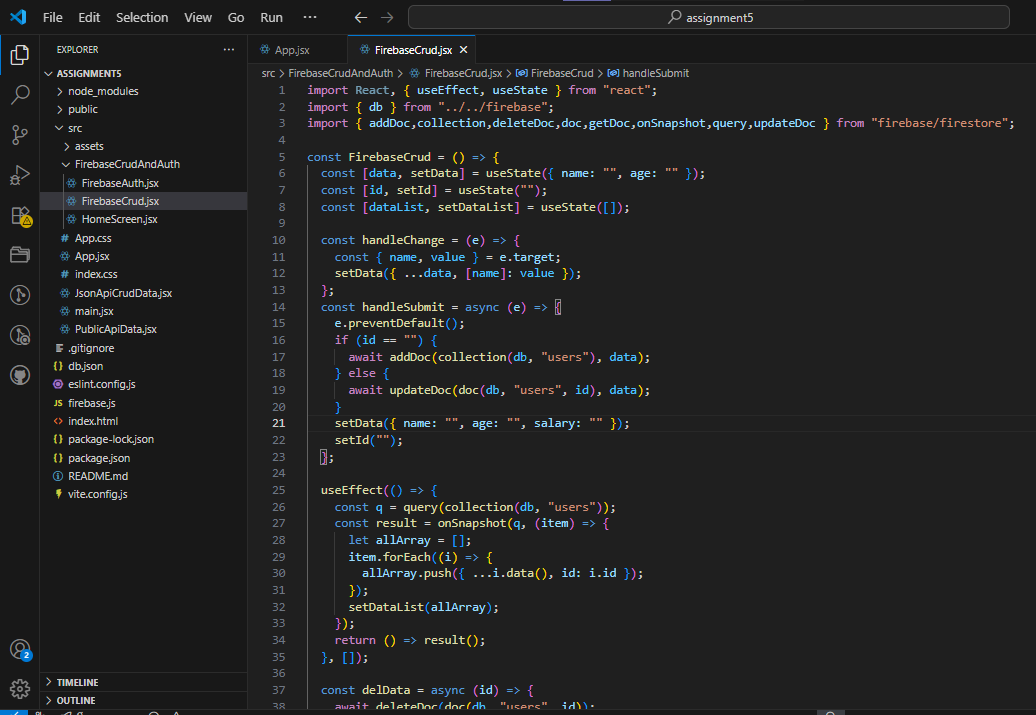
****

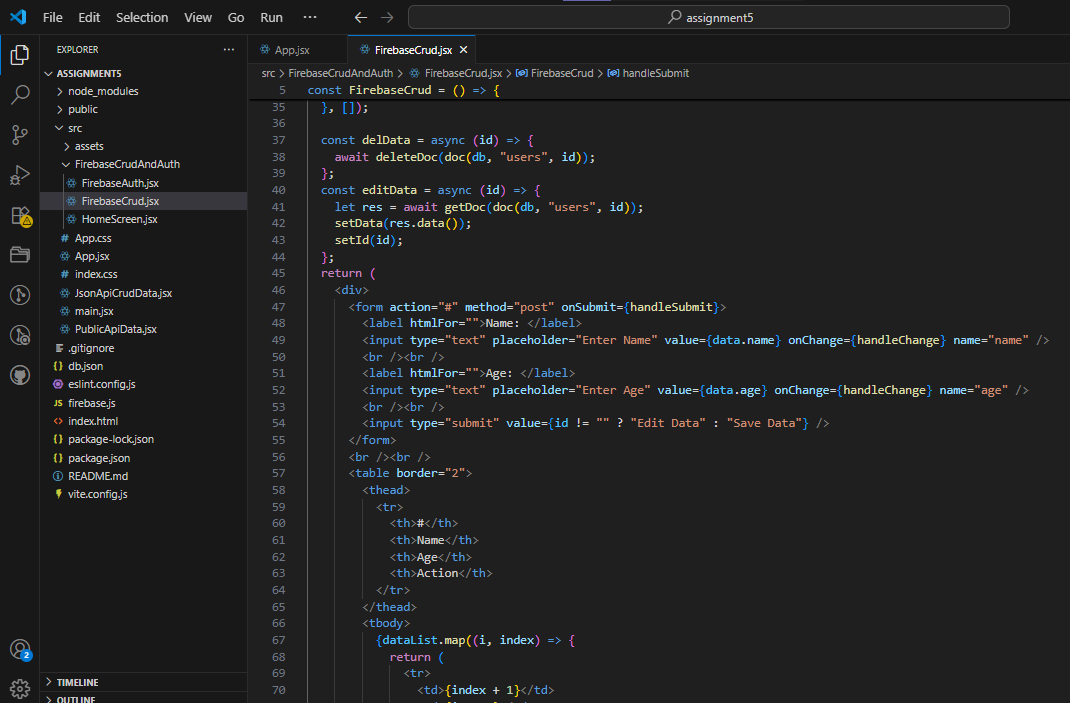
**Output:**

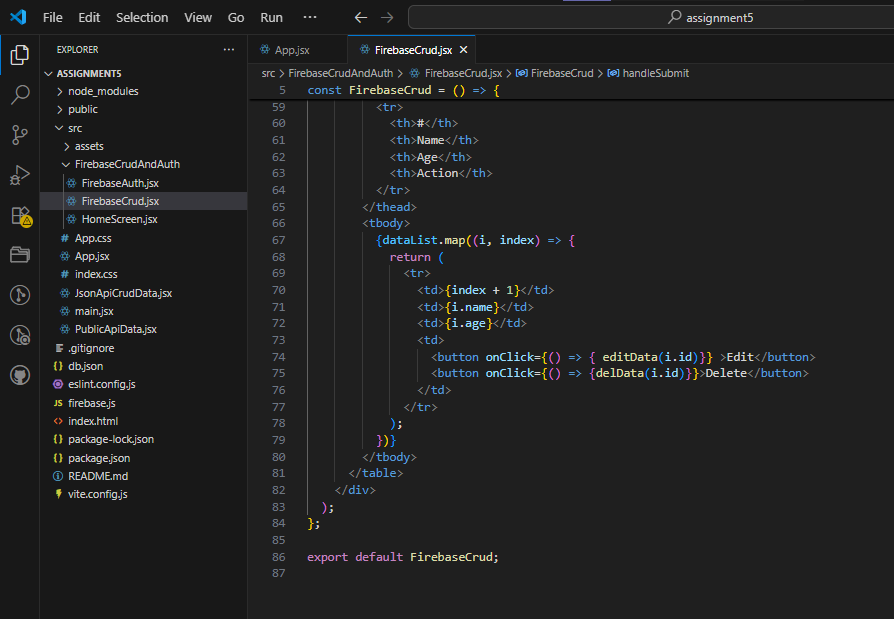
****

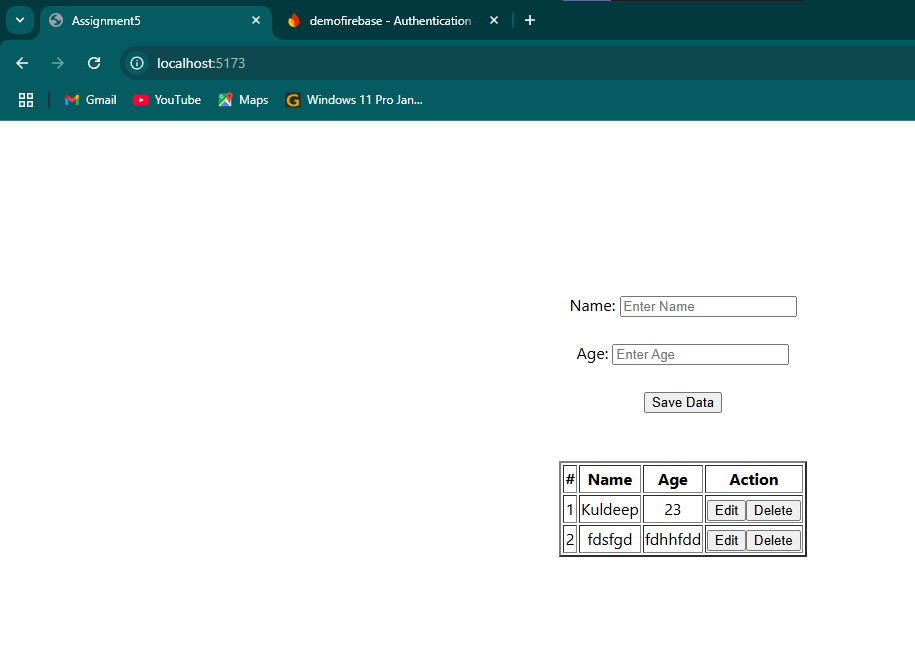
****

**Code:**

****

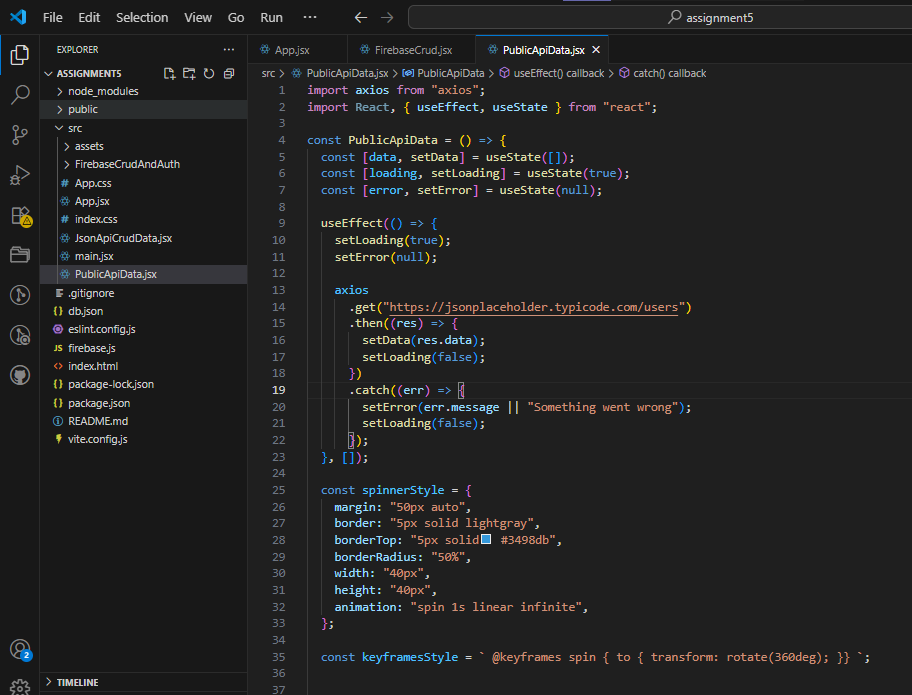
****

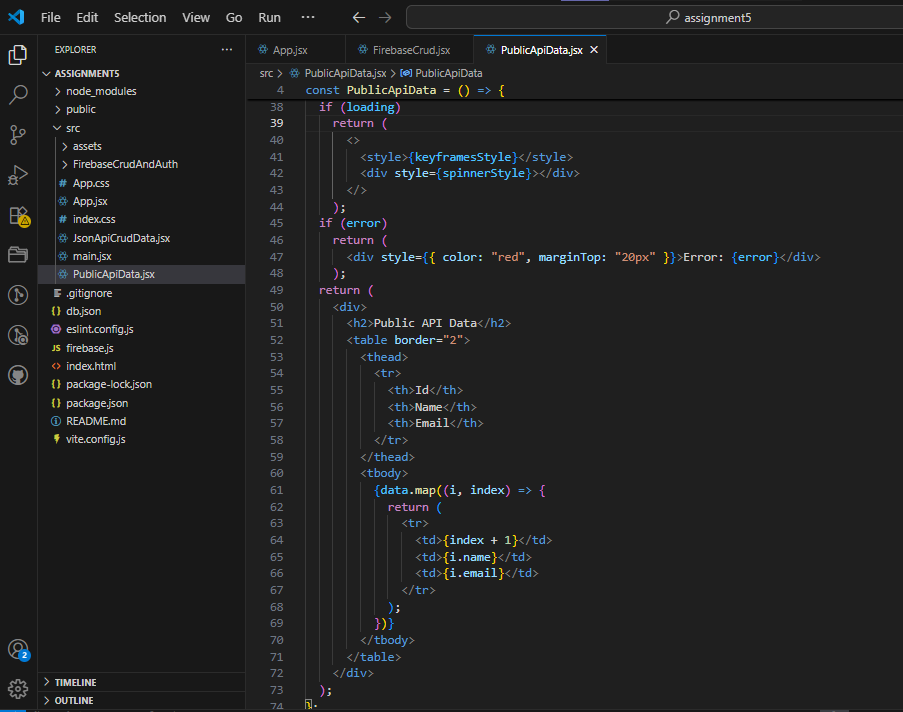
****

****

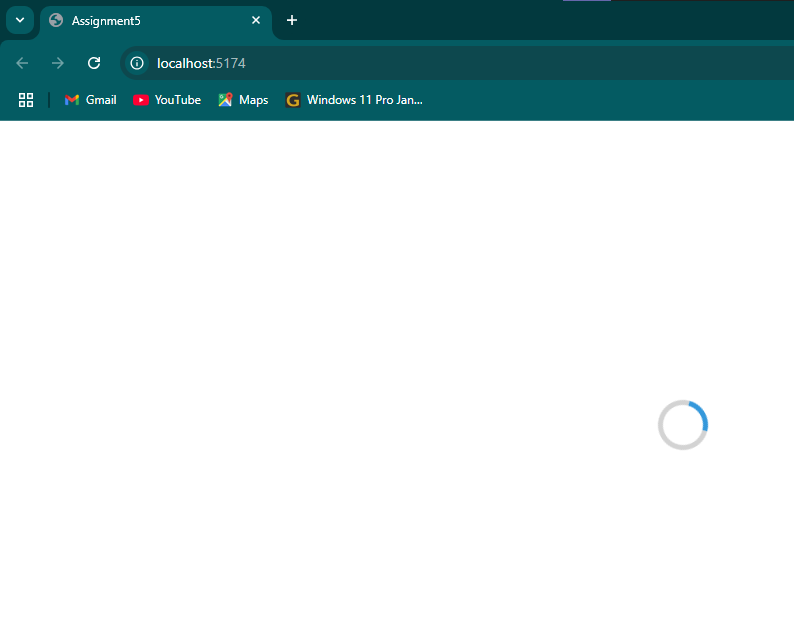
**Task 2 : Implement error handling and loading states for the API call. Display a loading spinner while the data is being fetched.**

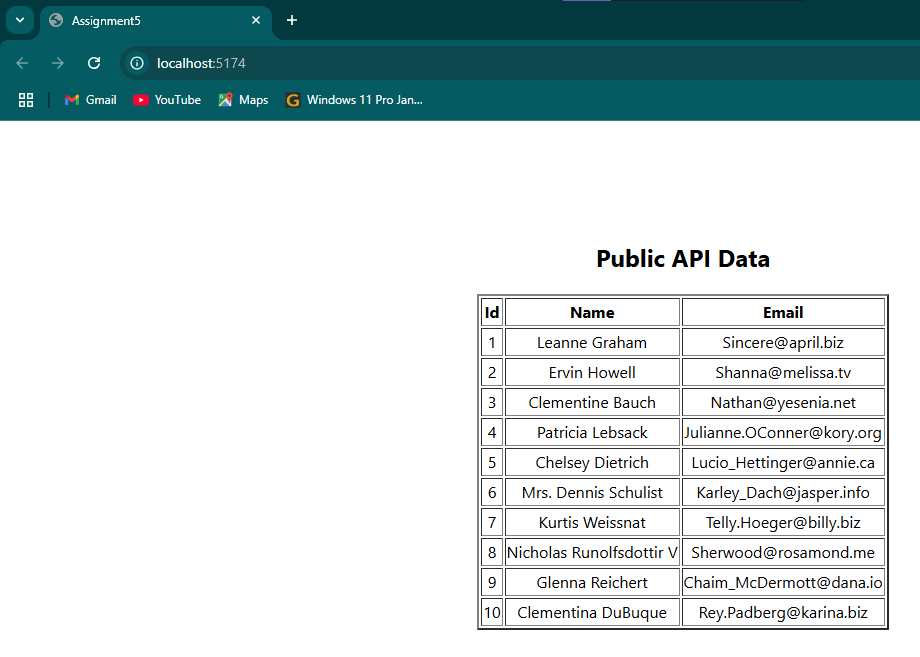
**Code:**

****

****

**Output:**

****

****

**Context API**

Q1: What is the Context API in React? How is it used to manage global state across multiple components?

Ans: - The Context API in React is a built-in tool that allows you to manage and share global state across multiple components without passing props manually at every level (avoiding prop drilling). It is useful when several components need access to the same data.

**its works:-**

1. **createContext():** This function creates a Context object. It defines the default value and returns a Provider and Consumer.
2. **<Context.Provider>:** Wraps the part of your app where you want the context to be available and provides the state.
3. **useContext():** A React Hook used inside components to access the current value of the context.

Q2: Explain how createContext() and useContext() are used in React for sharing state.

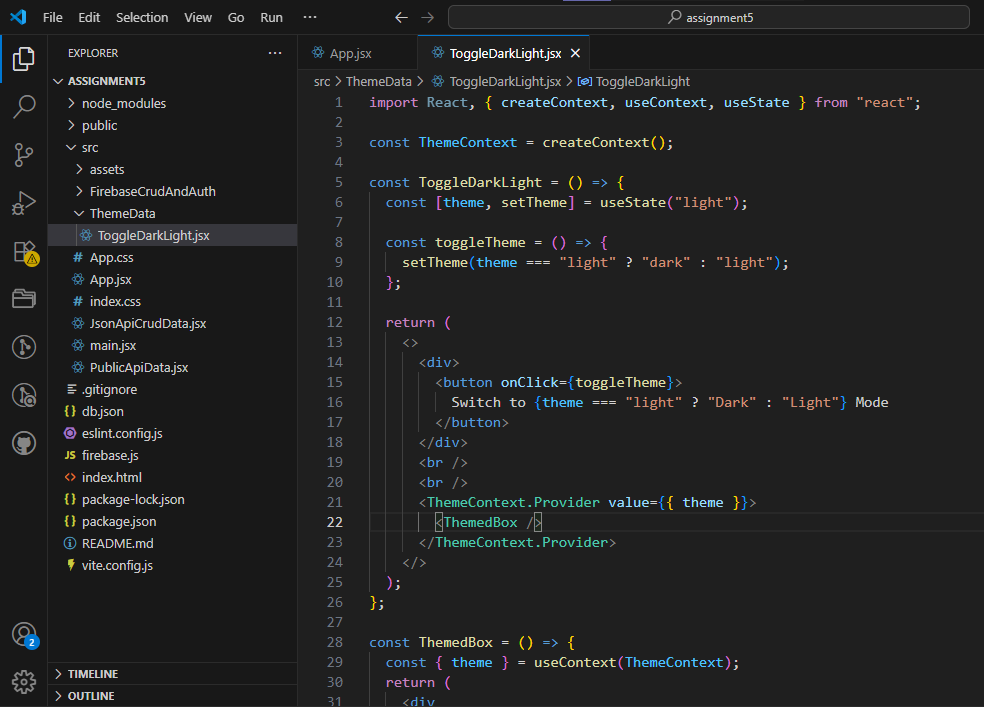
Ans: - createContext() and useContext() are used together to share state or data between components without passing props manually through each level of the component tree.

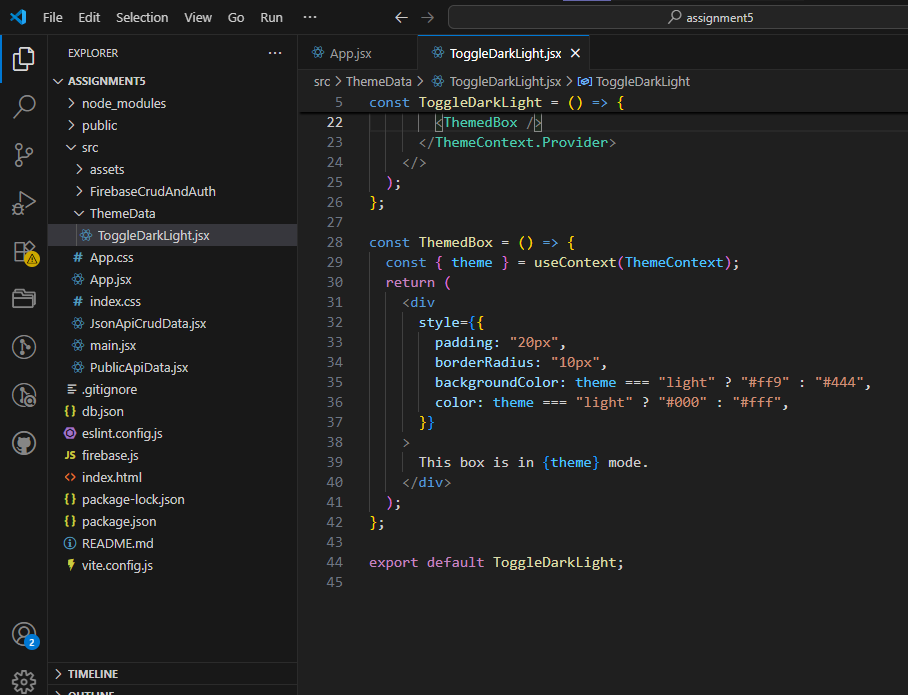
* **createContext()** :- It creates a Context object that holds the value you want to share. Returns a Provider and a Consumer.
* **useContext() :-** A React hook that lets you access the current value of the context inside any functional component. It must be used within the component tree wrapped by the corresponding Provider.

**Lab Tasks**

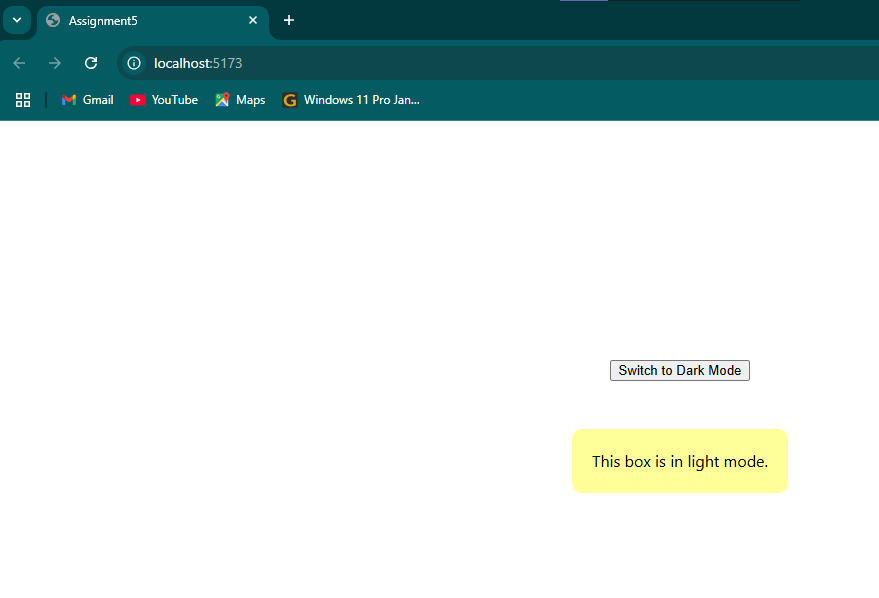
**Task 1 : Create a simple theme toggle (light/dark mode) using the Context API. The theme state should be shared across multiple components.**

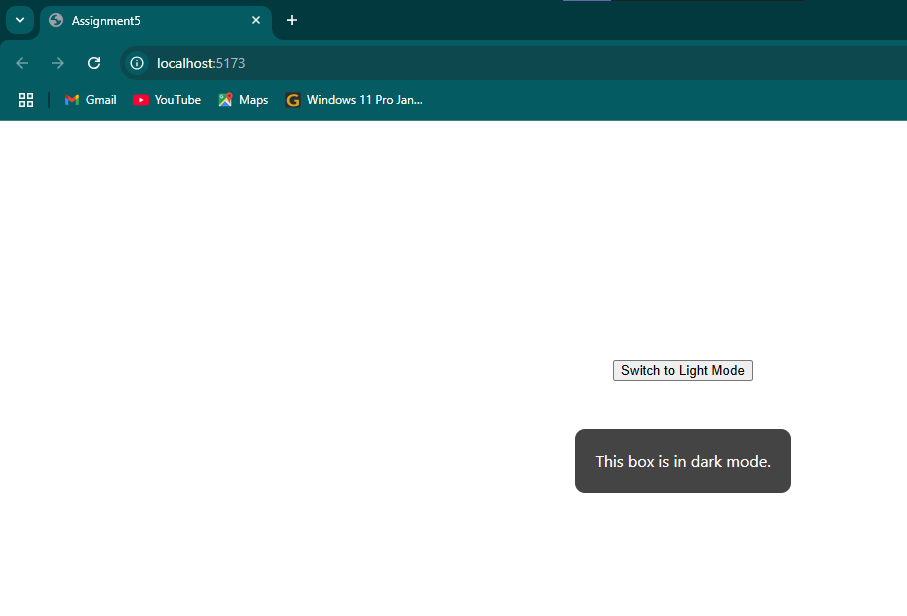
**Code:**





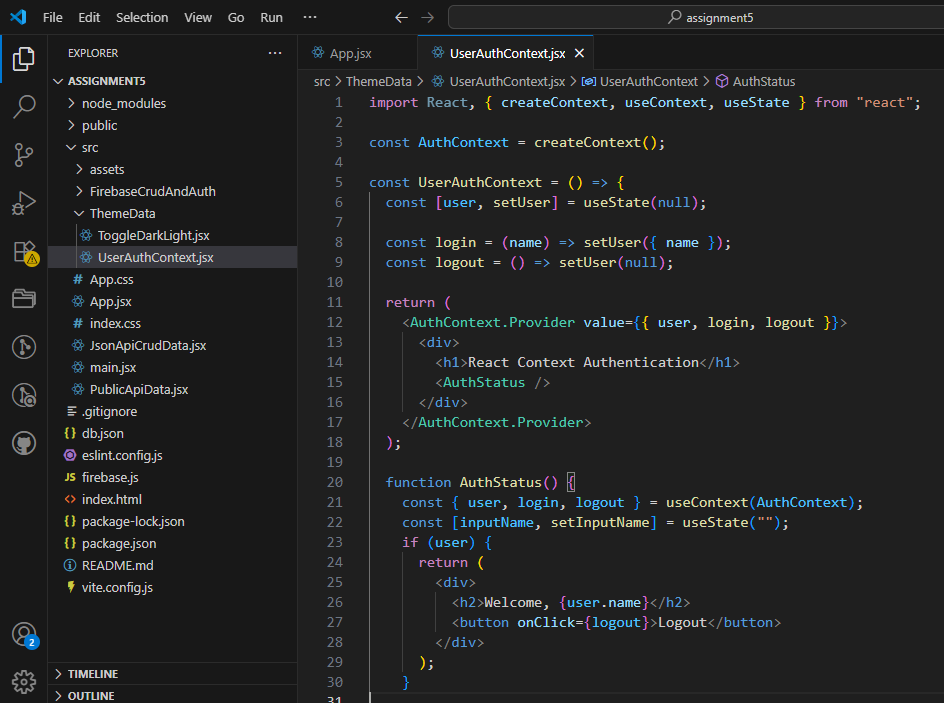
**Output:**

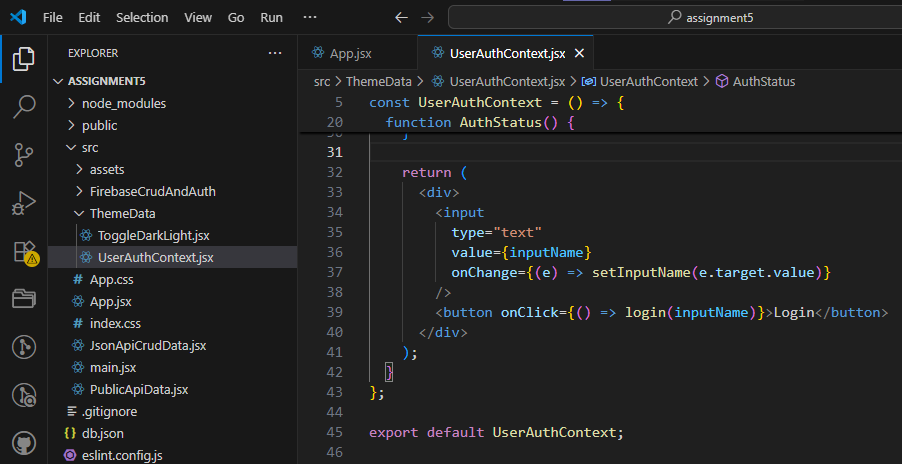
****



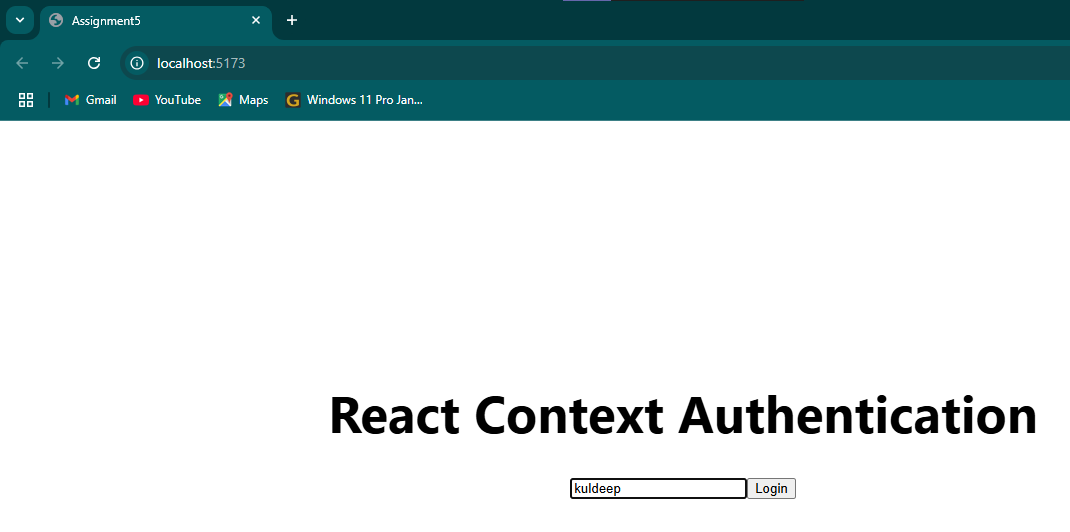
**Task 2 : Use the Context API to create a global user authentication system. If the user is logged in, display a welcome message; otherwise, prompt them to log in.**

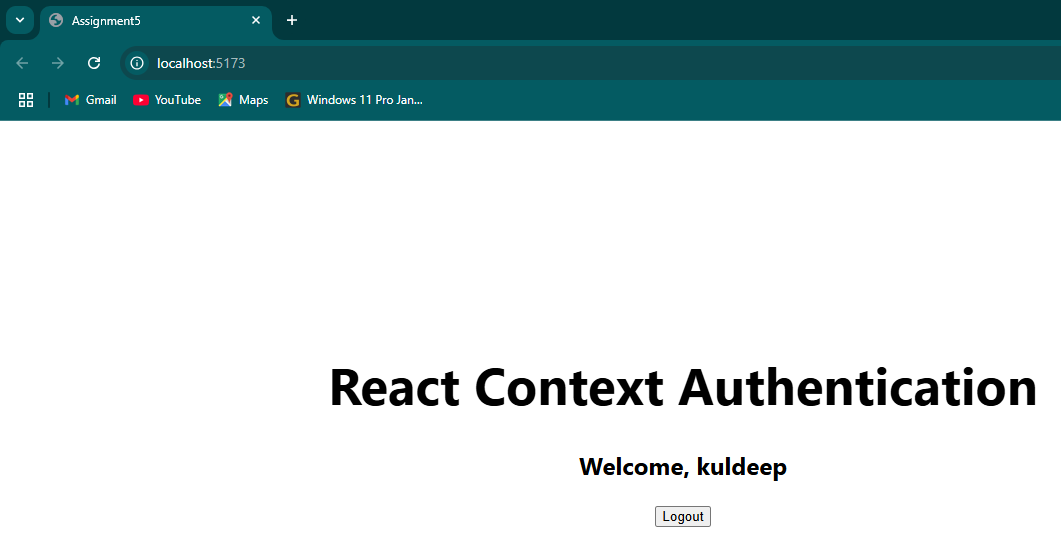
**Code:**





**Output:**

****



**State Management (Redux, Redux-Toolkit or Recoil)**

Q1: What is Redux, and why is it used in React applications? Explain the core concepts of actions, reducers, and the store.

Ans: - Redux is a predictable state container for JavaScript applications, most commonly used with React for managing application-level state. It helps you write applications that behave consistently, are easy to test, and can run in different environments. React has a built-in component state (useState, useReducer), but it becomes hard to manage when:

* Multiple components need access to the same state.
* You need a global state (e.g., user authentication, theme, cart data).
* You want better control over state changes and debugging.

**Core Concepts of Redux :-**

1. **Store :-** The single source of truth. Holds the entire state of the app. Created using createStore(reducer) (or configureStore in Redux Toolkit).
2. **Actions :-** Plain JavaScript objects that describe what happened. Must have a type field. Can include a payload with additional data.
3. **Reducers :-** Pure functions that receive actions and update state accordingly. Return the new state based on the action type.

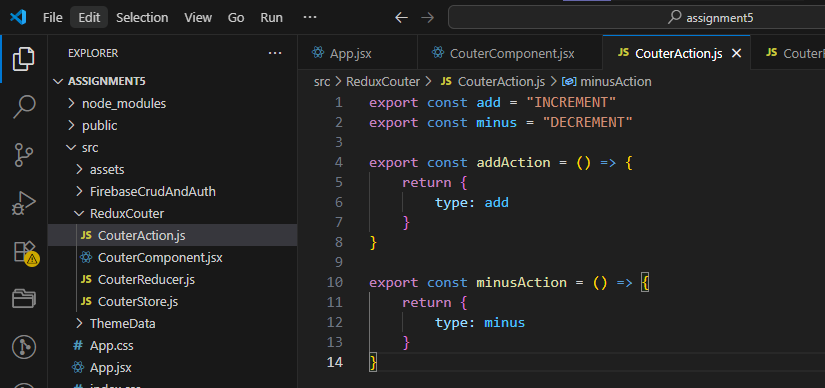
Q2: How does Recoil simplify state management in React compared to Redux?

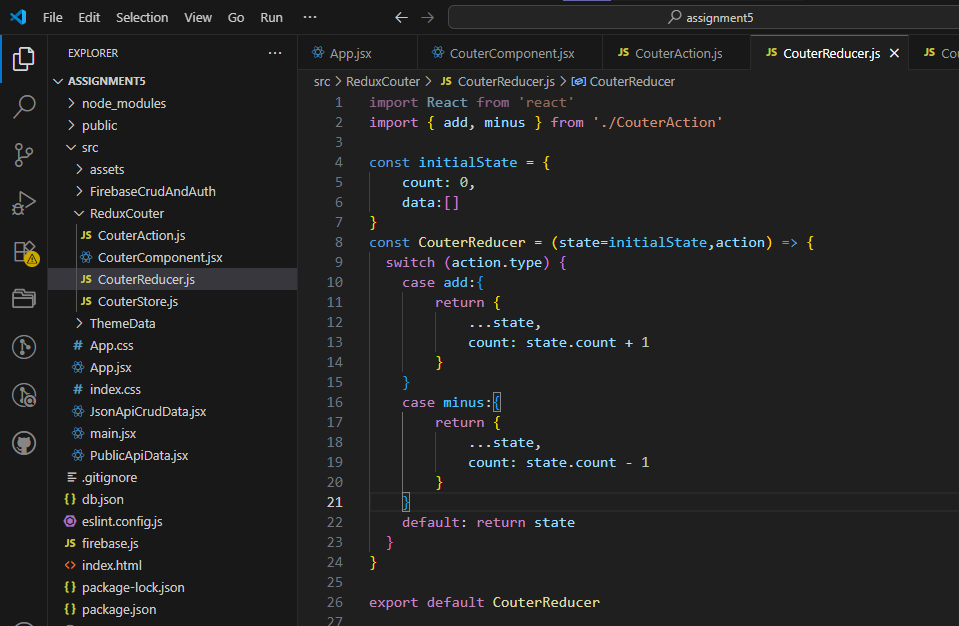
Ans: - Recoil is a state management library for React developed by Facebook. It offers a simpler, more React-like way to handle shared and global state, especially compared to Redux.

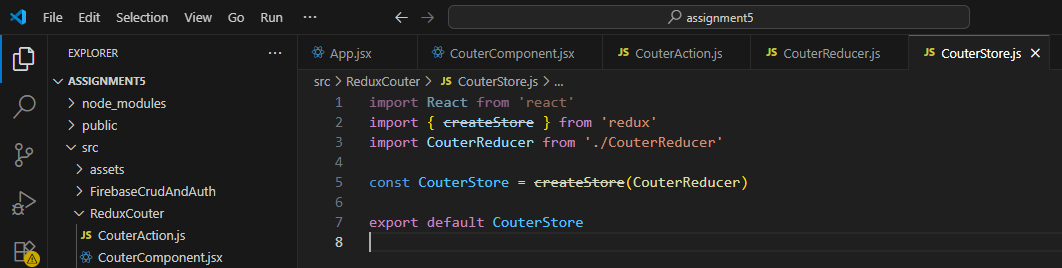
**Lab Tasks**

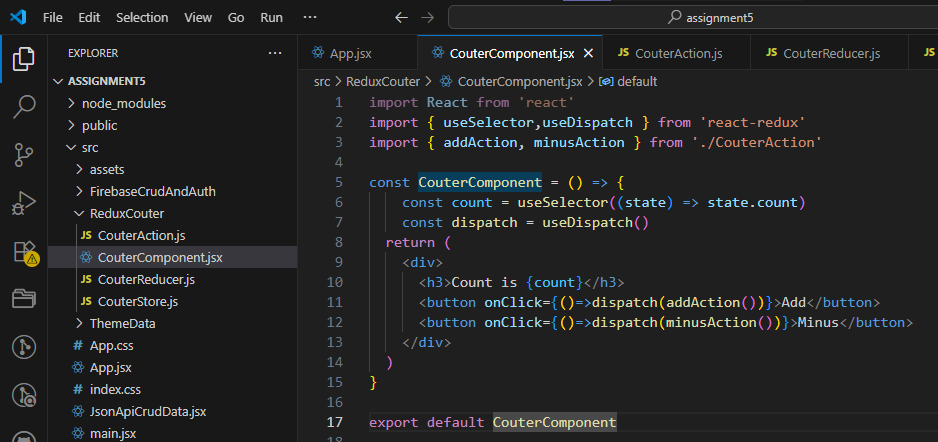
**Task 1 : Create a simple counter application using Redux for state management. Implement actions to increment and decrement the counter.**

**Code:**

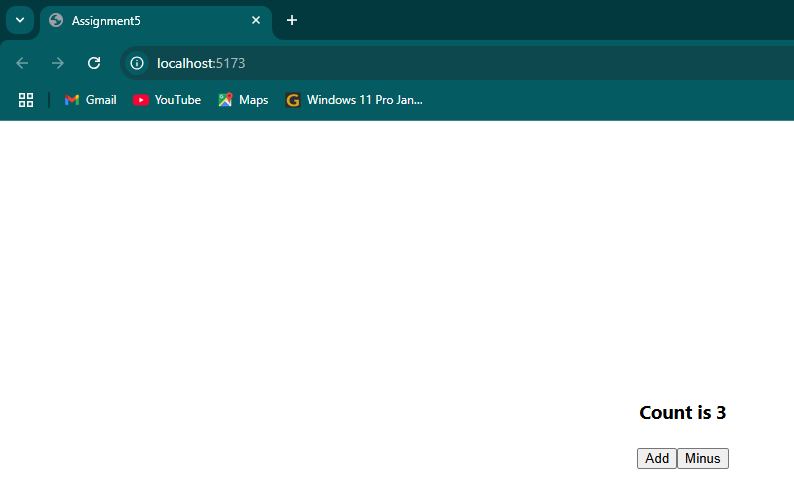


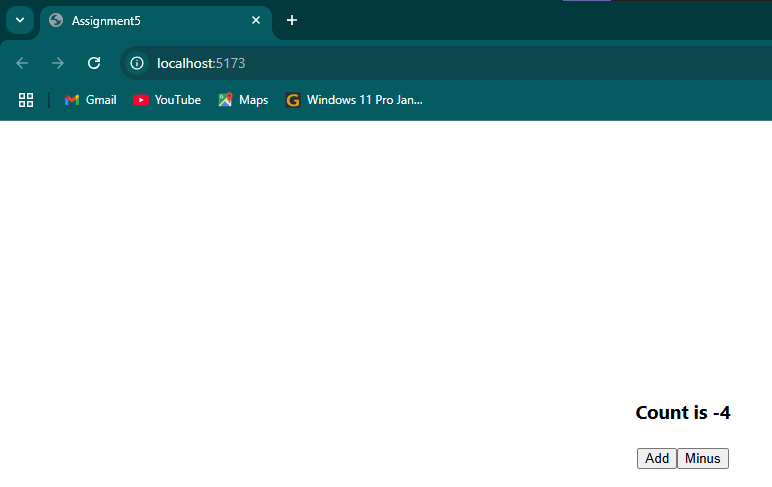






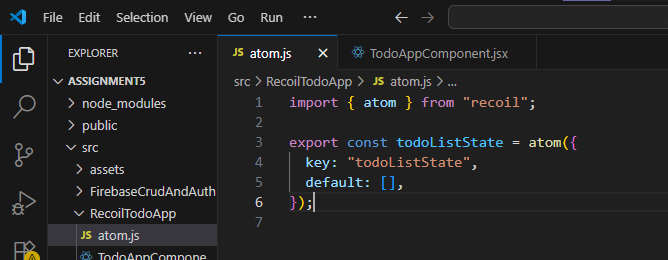
**Output:**

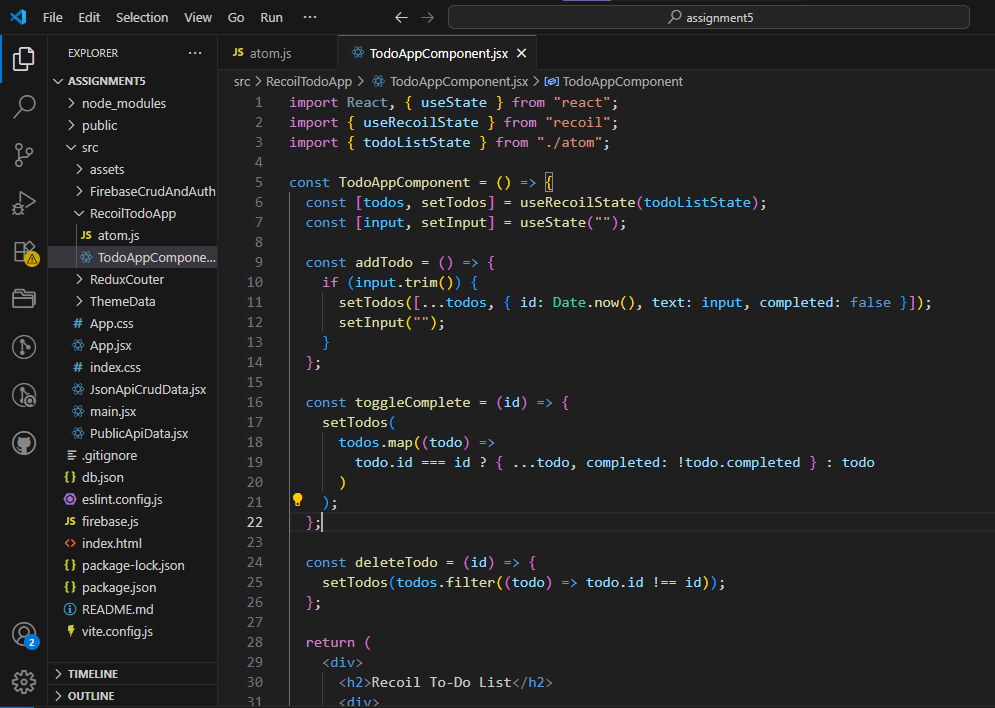
****

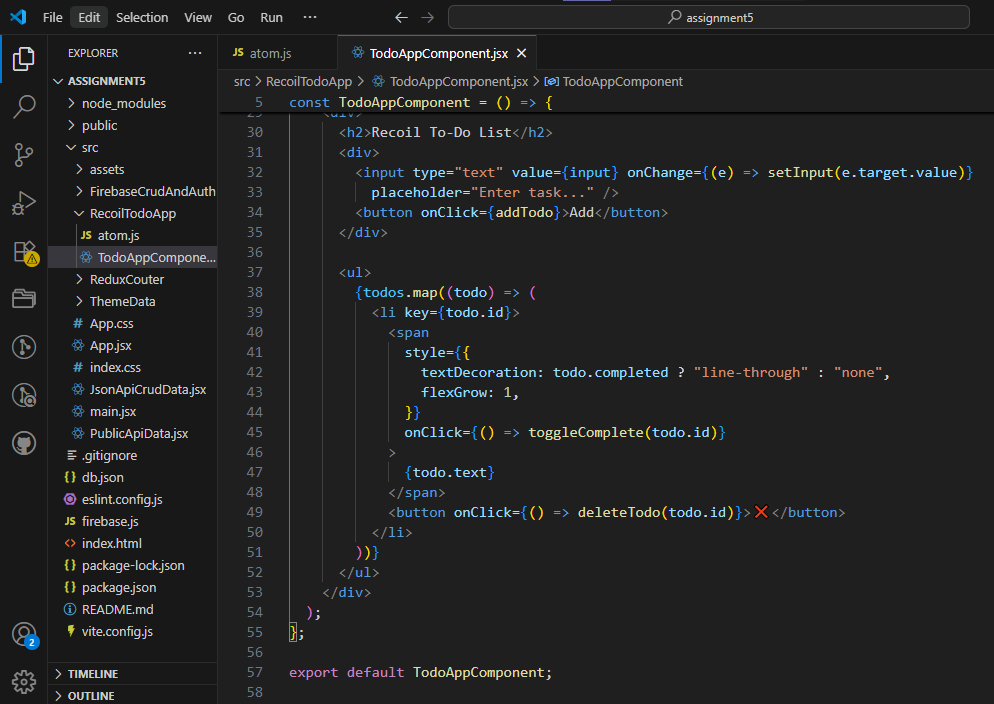
****

**Task 2 : Build a Todo list application using Recoil for state management. Allow users to add, remove, and mark tasks as complete.**

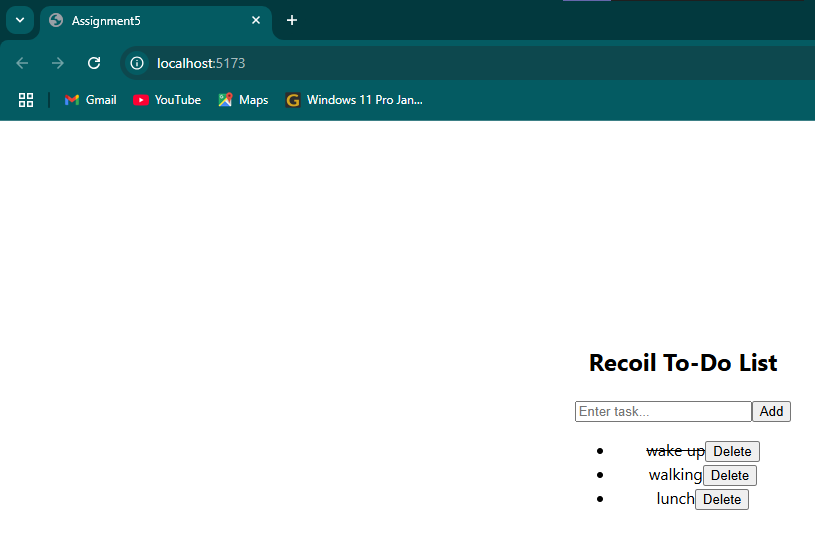
**Code:**

****

****

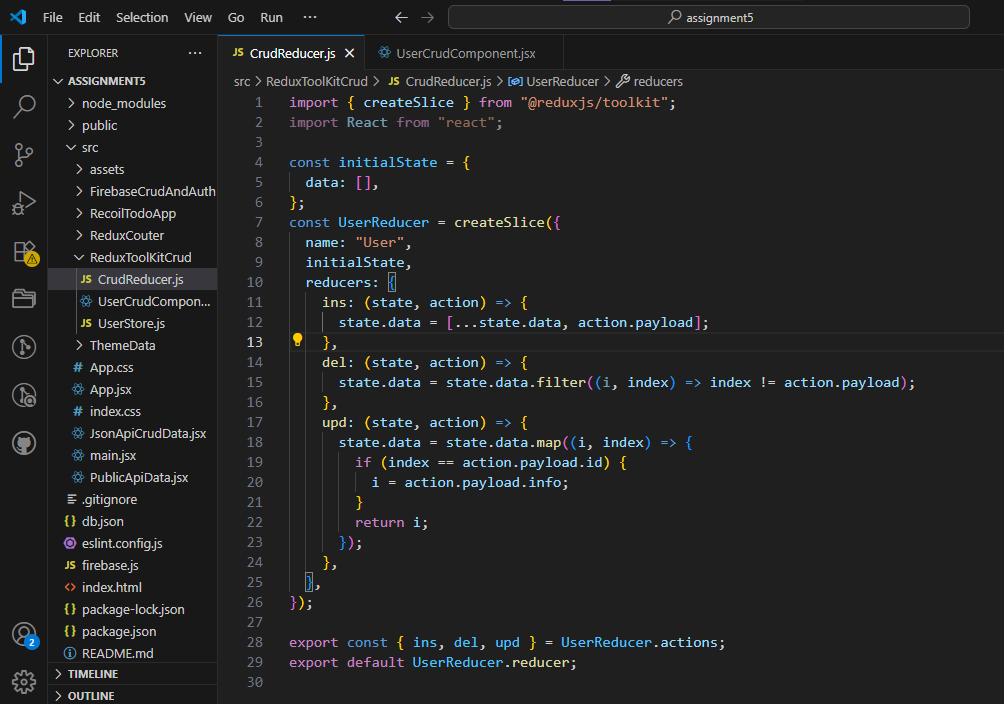
****

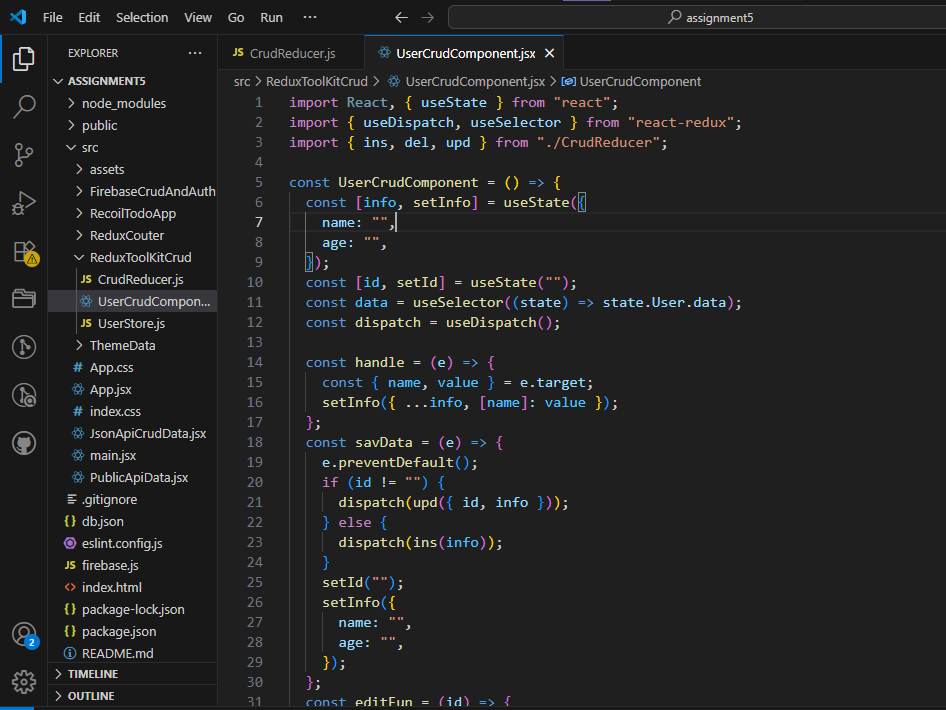
**Output:**

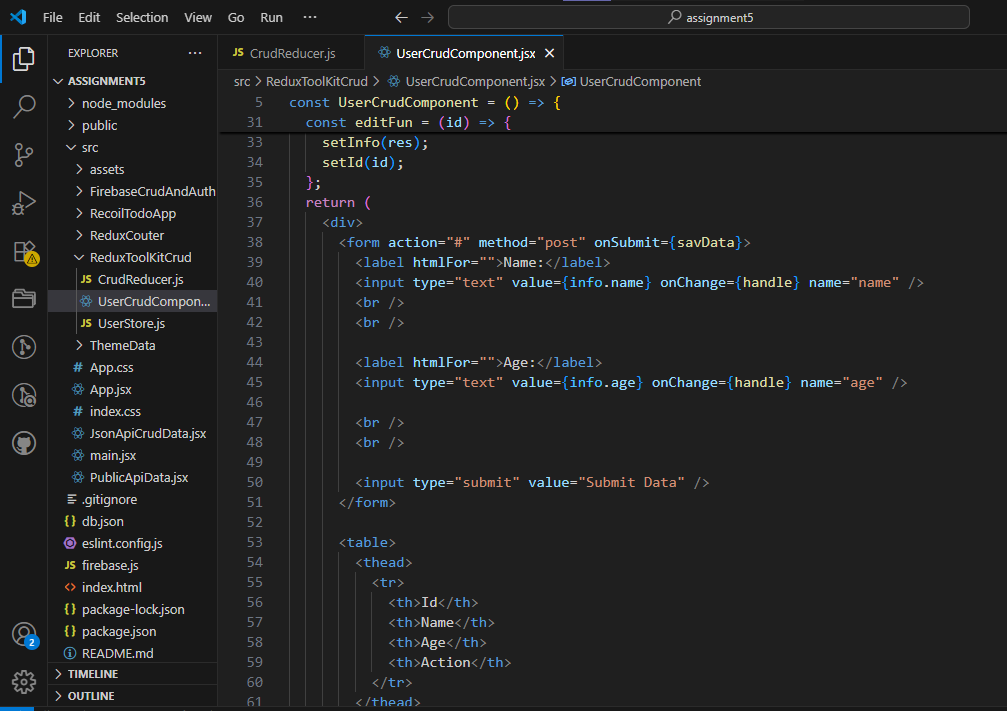
****

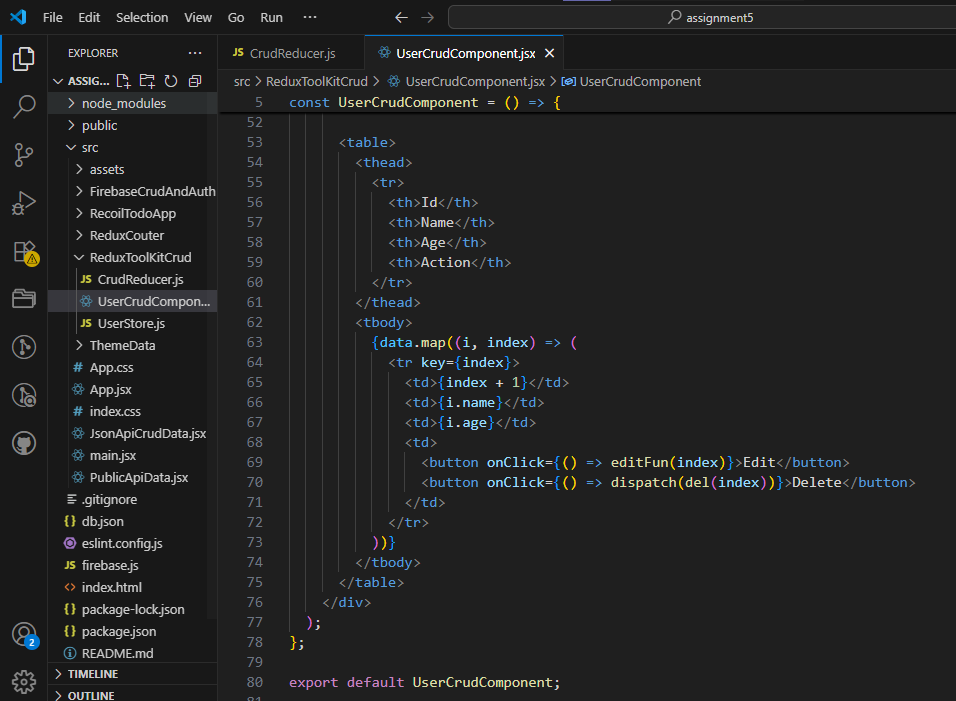
**Task 3 : Build a crud application using Redux-Toolkit for state management. Allow users to add, remove, delete and update.**

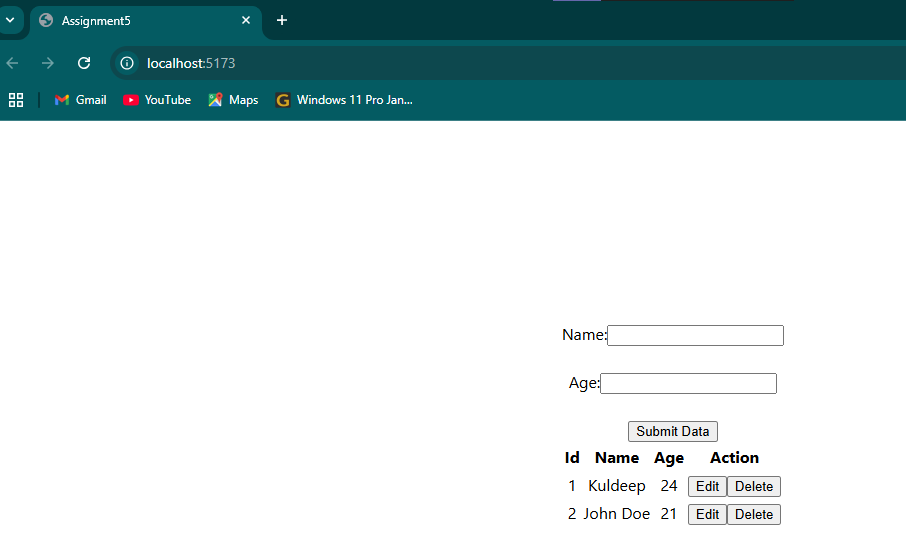
**Code:**

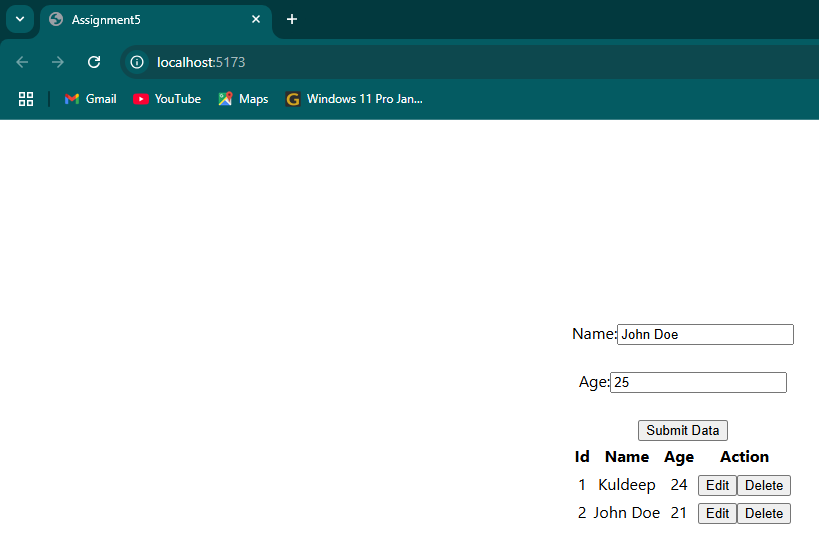
****

****

****

****

****

****