T3 SKILLS CENTER

Foundation Day
30-09-2023

T3 कौशल केंद्र

TWKSAA REACT JS

Er. Rajesh Prasad

- आप एक सागर हो बहते नदी का जल नहीं आप एक बदलाव हो भटकाव की कोई राह नहीं
- उस रास्ते पर चलो जिस रास्ते पर भीड़ कम हो (हर काम हो कुछ अलग)
- देश की मिट्टी से करो आप इतना प्यार जहाँ जाओ वहाँ मिले खूब इज्जत और सम्मान
- छह दिन कीजिए अपना काम एक दिन कीजिए त्वक्सा को दान
- त्वक्सा एक चिंगारी हैं हर जगह जलना हम सब की जिमेवारी हैं
- **Motive: - New (RID PMS & TLR)**

"त्वक्सा रियेक्ट जेयस के इस पुस्तक में आप कोर रियेक्ट जेयस के संबंध में सभी बुनियादी अवधारणाएँ सीखेंगे। मुझे आशा है कि इस पुस्तक को पढ़ने के बाद आपके ज्ञान में वृद्धि होगी और आपको कंप्यूटर विज्ञान के बारे में और अधिक जानने में रुचि होगी"

"In this TWKSAA React Js book you will learn all basic concept regarding React Js. I hope after reading this book your knowledge will be improve and you will get more interest to know more thing about computer Science".

"Skill कौशल एक व्यक्ति के पास उनके ज्ञान, अनुभव, तत्वशास्त्रीय योग्यता, और प्रैक्टिकल अभियांत्रिकी के साथ संचित नौकरी, व्यापार, या अन्य चुनौतीपूर्ण परिस्थितियों में सक्रिय रूप से काम करने की क्षमता को कहते हैं। यह व्यक्ति के द्वारा सीखी जाने वाली कौशलों की प्रतिभा, क्षमता और निपुणता को संक्षेप में व्यक्त करता है"।

TWKSAA RID MISSION

(Research)	(Innovation)	(Discovery)
अनुसंधान करने के महत्वपूर्ण कारण:	**नवीनीकरण करने के महत्वपूर्ण कारण:**	**खोज करने के महत्वपूर्ण कारण:**
1. नई ज्ञान की प्राप्ति	1. प्रगति के लिए	1. नए ज्ञान की प्राप्ति
2. समस्याओं का समाधान	2. परिवर्तन के लिए	2. ज्ञान के विकास में योगदान
3. तकनीकी और व्यापार में उन्नति	3. उत्पादन में सुधार	3. अविष्कारों की खोज
4. विकास को बढ़ावा देना	4. प्रतिस्पर्धा में अग्रणी होने के लिए	4. समस्याओं का समाधान
5. सामाजिक प्रगति	5. समाज को लाभ	5. समाज के उन्नति का माध्यम
6. देश विज्ञान और प्रौद्योगिकी का विकास	6. देश विज्ञान और प्रौद्योगिकी के विकास।	6. देश विज्ञान और तकनीक के विकास

"T3 Skills Center is a Learning Earning and Development Based Skill Center."

T3 कौशल केंद्र एक सीखने कमाई और विकास आधारित कौशल केंद्र है।

REACT JS

- React, is an open-source JavaScript library for building user interfaces (UIs) and single-page applications (SPAs).
- it is component-based front end library responsible only for the view layer of the application.

HISTORY

1) Internal Development at Facebook (2011-2013):
- React was initially developed by Jordan Walke, a software engineer at Facebook.
- It was first used in Facebook's newsfeed in 2011 as a way to improve the performance and efficiency of rendering user interfaces.

2) Open-Sourced by Facebook (May 2013):
- React was open-sourced by Facebook at the JSConf US conference in May 2013.
- It was released under the MIT License, making it free for anyone to use and contribute to.

3) Early Adoption (2013-2014):
- React quickly gained attention from the developer community due to its innovative approach to building user interfaces with a virtual DOM.
- Various companies and projects began adopting React for their web applications.

4) Introduction of JSX (May 2013):
- React introduced JSX, a syntax extension for JavaScript that allowed developers to write HTML-like code within JavaScript, making it easier to define component structures.

5) Flux Architecture (2014):
- Facebook introduced Flux, a design pattern for managing the flow of data in React applications. While not a part of React itself, Flux became a popular choice for state management in React apps.

6) React Native (March 2015):
- React Native was announced by Facebook. It's a framework for building mobile applications using React, allowing developers to use React to build native mobile apps for iOS and Android.

7) Release of React 0.14 (October 2015):
- React 0.14 introduced functional stateless components, enabling developers to create simple, stateless UI components more easily.

8) React Fiber (September 2017):
- React Fiber, a complete rewrite of the React core algorithm, was introduced to improve performance and enable support for concurrent rendering, making React even more efficient.

9) React Hooks (February 2019):
- React Hooks were introduced, providing a new way to manage state and side effects in functional components. This simplified component logic and reduced the need for class components.

10) React Concurrent Mode (Experimental):
- React Concurrent Mode, an experimental feature, was introduced to allow concurrent rendering of components, potentially improving the user experience by prioritizing important updates.

REACT VERSION

➢ Certainly, list of React.js versions and their associated functions or features introduced.

1) . React 0.3 (May 2013):
- Initial open-source release by Facebook.

2) . React 0.4 (June 2013):
- Introduction of server-side rendering (React Server).

3) . React 0.5 (August 2013):
- Improved performance with various optimizations.
- Enhanced server-side rendering capabilities.

4) . React 0.8 (December 2013):
- Introduction of key reconciliation algorithm for more efficient updates.

5) . React 0.9 (February 2014):
- Improved support for non-browser environments (e.g., React Native).

6) . React 0.10 (April 2014):
- Introduced PropTypes for defining component prop types.

7) . React 0.11 (June 2014):
- Improved support for SVG elements.

8) . React 0.12 (October 2014):
- Improved performance and memory usage.
- Introduced context API (React.createClassContext).

9) . React 0.13 (March 2015):
- Introduction of stateless functional components.
- Support for ES6 classes in addition to createClass. React Perf tools for profiling and debugging.

10) React 0.14 (October 2015):
- Improved support for stateless functional components.
- Removal of \`React.createClass\`. Introduction of \`React.Component\` and class-based components. Added support for refs on functional components.

11) React 15 (April 2016):
- Improved performance with optimizations to the React core.
- Enhanced support for SVG.
- React.PropTypes moved to a separate prop-types package.

12) React 16 (September 2017):
- Introduction of React Fiber, a complete rewrite of core reconciliation algorithm for better performance Support for returning arrays and strings as components. Error boundaries for better error handling. Improved server-side rendering.

13) React 16.3 (March 2018):
- Context API improvements and a separate \`React.createContext\` method.
- Introduction of the new \`getDerivedStateFromProps\` lifecycle method.

14) React 16.8 (February 2019):
- Introduction of React Hooks, enabling state and side effects in functional components.
- \`useState\`, \`useEffect\`, \`useContext\`, and other hooks introduced.

15) React 17 (October 2020):
- No new major features introduced. Focused on making it easier to upgrade between versions.
- Laying the groundwork for future concurrent rendering features.

FEATURES OF REACT JS

1. **Component-Based Architecture:** React applications are built using reusable components. Each component encapsulates a part of the user interface and its behavior. This promotes code reusability and maintainability.
2. **Virtual DOM:** React uses a virtual representation of the DOM (Virtual DOM) to optimize updates. Instead of directly manipulating the actual DOM, React calculates the difference between the virtual DOM and the real DOM, making updates more efficient.
3. **Declarative Syntax:** React encourages a declarative approach to building UIs. Developers describe what UI should look like based on application's state, and React takes care of updating DOM to match that description.
4. **Reconciliation:** React efficiently updates the DOM by minimizing the number of changes required. It uses a process called "reconciliation" to determine the most efficient way to update the user interface when the application's state changes.
5. **Unidirectional Data Flow:** React enforces a one-way data flow, meaning data flows from parent components to child components. This makes it easier to understand how data changes affect the UI and helps prevent bugs related to data mutation.
6. **Component Lifecycle:** React components have a lifecycle with methods that are automatically called at different stages of a component's existence. These methods allow developers to perform setup, update, and cleanup operations
7. **JSX (JavaScript XML):** React uses JSX, a syntax extension for JavaScript, to define component structures. JSX allows developers to write HTML-like code within JavaScript, making it easier to visualize and work with components.
8. **Conditional Rendering:** React enables conditional rendering of components based on the application's state. This allows for dynamic UIs that change in response to user interactions.
9. **State Management:** React provides a built-in way to manage component state. Component state allows developers to store and manage data that can change over time, such as user input or application data.
10. **Props (Properties):** React components can receive data through props, which are read-only and passed down from parent to child components. Props are a way to pass information to child components and make them configurable.
11. **Event Handling:** React supports event handling by allowing developers to define event listeners within components. This makes it easy to respond to user interactions like clicks, input changes.
12. **Server-Side Rendering (SSR):** React supports server-side rendering, allowing you to render React components on server and send HTML to the client. This improves initial page load times and SEO
13. **React Native:** React can be used to build mobile applications for iOS and Android through React Native. It allows developers to write code in React and deploy it as a native mobile app.
14. **Large Ecosystem:** React has a vast ecosystem of libraries and tools, including state management solutions like Redux and MobX, routing libraries like React Router, and UI component libraries like Material-UI.
15. **Community and Resources:** React has a large and active community of developers, which means there are plenty of resources, tutorials, and third-party packages available to help with development and problem-solving.

REACT ENVIRONMENT SETUP

➢ Setting up a development environment for React involves several steps, including installing Node.js, npm (Node Package Manager), and a code editor. Her step-by-step guide for you

1. Install Node.js and npm:

- React applications rely on Node.js and npm for development and package management.
- Download and install Node.js from the official website: Node.js Downloads.
- verify Node.js and npm installation by running these commands **Example:** node -v, & npm -v

2. Choose a Code Editor:

- You'll need a Text editor for writing React code. Like VS Code, Sublime Text, Atom, or WebStorm.

3. Create a React Application:

❖ **Option 1:** Using Create React App:

CREATE-REACT-APP

❖ Create a new React project folder and run the following command

❖ 1. npm install -g create-creat-app

❖ 2. To create a new React app using Create React App, open your terminal and run the following commands: **create-react-app appname** or npx create-react-app appname

PS C:\Users\VFS-47\Desktop\skills> **create-react-app twksaa**

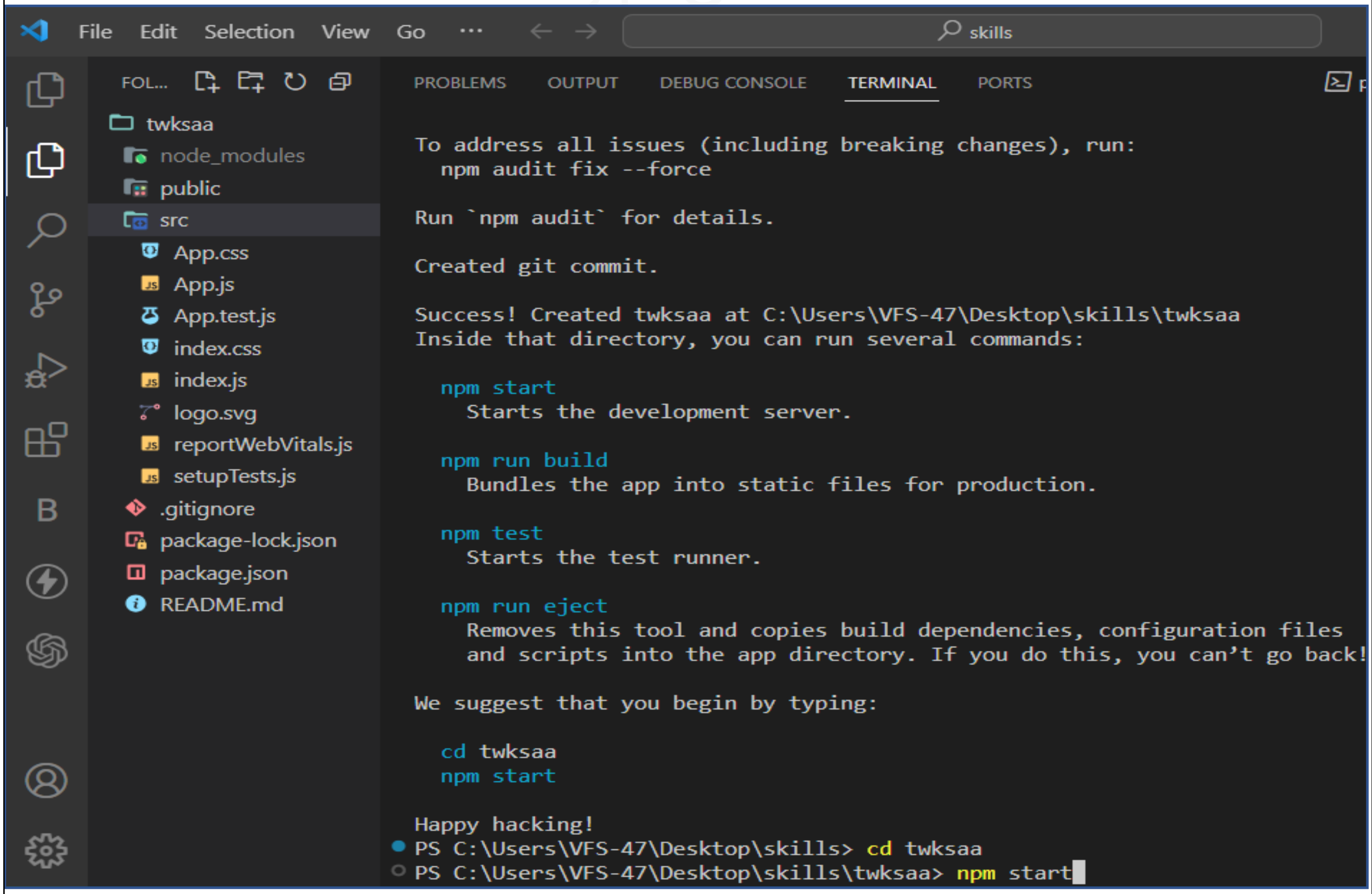
Creating a new React app in C:\Users\VFS-47\Desktop\skills\twksaa.

....

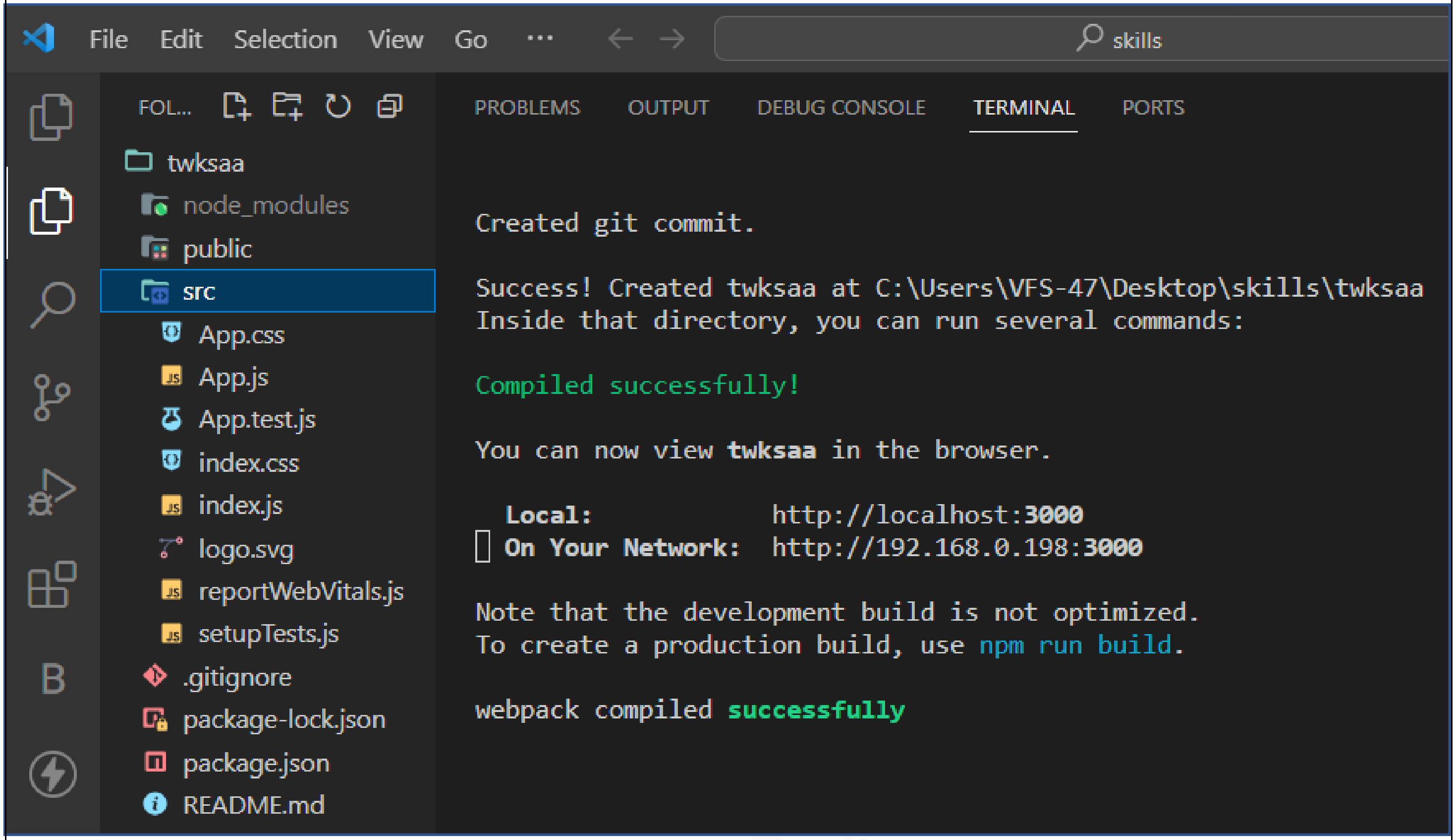
Success! Created twksaa at C:\Users\VFS-47\Desktop\skills\twksaa

Here twksaa is app name

Press Enter



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

Run `npm audit` for details.

Created git commit.

Success! Created twksaa at C:\Users\VFS-47\Desktop\skills\twksaa
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd twksaa
  npm start

Happy hacking!
PS C:\Users\VFS-47\Desktop\skills> cd twksaa
PS C:\Users\VFS-47\Desktop\skills\twksaa> npm start
```



```
Created git commit.

Success! Created twksaa at C:\Users\VFS-47\Desktop\skills\twksaa
Inside that directory, you can run several commands:

Compiled successfully!

You can now view twksaa in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.198:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

- **node_modules:** It contains the React library and any other third party libraries needed.
- **public:** It holds the public assets of the application. It contains the index.html where React will mount the application by default on the <div id="root"></div> element.
- **src:** It contains the App.css, App.js, App.test.js, index.css, index.js, and serviceWorker.js files. Here, the App.js file always responsible for displaying the output screen in React.
- **package-lock.json:** It is generated automatically for any operations where npm package modifies either the node_modules tree or package.json. It cannot be published. It will be ignored if it finds any other place rather than the top-level package.
- **package.json:** It holds various metadata required for the project. It gives information to npm, which allows to identify the project as well as handle the project?s dependencies.
- **README.md:** It provides the documentation to read about React topics.
- **Now, open the src >> App.js file and make changes which you want to write code.**

❖ if any react application is running then how to stop:

webpack compiled successfully

step-1: press ctrl +c

step-2 Terminate batch job (Y/N)? press y

step-3: again we can start now

```
PS C:\Users\VFS-47\Desktop\rajreact\raj> npm start
raj@0.1.0 start
 Local:          http://localhost:3000
 On Your Network:  http://192.168.0.198:3000
```

- Note that the development build is not optimized.

➢ To create a production build, use npm run build.

- webpack compiled successfully

❖ **Option 2:** Manual Setup

- If you want more control over the project configuration, you can set up a React app manually.

  Create a new directory for your project:
  ```
  mkdir my-react-app
  cd my-react-app
  ```
  Initialize a new Node.js project by running:
  ```
  npm init -y
  ```
  Install React and ReactDOM as dependencies:
  ```
  npm install react react-dom
  ```
  Create an HTML file (e.g., index.html) and a JavaScript file (e.g., index.js) in your project directory.

  Write your React code in index.js, and set up your HTML file to include the necessary scripts and a root <div> element where React will render your app.

4. Write Your React Code:

- Open your code editor and start writing React components in your JavaScript files

  example:
  ```
  import React from 'react';
  import ReactDOM from 'react-dom';
  const App = () => {
    return <h1>Hello, React!</h1>;
  };
  ReactDOM.render(<App />, document.getElementById('root'));
  ```

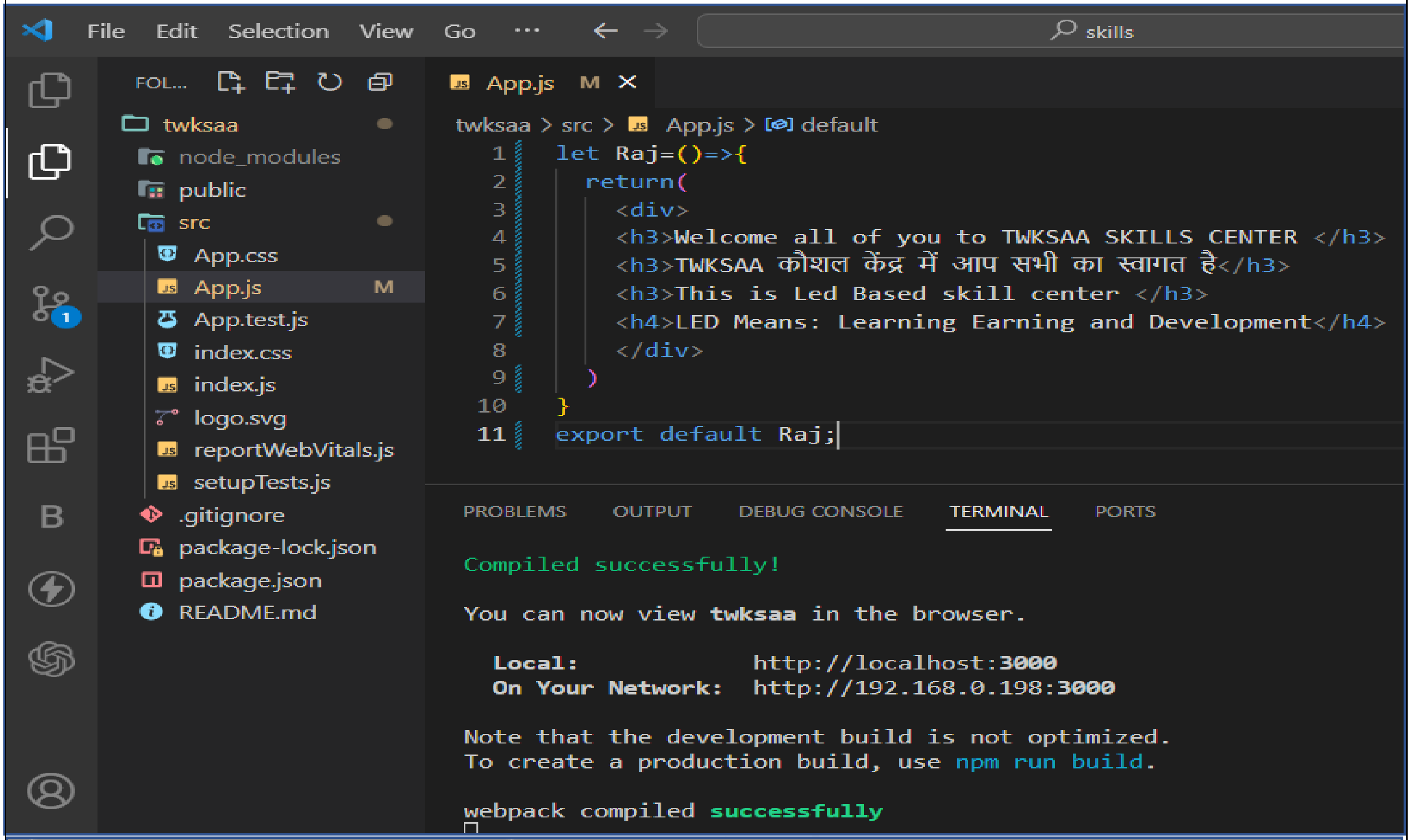
5. Run Your React Application:

- If you're using Create React App, the development server will automatically start when you run npm start. Your React app will be accessible in your web browser at http://localhost:3000.
- If you set up your project manually, you may need to configure a development server (e.g., webpack-dev-server) to run your app.

6. Start Developing:

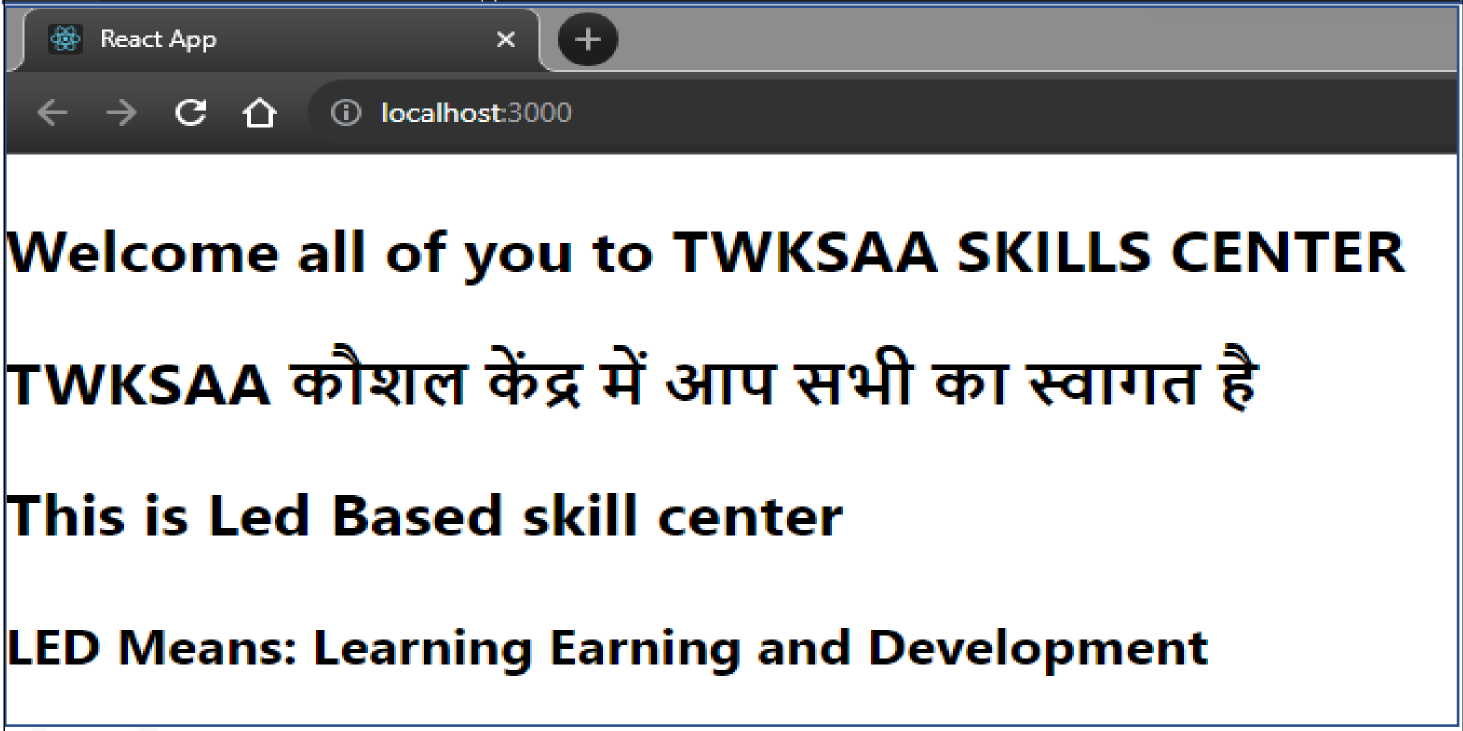
- You can now start building your React application, creating components, managing state, and adding functionality as needed.

Example:
```
let App=()=>{
 return(
  <div>
   <h1>Welcome all of you to Twksaa Skills Center.</h1>
   <h3>Twksaa कौशल केंद्र में आप सभी का स्वागत है।</h3>
   <h2>This is Led Based skill Center</h2>
   <h2>Led Means: Learining Earning and development</h2>
  </div>
 )

}

export default App;
```



```
let Raj=()=>{
  return(
    <div>
    <h3>Welcome all of you to TWKSAA SKILLS CENTER </h3>
    <h3>TWKSAA कौशल केंद्र में आप सभी का स्वागत है</h3>
    <h3>This is Led Based skill center </h3>
    <h4>LED Means: Learning Earning and Development</h4>
    </div>
  )
}
export default Raj;
```

```
Compiled successfully!

You can now view twksaa in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.198:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



Welcome all of you to TWKSAA SKILLS CENTER

TWKSAA कौशल केंद्र में आप सभी का स्वागत है

This is Led Based skill center

LED Means: Learning Earning and Development

- **To create a production build, use npm run build.**

```
                webpack compiled successfully
PS C:\Users\VFS-47\Desktop\rajreact\raj> npm run build
                > raj@0.1.0 build
                > react-scripts build
                Creating an optimized production build...
                Compiled successfully.
                File sizes after gzip:
```

```
  46.43 kB  build\static\js\main.2791fa93.js
  1.78 kB   build\static\js\787.6df0c601.chunk.js
  264 B     build\static\css\main.e6c13ad2.css
The project was built assuming it is hosted at /.
```

➢ You can control this with the homepage field in your package.json.

```
The build folder is ready to be deployed.
You may serve it with a static server:
  npm install -g serve
  serve -s build
```

➢ Find out more about deployment here:

```
  https://cra.link/deployment
```

➢ build file are used for real time depolyment or production:

❖ Install Webpack:

- Webpack is used for module packaging, development, and production pipeline automation. We will use webpack-dev-server during development, webpack to create production builds, and webpack CLI provides a set of commands. Webpack compiles these into a single file(bundle).
- To install Webpack, you'll need Node.js and npm (Node Package Manager) installed on your system. Webpack is typically installed globally as a development tool.

1. Check if Node.js and npm are installed:

- Open your terminal or command prompt and run

```
node -v
npm -v
```

- If Node.js and npm are not installed, download and install them from the official website: Node.js Downloads.

2. Install Webpack Globally:

- To install Webpack globally on your system, use the following command:

```
npm install -g webpack
```

- This command tells npm to install the webpack package globally, making it available as a command-line tool throughout your system.

3. Verify Webpack Installation:

- After installation is complete, you can verify that Webpack is installed by running:

❖ **webpack -v**

- This should display the installed version of Webpack.

4. Create a Webpack Configuration:

- Webpack is highly configurable and typically requires a configuration file (usually named webpack.config.js) to define how your project should be bundled. You can create this configuration file in the root directory of your project.

❖ Here's a basic example of a webpack.config.js file:

```
const path = require('path');
module.exports = {
  entry: './src/index.js',   // Entry point of your application
  output: {
    filename: 'bundle.js',   // Output file name
    path: path.resolve(__dirname, 'dist'),  // Output directory
  },
};
```

- In this example, we specify the entry point (entry) and the output file (output) for our bundled JavaScript. You can customize this configuration to suit your project's needs.

5. Use Webpack in Your Project:

- Now that Webpack is installed and you have a configuration file, you can use it to bundle your project's JavaScript files. Here's how to run Webpack:

```
webpack --config webpack.config.js
```

- replace webpack.config.js with the name of your configuration file if it's different.

REACT COMPONENT

- component is reusable, self-contained, and modular building blocks in React applications that represent specific parts of the user interface.
- Each component exists in the same space, but they work independently from one another and merge all in a parent component,
- Every React component have their own structure, methods as well as APIs.

❖ Types of React component:

❖ There are two types of components

1. functional component
2. class component

Function component

- function components are a way to write components that only contain a render method and don't have their own state.
- it is also known as stateless components or presentational components.
- Functional Components are used for rendering UI elements based on the data (props) they receive and don't have their own internal state

❖ characteristics of Functional Components:

1. **Function Definition:** Functional Components are defined as JavaScript functions that take a set of properties (props) as input and return JSX (JavaScript XML) to describe what should be rendered in the user interface.

Example:

```
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}
```

2. **No Internal State:** Functional Components do not have their own state. They are entirely dependent on the data passed to them via props for rendering.
3. **Pure Functions:** They are essentially pure functions, meaning that for the same input (props), they will always produce the same output. This predictability simplifies debugging and testing

Class component

- Class Component is a type of component that is implemented as a JavaScript class. Class components are also known as stateful components or container components. They are used to create components with more advanced features, such as managing their own internal state and handling lifecycle methods.

❖ characteristics of Class Components:

1. Class Definition: Class Components are defined by creating a JavaScript class that extends the React.Component class or its subclass. This class definition typically includes a render() method, which describes what should be rendered in the user interface.

Example:

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }
  increment() {
    this.setState({ count: this.state.count + 1 });
  }
  render() {
    return (
      <div>
        <p>Count: {this.state.count}</p>
        <button onClick={() => this.increment()}>Increment</button>
      </div>
    );  }}
```

2.Internal State: Class Components have the ability to manage their own internal state using this.state. This allows them to store and update data that affects their rendering.

3.Lifecycle Methods: Class Components can define lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount. These methods provide control over when certain actions should be performed during the component's lifecycle.

4. Complex Logic: Class Components are often used when a component needs to manage complex state, handle side effects (e.g., data fetching), or perform actions in response to lifecycle events.

How to create functional component:

- step-by-step guide for create functional component:

Step 1: Setting Up a React Project

- Before you create a functional component, you need to set up a React project. You can do this using tools like `create-react-app` or by configuring your own build setup.
- Assuming you have a React project set up, let's create a functional component:

Step 2: Create a New Component File

- Create a new `.jsx` or `.js` file for your functional component. For example, you can create a file called `MyComponent.jsx`.

Step 3: Import React

- In your component file, start by importing React. Even though functional components don't require you to extend a class like class components, you still need to import React when using JSX.
- import React from 'react';

Step 4: Define the Functional Component

- Create a JavaScript function that represents your functional component. This function should take `props` as its argument (if your component needs to receive any data from its parent).

```
function MyComponent(props) {
  // Component logic and JSX here
}
```

Step 5: Define the Component Logic

- Within your function, you can define the logic for your component. This can include rendering JSX elements based on the data received via `props`. For example:

Example:

```
function MyComponent(props) {
  return (
    <div>
      <h1>Hello, {props.name}!</h1>
      <p>Age: {props.age}</p>
    </div>
  ) }
```

- In this example, the component takes `name` and `age` as props and renders them in JSX.

Step 6: Export the Component

- Export your functional component so that it can be used in other parts of your application.
- export default MyComponent;

Step 7: Use the Functional Component

- Now you can use your functional component in other parts of your React application. Import it and include it in the JSX of another component.

For example:

```
import React from 'react';
import MyComponent from './MyComponent';
function App() {
  return (
    <div>
      <h1>My App</h1>
      <MyComponent name="Alice" age={30} />
    </div>
  ) }
export default App;
```

Step 8: Expected Output

- When you run your React application and render the `App` component, the `MyComponent` functional component will be included, and you will see the expected output in the browser:

```
My App
Hello, Alice!
Age: 30
```

step-1:

❖ **write the component name with .js extension in src file.**

example: Raj.js

Example:

```
let ComponetName=()=>{
   return( //return can return only single value.
      <div></div>
   )}
export default ComponetName;
```

step-2:

❖ import the Raj.js file in App.js

App.js

Example:

```
import Raj from "./Raj"
let App=()=>{
   return(
   <div> </div>
  )}
export default App
```

Example:

❖ **App.js**

```
import Nav from "./Nav"
import Smenu from "./Smenu"
import Foot from "./Foot"
import Mainc from "./Mainc"
import "./App.css"
let App = () => {
  return (
    <div className="con">
      <Nav />
      <div className="mainmenu">
      <Smenu />
      <Mainc />
      </div>
    <Foot />
  </div>
 )}
export default App;
```

❖ **Nav.js**

```
let Nav=()=>{
  return(
    <nav>
    <h3>Navigation Bar</h3>
    </nav>
  ) }
export default Nav;
```

❖ **Smenu.js**

```
let Smenu=()=>{
  return(
    <aside>side menu</aside>
  )}
export default Smenu;
```

❖ **Mainc.js**

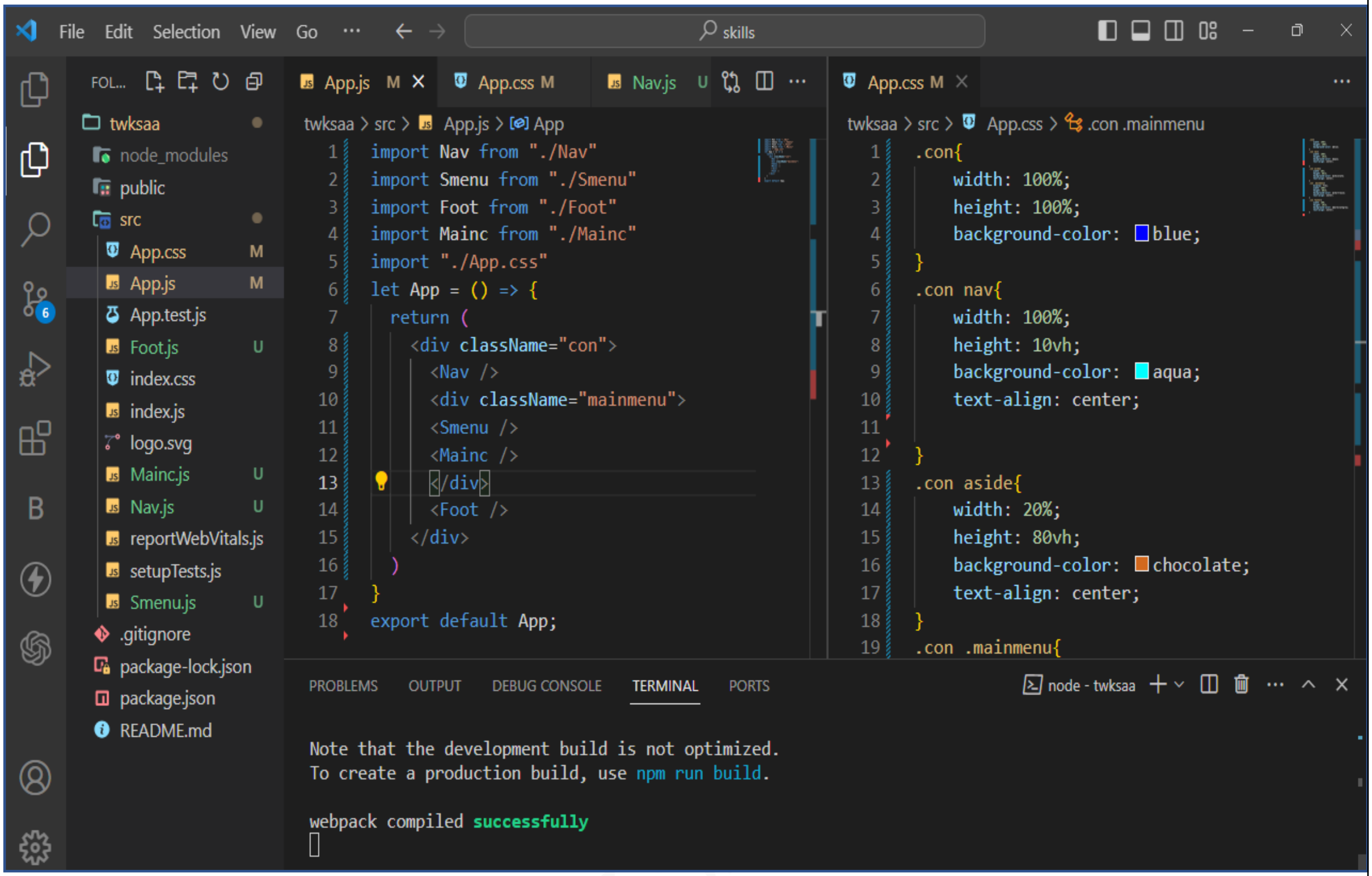
```
let Mainc=()=>{
  return(
    <div>Main Componet</div>
  )}
export default Mainc;
```

❖ **Foot.js**

```
let Foot=()=>{
  return(
    <footer>footer</footer>
  )}
export default Foot;
```

❖ **App.css**

```
.con{
  width: 100%;
  height: 100%;
  background-color: blue; }
.con nav{
  width: 100%;
  height: 10vh;
  background-color: aqua;
  text-align: center; }
.con aside{
  width: 20%;
  height: 80vh;
  background-color: chocolate;
  text-align: center;}
.con .mainmenu{
  display: flex;
  width: 100%;
height: 80vh;
  background-color: chartreuse;
  text-align: center; }
.con footer{
  width: 100;
  height: 10vh;
  background-color: darkslategrey;
  text-align: center;
  color: white;}
```

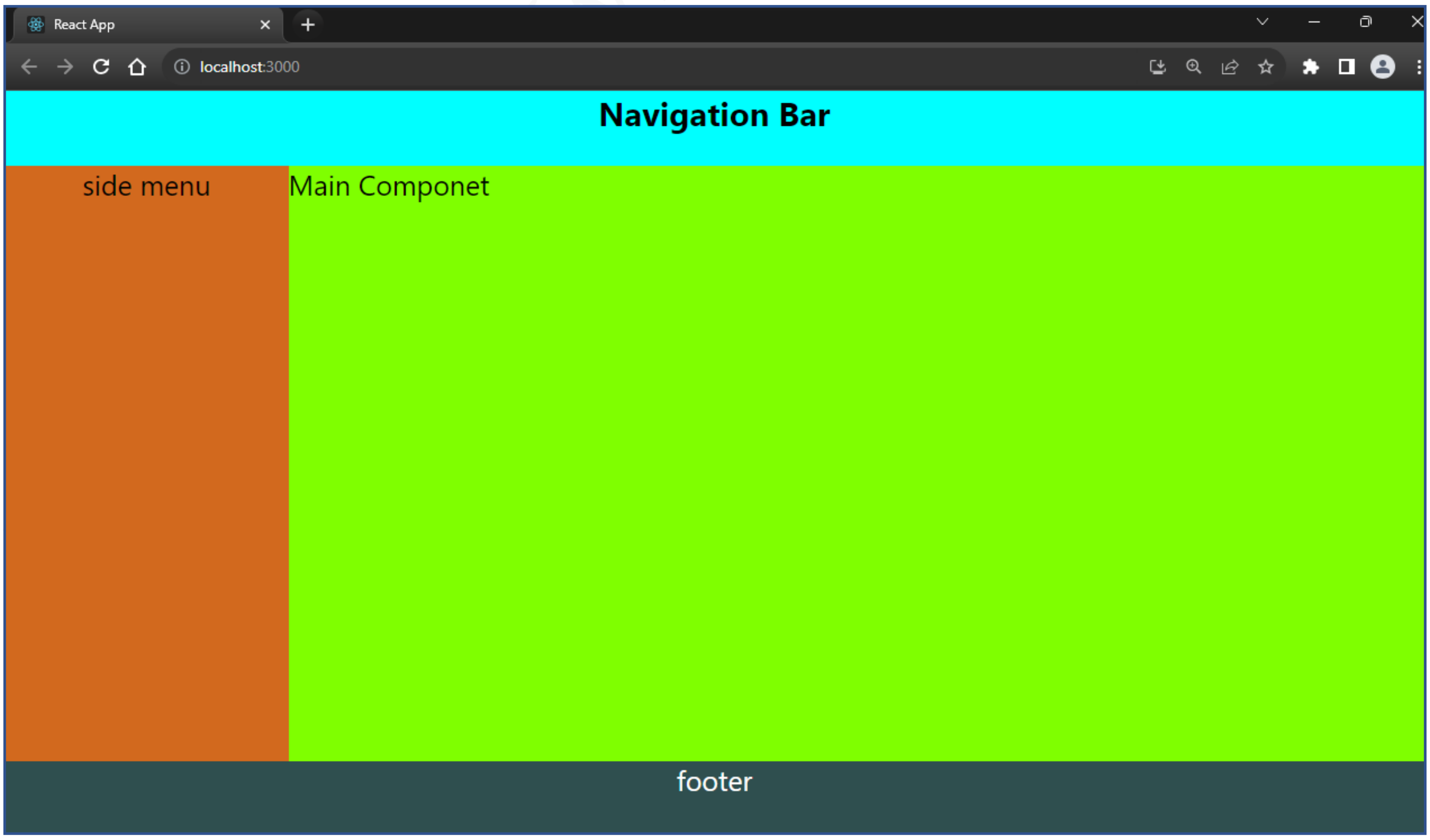


```
import Nav from "./Nav"
import Smenu from "./Smenu"
import Foot from "./Foot"
import Mainc from "./Mainc"
import "./App.css"
let App = () => {
  return (
    <div className="con">
      <Nav />
      <div className="mainmenu">
      <Smenu />
      <Mainc />
      </div>
      <Foot />
    </div>
  )
}
export default App;
```

```
.con{
    width: 100%;
    height: 100%;
    background-color: blue;
}
.con nav{
    width: 100%;
    height: 10vh;
    background-color: aqua;
    text-align: center;

}
.con aside{
    width: 20%;
    height: 80vh;
    background-color: chocolate;
    text-align: center;
}
.con .mainmenu{
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Output:



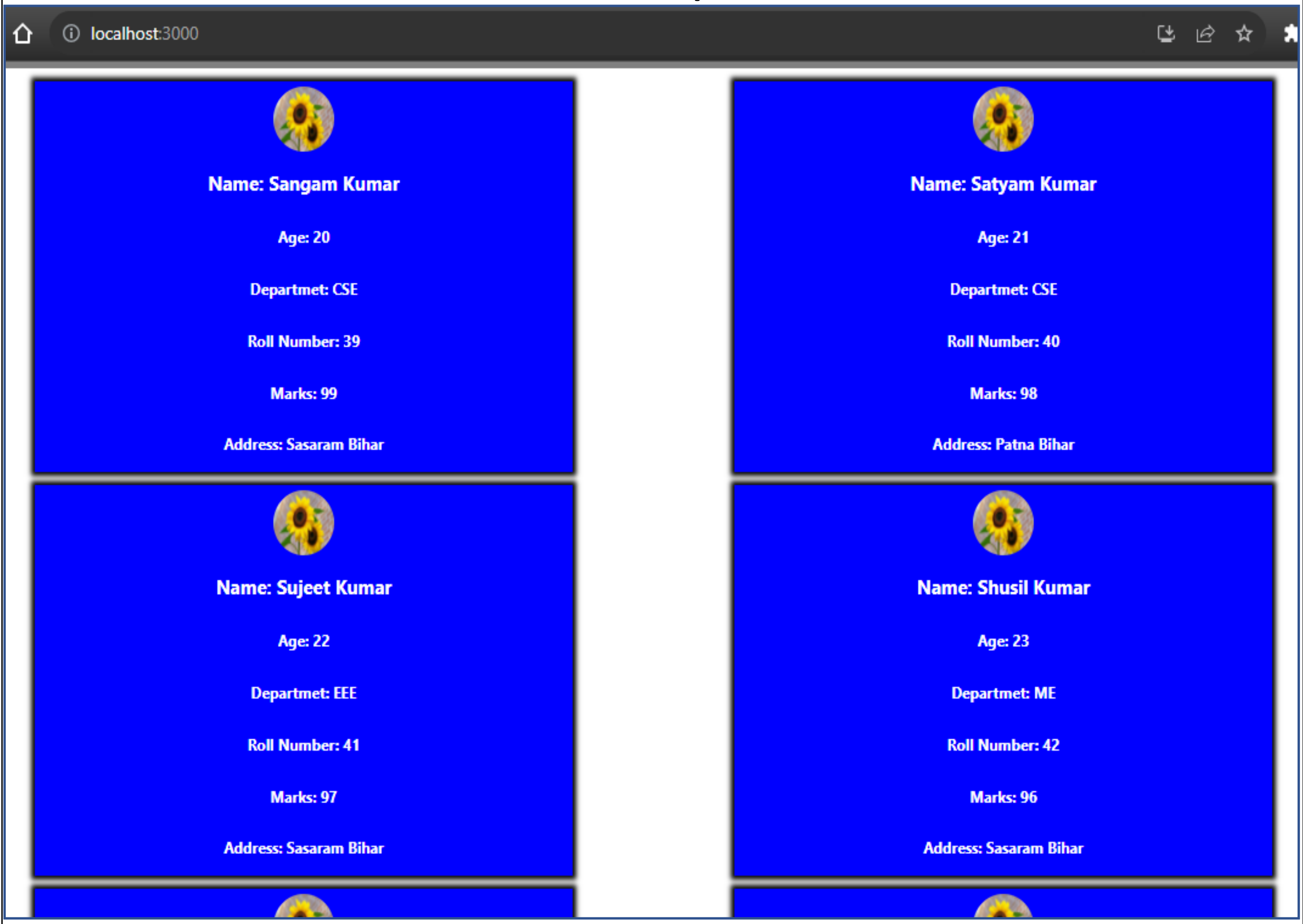
- **Example:**
- **App.js**

```
import './App.css'
let App=()=>{
  let data=[{"name":"Sangam Kumar", "age":20, "dept":"CSE",
"Roll_No":39,"marks":99,"addr":"Sasaram
Bihar","pic":"https://th.bing.com/th/id/OIP.pJCU0PKFKXqhd9SO40OObwHaE8?pid=ImgDet&rs=1"},
  {"name":"Satyam Kumar", "age":21 ,"dept":"CSE", "Roll_No":40,"marks":98,"addr":"Patna
Bihar","pic":"https://th.bing.com/th/id/OIP.pJCU0PKFKXqhd9SO40OObwHaE8?pid=ImgDet&rs=1"},
  {"name":"Sujeet Kumar", "age":22 ,"dept":"EEE", "Roll_No":41,"marks":97,"addr":"Sasaram
Bihar","pic":"https://th.bing.com/th/id/OIP.pJCU0PKFKXqhd9SO40OObwHaE8?pid=ImgDet&rs=1"},
  {"name":"Shushil Kumar", "age":23 ,"dept":"ME", "Roll_No":42,"marks":96,"addr":"Sasaram
Bihar","pic":"https://th.bing.com/th/id/OIP.pJCU0PKFKXqhd9SO40OObwHaE8?pid=ImgDet&rs=1"},
  {"name":"Subham Kumar", "age":24 ,"dept":"CSE", "Roll_No":43,"marks":99,"addr":"Sasaram
Bihar","pic":"https://th.bing.com/th/id/OIP.pJCU0PKFKXqhd9SO40OObwHaE8?pid=ImgDet&rs=1"},
  {"name":"Saurav Kumar", "age":25 ,"dept":"CSE", "Roll_No":44,"marks":99,"addr":"Sasaram
Bihar","pic":"https://th.bing.com/th/id/OIP.pJCU0PKFKXqhd9SO40OObwHaE8?pid=ImgDet&rs=1"}]
  return(
    <div className='con'>
      {
        data.map((obj)=>{
          return(
            <div className='card'>
              <div className='img'><img src={obj.pic}/></div>
              <h1>Name: {obj.name}</h1>
              <h2>Age: {obj.age}</h2>
              <h2>Department: {obj.dept}</h2>
              <h2>Roll Number: {obj.Roll_No}</h2>
              <h2>Marks: {obj.marks}</h2>
              <h2>Address: {obj.addr}</h2>
            </div>
          )
        })
      }
    </div>
  )
}
export default App
```

- **App.js:**

```
.con{
  width: 100%;
  display: flex;
  flex-wrap: wrap;
  justify-content: space-evenly;
  gap:10px;
  padding: 10px 0px;
  border: 5px solid gray ;}
.card{
  width: 35%;
  height: 300px;
  background-color: blue;
  color: white;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-evenly;
  border: 2px double gray;}
.card h1{
  font-size: 15px;}
.card h2{
  font-size: 12px;}
.img{
  width: 50px;
  height: 50px;
  border-radius: 50%;}
.img img{
  width: 100%;
  height: 100%;
  border-radius: 50%;}
```

Output:



```
Name: Sangam Kumar
Age: 20
Departmet: CSE
Roll Number: 39
Marks: 99
Address: Sasaram Bihar

Name: Satyam Kumar
Age: 21
Departmet: CSE
Roll Number: 40
Marks: 98
Address: Patna Bihar

Name: Sujeet Kumar
Age: 22
Departmet: EEE
Roll Number: 41
Marks: 97
Address: Sasaram Bihar

Name: Shusil Kumar
Age: 23
Departmet: ME
Roll Number: 42
Marks: 96
Address: Sasaram Bihar
```

PROPS:

- Props stand for "Properties." They are read-only components.
- It gives a way to pass data from one component to other components. It is similar to function arguments.
- it is a mechanism for passing data from a parent component to a child component.
- props are unidirectional we can send information from parent to child but we can not send data from child to parent.
- props are read only (you can not change content in parent component through props, or through child component you can not change content of parent component.
- react component Ent should be a pure function.
- pure function: for the same input if you will call function n number of times or n-end different place it will give same output.

Example. add max, min etc

❖why we need to send data from parent to child?

❖ passing data from a parent component to a child component is a fundamental concept that serves several important purposes:

1. **Component Composition:** To build complex user interfaces by combining smaller, reusable components.
2. **Reusability:** To make child components reusable in different parts of your application with different data.
3. **Separation of Concerns:** To separate data management and logic in parent components from rendering and presentation in child components.
4. **Maintainability:** To isolate changes to specific components, making your codebase more maintainable.
5. **Predictability:** To ensure a predictable one-way data flow in React applications.
6. Performance Optimization: To efficiently update child components when their props change, improving overall performance.
7. **Testing:** To simplify the testing of individual components by providing different sets of props for various scenarios.

Example-1:

App.js

```
// Parent component
import Greetingprops from "./greetinprops";
function App() {
  return (
    <div>
      <h1>Welcome to My React App</h1>
      <Greetingprops name="Sangam Kumar" />
      <Greetingprops name="Satyam Kumar" />
      <Greetingprops name="Sujeet Kumar" />
    </div>
  );
}
export default App;
```

```
// Child component
greetingprops
function Greetingprops(props)
 {
   return <p>Hello, {props.name}!</p>;

  }
  export default Greetingprops;
```

output:

```
Welcome to My React App
Hello, Sangam Kumar!
Hello, Satyam Kumar!
Hello, Sujeet Kumar!
```

Example-2:

❖ **App.js**

```
import Btn from "./Btn";
//parent componet
let App=()=>{
  return(
    <div>
      <Btn label="ok" color="greed"/>
      <Btn label="Login" color="blue"/>
      <Btn label="Reset" color="red"/>
      <Btn label="Submit" color="orange"/>
    </div>
  )
}
export default App;
```

❖ **Btn.js**

```
// Child component
let Btn=(props)=>{
   return(

         <button style={{"color": props.color}}> {props.label}</button>
   )
}

export default Btn;
```

output:

```
ok Login Reset Submit
```

Example-3:

❖ App.js

```
import Heading from "./Heading"
import Btn from "./Btn"
// Parent component of Heading and Btn
let App=()=>{
  return (
    <div>
     <Heading name="Sangam Kumar" age="15" marks="99"/>
     <Heading name="Satayam Kumar" age="18" marks="98"/>
     <Heading name="Sujeet Kumar" age="21" marks="97"/>
     <Heading name="Shusil Kumar" age="24" marks="96"/>
     <Heading name="Suraj Kumar" age="27" marks="95"/>
     <Heading name="Saroj Kumar" age="30" marks="94"/>
     </div>
  )
}

export default App
```

❖ **Heading.**js

```
import Btn from "./Btn";
// Child component of App.js
let Heading=(props)=>{
    return(
       <div>
       <h3>Your Name is: {props.name} </h3>
       <h4>Your Age is: {props.age} </h4>
       <h4>Your Marks is: {props.marks} </h4>
       <div>
       <Btn label="Add" color="blue"/>
       <Btn label="Delete" color="red"/>
       <Btn label="Update" color="green"/>
     </div>
       </div>
    )
}

export default Heading;
```

❖ **Btn.js**

```
// Child component of heading
let Btn=(props)=>{
    return(
          <button style={{"color": props.color}}> {props.label}</button>
    )
}
export default Btn;
```

output:

```
Your Name is: Sangam Kumar
Your Age is: 15
Your Marks is: 99
Add Delete Update
Your Name is: Satayam Kumar
Your Age is: 18
Your Marks is: 98
```

```
Add Delete Update
Your Name is: Sujeet Kumar
Your Age is: 21
Your Marks is: 97
Add Delete Update
............
............
............
```

❖ Passing variable as props:

Example-1:

❖ **App.js**

```
import Dp from "./Dp";
// Parent component
let App=()=>{
  let name="Sangam Kumar"
  let age=15
  return(
    <div>
      <Dp nm={name} age={age}/>
    </div>
  )
}

export default App;
```

❖ **Dp.js**

```
// Child component
let Dp=(data)=>{
    return(
        <div>
            <h3>Name:{data.nm}</h3>
            <h4>Age:{data.age}</h4>
        </div>
    )
}

export default Dp;
```

output:

```
Name:Sangam Kumar
Age:15
```

Example-2:

❖ **App.js**

```
//sending object as props
import Comp from "./Comp"
//parent componet
let App=()=>{
  let obj={"id":101,"name":"Sangam Kumar","age":15,"marks":99,"add":"Patna"}
  return(
  <Comp data={obj}/>
 )
}
export default App;
```

❖ **Comp.js**

```
let Comp=(props)=>{
   let obj=props.data
   return(
      <div>
         <p>ID:{obj.id}</p>
         <p>Name:{obj.name}</p>
         <p>Age:{obj.age}</p>
         <p>Marks:{obj.marks}</p>
         <p>Address:{obj.add}</p>
      </div>
   )

}

export default Comp;
```

output:

```
ID:101
Name:Sangam Kumar
Age:15
Marks:99
Address:Patna
```

Passing object as props for the child class:

Example:

❖ App.js

```
import Prof1 from "./Profile-1"
import Prof2 from "./Profile-2"
import Prof3 from "./Profile-3"
let App=()=>{
  let obj={"name":"Sangam Kumar", "age":20 ,"dept":"CSE",
"Roll_No":39,"marks":99,"add":"Sasaram Bihar"}
  return(
    <div>
      <Prof1 data={obj}/>
      <Prof2 data={obj}/>
      <Prof3 data={obj}/>
    </div>
  )

}

export default App
```

- **1st method:**
- **profile-1.js**

```
let Prof1=(props)=>{
  let {name,age,dept,Roll_No,marks,add}=props.data
  return (
    <div>
      <h1>Name:{name}</h1>
      <h2>Age:{age}</h2>
      <h3>Departmet:{dept}</h3>
      <h3>Roll Number:{Roll_No}</h3>
      <h3>Marks:{marks}</h3>
      <h3>Address:{add}</h3>
    </div>
  )
}
export default Prof1
```

- **2nd method:**
- **profile-2.js**

```
let Prof2=(props)=>{
  let raj=props.data
  return(
    <div>
      <h1>Name:{raj.name}</h1>
      <h2>Age:{raj.age}</h2>
      <h3>Departmet:{raj.dept}</h3>
      <h3>Roll Number:{raj.Roll_No}</h3>
      <h3>Marks:{raj.marks}</h3>
      <h3>Address:{raj.add}</h3>
    </div>
  )
}
export default Prof2
```

- **3rd method:**
- **profile-3.js**

```
let Prof3=(props)=>{
  return(
    <div>
      <h1>Name:{props.data.name}</h1>
      <h2>Age:{props.data.age}</h2>
      <h3>Departmet:{props.data.dept}</h3>
      <h3>Roll Number{props.data.Roll_No}</h3>
      <h3>Marks:{props.data.marks}</h3>
      <h3>Address:{props.data.add}</h3>
    </div>
  )
}
export default Prof3
```

output:

```
Name:Sangam Kumar
Age:20
Departmet:CSE
Roll Number:39
Marks:99
Address:Sasaram Bihar
Name:Sangam Kumar
Age:20
Departmet:CSE
Roll Number:39
Marks:99
Address:Sasaram Bihar
Name:Sangam Kumar
Age:20
Departmet:CSE
Roll Number39
Marks:99
Address:Sasaram Bihar
```

RENDERING ARRAY OF OBJECT

Example:

❖ **App.js**

```
import Props from "./Profile-1";
let App=()=>{
 let std=[{"name":"Sangam Kumar", "age":20, "dept":"CSE", "Roll_No":39,"marks":99,"add":"Sasaram Bihar"},
 {"name":"Satyam Kumar", "age":21 ,"dept":"CSE", "Roll_No":40,"marks":98,"add":"Patna Bihar"},
 {"name":"Sujeet Kumar", "age":22 ,"dept":"EEE", "Roll_No":41,"marks":97,"add":"Sasaram Bihar"},
 {"name":"Shushil Kumar", "age":23 ,"dept":"ME", "Roll_No":42,"marks":96,"add":"Sasaram Bihar"},
 {"name":"Subham Kumar", "age":24 ,"dept":"CSE", "Roll_No":43,"marks":99,"add":"Sasaram Bihar"},
 {"name":"Saurav Kumar", "age":25 ,"dept":"CSE", "Roll_No":44,"marks":99,"add":"Sasaram Bihar"}]
 return(
  <div>
   { std.map((obj)=> <Props data={obj}/>) }

  </div>
 )
}

export default App;
```

❖ **Profile-1.js**

```
let Prof=(Props)=>{
   let {name,age,dept,Roll_No,marks,add}=Props.data
   return (
      <div>
         <h1>Name: {name}</h1>
         <h2>Age: {age}</h2>
         <h3>Department: {dept}</h3>
         <h3>Roll Number: {Roll_No}</h3>
         <h3>Marks: {marks}</h3>
         <h3>Address: {add}</h3>
      </div>
   )

}
export default Prof
```

output:

```
Name: Sangam Kumar
Age:20
Department: CSE
Roll Number:39
Marks:99
Address: Sasaram Bihar
Name: Satyam Kumar
Age:21
Department: CSE
```

Roll Number:40
Marks:98
Address: Patna Bihar
Name: Sujeet Kumar
Age:22
Department: EEE
Roll Number:41
Marks:97
Address: Sasaram Bihar
Name: Shushil Kumar
Age:23
Department: ME
Roll Number:42
Marks:96
Address: Sasaram Bihar
Name: Subham Kumar
Age:24
Department: CSE
Roll Number:43
Marks:99
Address: Sasaram Bihar
Name: Saurav Kumar
Age:25
Department: CSE
Roll Number:44
Marks:99
Address: Sasaram Bihar

LIST AND KEY:

- **list:** list refers to an array or iterable collection of elements that you want to render in your component's user interface.
  - Lists are commonly used to display dynamic data,
  - When you render a list of elements in a React component, you typically use the map() function to iterate over the list and generate a component or JSX element for each item in the list.

Note: When rendering lists in React components, it's important to understand the concept of keys.

- **Keys in React:**
  - Keys are special attributes that are used by React to uniquely identify elements in a list.
  - They help React identify which items have changed, been added, or been removed when the list is re-rendered.
  - Keys should be unique among sibling elements
  - when you are rendering array of object by the help of some child component to identify every component uniquely, we need to provide one unique from (key props)
  - you can not access in the child component

Note: passing index as key props value is not a good practice.

- index was not fixed it will change when you will add or delete element when index is change
- if you will access the index will render with change data component will Re-render with data modify.

Example:

- App.js

```
// rendering array of object
import Props from "./Profile-1";
let App=()=>{
let std=[{"ID":101,"name":"Sangam Kumar", "age":20, "dept":"CSE", "Roll_No":39,"marks":99,"add":"Sasaram Bihar"},
  {"ID":102,"name":"Satyam Kumar", "age":21 ,"dept":"CSE", "Roll_No":40,"marks":98,"add":"Patna Bihar"},
  {"ID":103,"name":"Sujeet Kumar", "age":22 ,"dept":"EEE", "Roll_No":41,"marks":97,"add":"Sasaram Bihar"},
  {"ID":104,"name":"Shushil Kumar", "age":23 ,"dept":"ME", "Roll_No":42,"marks":96,"add":"Sasaram Bihar"},
  {"ID":105,"name":"Subham Kumar", "age":24 ,"dept":"CSE", "Roll_No":43,"marks":99,"add":"Sasaram Bihar"},
  {"ID":106,"name":"Saurav Kumar", "age":25 ,"dept":"CSE", "Roll_No":44,"marks":99,"add":"Sasaram Bihar"}]
  return(
   <div>
    {/* { std.map((obj, index)=> <Props data={obj} key={index}/>) } */}
    {/* { std.map((obj)=> <Props data={obj} key={obj.Roll_No}/>) }  */}
    { std.map((obj)=> <Props data={obj} key={obj.ID}/>) }
   </div>
  )
}
export default App;
```

- here index and obj.Roll_No is key for idetify the react application unique object

- **Profile-1.js**

```
let Prof=(Props)=>{
   let {name,age,dept,Roll_No,marks,add}=Props.data
   return (
     <div>
       <h1>Name:{name}</h1>
```

```
            <h2>Age:{age}</h2>
            <h3>Department:{dept}</h3>
            <h3>Roll Number:{Roll_No}</h3>
            <h3>Marks:{marks}</h3>
            <h3>Address:{add}</h3>
        </div>
    )

}
export default Prof
```

STATE IN REACT JS

- state is an internal memory of the component,
- To store data in the component state will be used. only component changeable data are store
- the state is local to the component where you define

❖ what is used of state?

- if data is maintained in a variable if you did any modification in data that modification will not modify to User Interface.
- once the data is modified to rerender the dependent component we will maintain data in a state.
- state is a JavaScript object used to store and manage data that can change over time within a component.
- it allows components to keep track of dynamic information and re-render when that data changes.
- State is used to manage the local, internal data of a component, which can be different for each instance of that component.

❖ key points about state in React:

1. **Local Component Data:** State is used to store data that is specific to a particular component and is not shared with other components. Each component instance has its own state.
2. **Mutable Data:** State data can change during the lifetime of a component. When state data changes, React automatically re-renders the component to reflect the new data.
3. **Class Components:** In React, class components are used to work with state. You define and manage state using the this.state object within a class component.
4. **Immutable:** State should be treated as immutable in React. Instead of directly modifying state data, you use the setState() method provided by React to update state. This helps ensure predictable component updates.
5. **Initialization:** You typically initialize state in the component's constructor, and you can also set an initial state when defining a class component.

Example without using state:

```
let App=()=>{
 let c=0
 let inc=()=>{
  c=c+1
  console.log(c)

 }
 return(
  <div>
   <h1>Cout={c}</h1>
   <button onClick={inc}>inc</button>
  </div>
 )
}
export default App
```

- Problem: count will not increase in UI
- So, we are using state

❖ How to create state:

- useState is use to create a state object it will take initial state value as a argument it will return state object and state updation function as a return value

syntax:

➢ let[state obj, updationfun]=usestate(initial value)

- state variable should not update directly always state variable should be updated state updating function only.
- always state will aspect new object per rerendering
- current updated value will available for next rendering

Example:

➢ **App.js**

```
import { useState } from "react";
let App=()=>{
  let [c,setc]=useState(0)
  let inc=()=>{
    setc(c+1)
    console.log(c)
  }
  return(
    <div>
      <h3>count:{c}</h3>
      <button onClick={inc}>inc</button>
    </div>
  )
}
export default App;
```

❖ increment and decrement number application:

Example:

➢ App.js

```
import {useState} from 'react'
let App=()=>{
  let [count,setCount]=useState(0)
  let inc=()=>{
    // setCount((x)=>{
    //   console.log(x)
    //   return x+1
    // })
    setCount(count+1)
  }
  let dec=()=>{
    setCount(count-1)
  }
  return(
    <div>
      <h1>Cout={count}</h1>
      <button onClick={inc}>inc</button>
      <button onClick={dec}>dec</button>
    </div>
  )
}
export default App
```

output:

```
Cout=3
inc dec
```

❖ Batch updation:

```
import {useState} from 'react'
let App=()=>{
  let [c, setCount]=useState(0)
  let update=()=>{
    setCount(c+1)
    setCount(c+1)
    setCount(c+1)
    setCount(c+1)
    setCount(c+1)
        setCount(c+1)
  }
  return(
    <div>
<h3>Count:{c}</h3>
<button onClick={update}>inc</button>
    </div>
  )
}
export default App
```

output:

```
count: 0 here count will 0,1,2 so on so
inc
```

- ➢ in the above example even if you are updating 6 times c value it is considering old value in all the case to react follows batch process when it is updating state

or

- ➢ react state follow batch processing due to that if you are updating more than one time in a single function it will reflect only one time.

❖ what is batch processing?

- ➢ Batch processing maens all the statement will be considere as single group and update execute as a group or at time to avoid this
- to solve the this issue in above update the state by help of call by function then call by function will take previous state as argument updated values as return value

Note:

- ➢ To update the state using call back function as a proper way.

Example:

```
import { useState } from 'react';
let App = () => {
  let [c, setCount] = useState(0);
  let update = () => {
    // Use the functional update form to ensure correct updates
    setCount((prevCount) => prevCount + 1); // call by function
    setCount((prevCount) => prevCount + 1);
    setCount((prevCount) => prevCount + 1);
    setCount((prevCount) => prevCount + 1);
    setCount((prevCount) => prevCount + 1);
    setCount((prevCount) => prevCount + 1);
  };
  return (
    <div>
      <h3>Count: {c}</h3>
      <button onClick={update}>inc</button>
    </div>
  );
};

export default App;
```

output:

```
Count: 0  . here count will 0,6, 12 etc...
 inc
```

❖ when to use call by function?

➢ if current state depended into previous state that time, we are using call by function.

Example:

```
import React, { useState, useEffect } from "react";
let App = () => {
  let [c, setCount] = useState(0);
  let [IId, setIId] = useState(-1);
  useEffect(() => {
    // This effect will run when the component mounts
    // You should clear the interval when the component unmounts
    return () => {
      if (IId !== -1) {
        clearInterval(IId);
      }
    };
  }, [IId]); // Add IId as a dependency to clear the interval when it changes
  let inc = () => {
    setCount((prevCount) => prevCount + 1);
  };
  let start = () => {
    if (IId === -1) {
      // Use a function to set IId based on the previous state
      setIId((prevIId) => {
        const intervalId = setInterval(inc, 1000);
        return intervalId;
      });
    }
  };
  let stop = () => {
    if (IId !== -1) {
      clearInterval(IId);
      setIId(-1); // Reset IId when stopping the interval
    }
  };
  let reset = () => {
    setCount(0); // Reset the count to 0
    if (IId !== -1) {
      clearInterval(IId);
      setIId(-1); // Reset IId when stopping the interval
    }
  };
  return (
    <div>
      <h3>Count: {c}</h3>
      <button onClick={start}>Start</button>
      <button onClick={stop}>Stop</button>
      <button onClick={reset}>Reset</button>
    </div>
  );
};
export default App;
```

output:

```
Count: 0
Start Stop Reset
```

useEffect:

- useEffect is a built-in hook in React that allows you to perform side effects in functional components. Side effects can include data fetching, manually changing the DOM, subscribing to external data sources, and more.

syntax:

```
useEffect(effectFunction, dependencyArray)
useEffect(()=>{},[])
```

- **effectFunction:** This is a required function that contains the code for the side effect you want to perform. It's called after the component renders.
- **dependencyArray (optional):** This is an array of dependencies that determines when the effect should run. It's an optional argument. If you omit it, the effect will run after every render.
- dependencyArray:
- **Empty Array ([]):** The effect runs once after the initial render and doesn't re-run.

Example:

```
useEffect(() => {
  // This effect runs only once after the initial render
  // Add cleanup code if needed
}, []);
```

No Dependency Array (Omitted): The effect runs after every render.

```
useEffect(() => {
  // This effect runs after every render
});
```

With Dependencies: The effect runs whenever any of the specified dependencies change.

```
const [count, setCount] = useState(0);
useEffect(() => {
  // This effect runs whenever 'count' changes
}, [count]);
```

The return value of the useEffect hook is an optional cleanup function.

Example: Digital Watch:

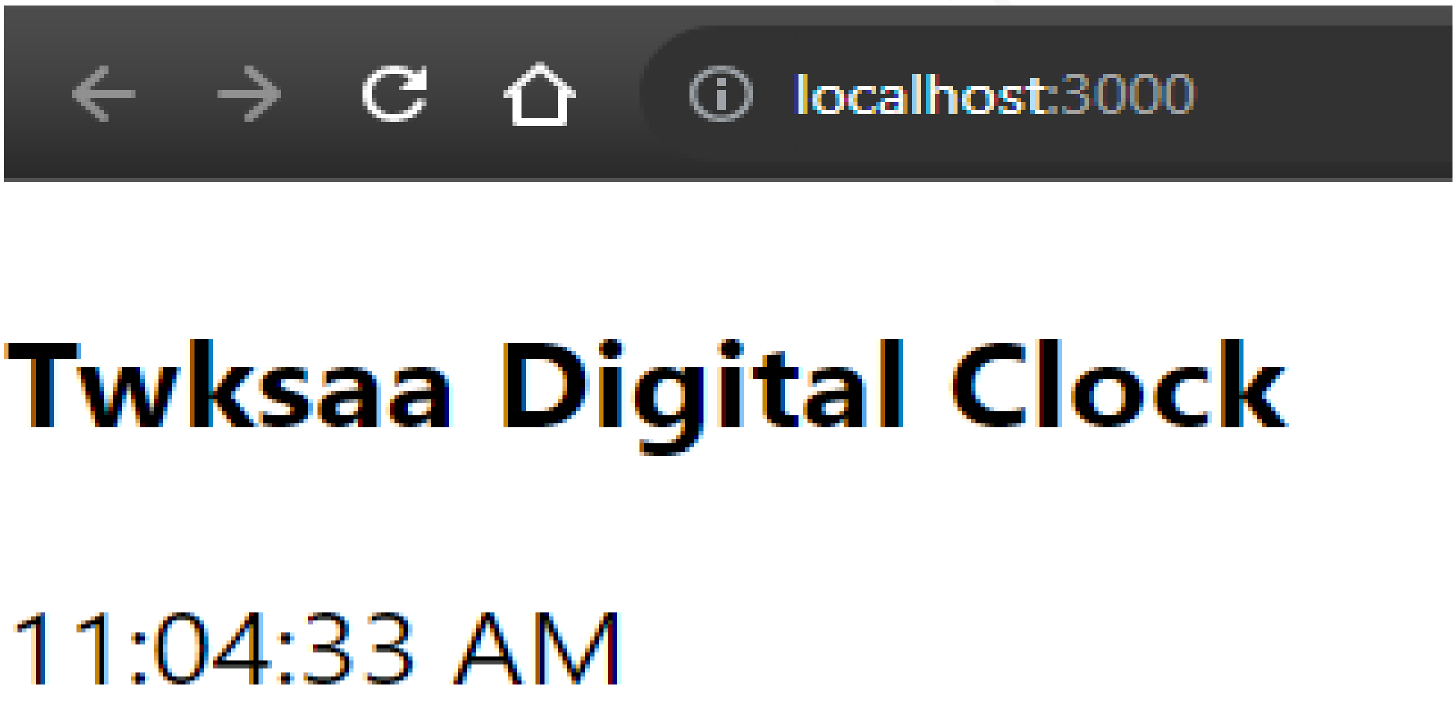
```
useEffect(() => {
  // Effect code
  // Cleanup function
  return () => {
    // Cleanup code
  };
}, [dependencies]);
```

digital clock watch:

Example:

```
import React, { useState, useEffect } from "react";
const DigitalClock = () => {
  const [time, setTime] = useState(new Date());
  useEffect(() => {
    // Create an interval to update the time every second
    const intervalId = setInterval(() => {
      setTime(new Date());
    }, 1000);
    // Return a cleanup function to clear the interval when the component unmounts
    return () => {
      clearInterval(intervalId);
    };
  }, []); // Empty dependency array, so the effect runs only once after the initial render
  // Format the time as HH:MM:SS
  const formattedTime = time.toLocaleTimeString();
  return (
    <div>
      <h2>Digital Clock</h2>
      <div className="clock">{formattedTime}</div>
    </div>
  );
};
export default DigitalClock;
```

output:



Twksaa Digital Clock

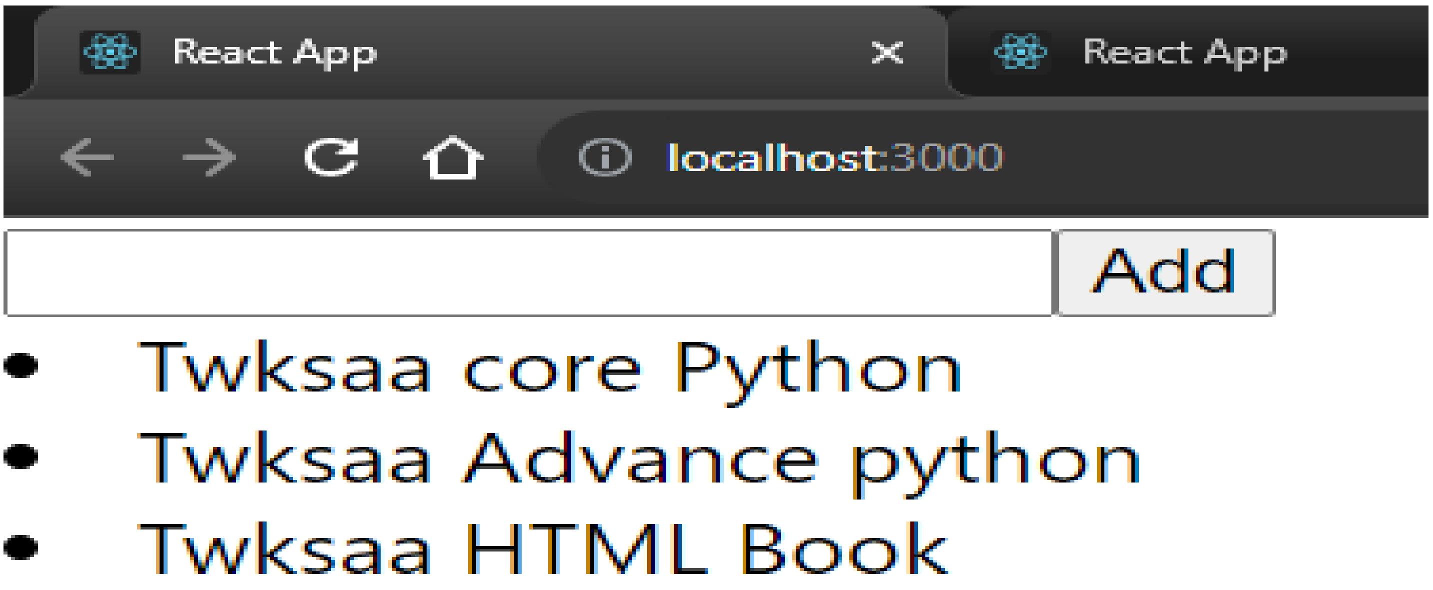
11:04:33 AM

Example:

```
import { useState } from "react"; // Importing the 'useState' hook from React
// Define the 'App' functional component
let App = () => {
  // Initialize state variables using 'useState'
  let [name, setName] = useState(""); // 'name' stores the input field value
  let [data, setData] = useState([]); // 'data' stores the list of names
  // Event handler for input field changes
  let fun = (e) => {
    setName(e.target.value);}     // Update the 'name' state with the current input value
  let add = () => { // Event handler for the 'Add' button click
```

```
        // Add the current 'name' to the 'data' array
        // Using the spread operator to create a new array with the updated data
        setData([...data, name]);
        setName("");} // Reset the 'name' state to an empty string, clearing the input field
      // JSX for rendering the UI
      return (
        <div>
          {/* Input field for entering a name */}
          <input type="text" onChange={fun} value={name} />
          {/* Button to trigger the 'add' function */}
          <button onClick={add}>Add</button>
          <div>
            {/* Render the list of names from the 'data' array */}
            {data.map((item) => <li>{item}</li>)}
          </div>
        </div>
      );}
    export default App;
```

output:



- Twksaa core Python
- Twksaa Advance python
- Twksaa HTML Book

useState

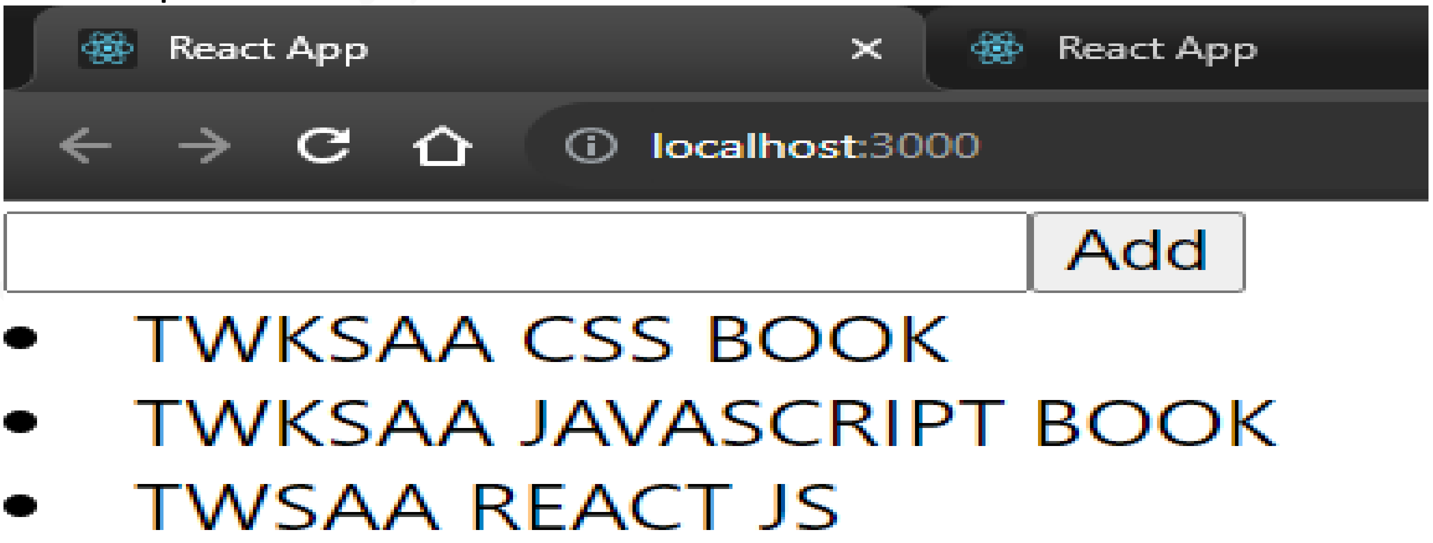
- useRef does not trigger re-renders when the value it references changes.
- To attach reference to the any HTML element we will create reference by helf useRef.
- we will attach reference by helf ref props.

2nd method

Example-2:

```
import React, { useRef, useState } from "react"; // Import necessary React components and hooks
let App = () => { // Define a functional component named 'App'
  let [data, setData] = useState([]); // Declare a state variable 'data' using the 'useState' hook
and Initialize it as an empty array []
  let name = useRef(); // Create a ref called 'name' using the 'useRef' hook
  let add = () => { // Define a function 'add'
    // Update the 'data' state by adding the current value of 'name' ref
    // to the existing 'data' array using the spread operator [...data]
    setData([...data, name.current.value]);
    name.current.value = ""; // Clear the input field by setting its value to an empty string
  };
  return ( // Render the component's UI
    <div>
      <input type="text" ref={name} /> {/* Input field for entering text, with a reference to
'name' */
      <button onClick={add}>Add</button> {/* Button with an 'onClick' event handler to call the 'add'
function */}
      <div>
        {data.map((item, index) => ( {/* Map over the 'data' array and render each item as a list item */}
          <li key={index}>{item}</li>
        ))}
      </div>
    </div>
  );};
export default App;
```

output:



React App

localhost:3000

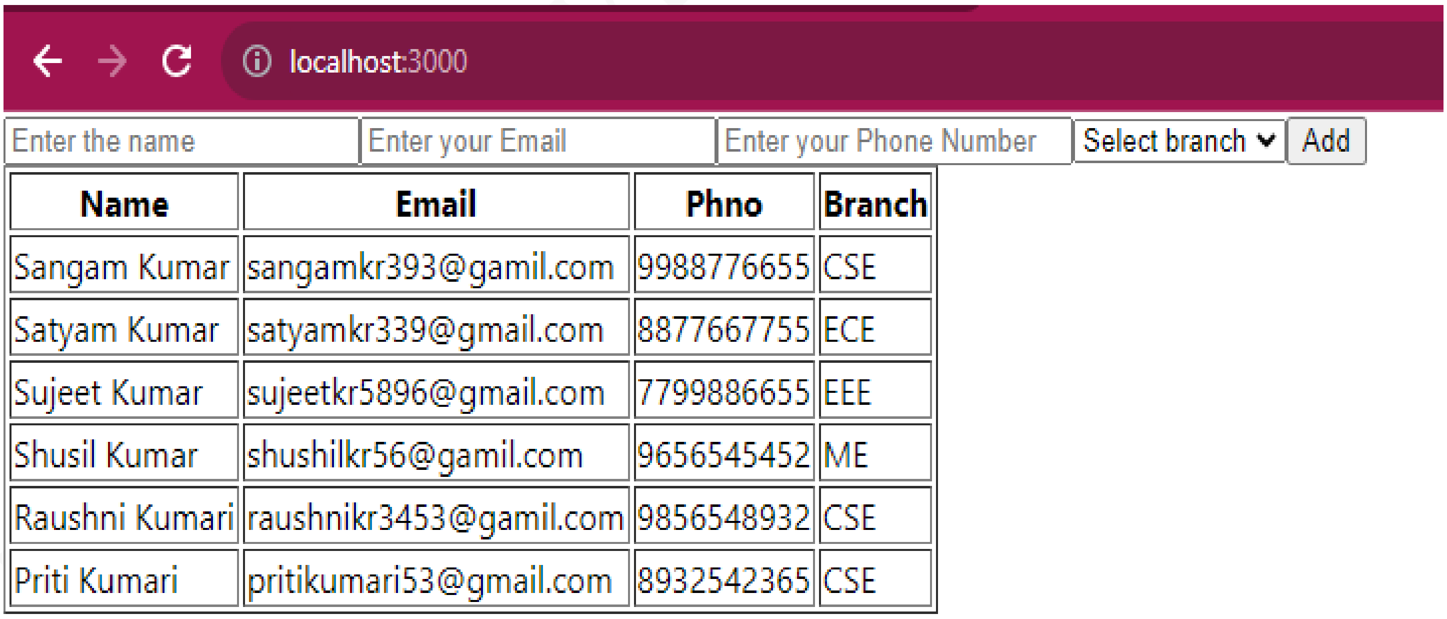
Add

- TWKSAA CSS BOOK
- TWKSAA JAVASCRIPT BOOK
- TWSAA REACT JS

Mini Project-1:

```
import { useState } from "react";
let App = () => {  // Defining a functional component named 'App'.
  // Initializing a state variable 'std' with an object containing name, email, phno, and branch fields.
  // 'setStd' is a function used to update the 'std' state.
  let [std, setStd] = useState({"name":"","email":"","phno":"","branch":""});
  // Initializing a state variable 'data' as an empty array.
  // 'setData' is a function used to update the 'data' state.
  let [data, setData] = useState([]);
  let fun = (e) => { // A function 'fun' that updates the 'std' state based on user input.
    setStd({...std, [e.target.name]: e.target.value});
  }
  // A function 'add' that adds the current 'std' object to the 'data' array and resets 'std' to empty
values.
  let add = () => {
    setData([...data, std]);
    setStd({"name":"","email":"","phno":"","branch":""});
  }
  // The 'return' statement starts the rendering of the component.
  return (
    <div>
      <div>
        {/* Input fields for name, email, phone number, and branch, with 'onChange' event handlers to
call 'fun' and 'value' to display the values from the 'std' state. */}
        <input type="text" placeholder="Enter the name" name="name" onChange={fun}
value={std.name} />
        <input type="text" placeholder="Enter your Email" name="email" onChange={fun}
value={std.email} />
        <input type="text" placeholder="Enter your Phone Number" name="phno" onChange={fun}
value={std.phno} />
        <select onChange={fun} name="branch" value={std.branch}>
          <option disabled selected value="">Select branch</option>
          <option value="CSE">CSE</option>
          <option value="ECE">ECE</option>
          <option value="EEE">EEE</option>
          <option value="ME">ME</option>
          <option value="CE">CE</option>
        </select>
        <button onClick={add}>Add</button>
      </div>
      {/* Conditionally rendering a table if 'data' is not empty. */}
      { data.length !== 0 &&
        <table border={1}>
          <thead>
            <tr>
              <th>Name</th><th>Email</th><th>Phno</th><th>Branch</th>
            </tr>
          </thead>
          <tbody>
           {
             // Mapping over the 'data' array to display its content in rows of the table.
             data.map((std) => {
               return (
                 <tr>
                   <td>{std.name}</td>
                   <td>{std.email}</td>
                   <td>{std.phno}</td>
                   <td>{std.branch}</td>
                 </tr>
               )
             })
           }
          </tbody>
        </table>
      }
    </div>
  )
}

export default App;
```

Output:



localhost:3000

Enter the name | Enter your Email | Enter your Phone Number | Select branch | Add

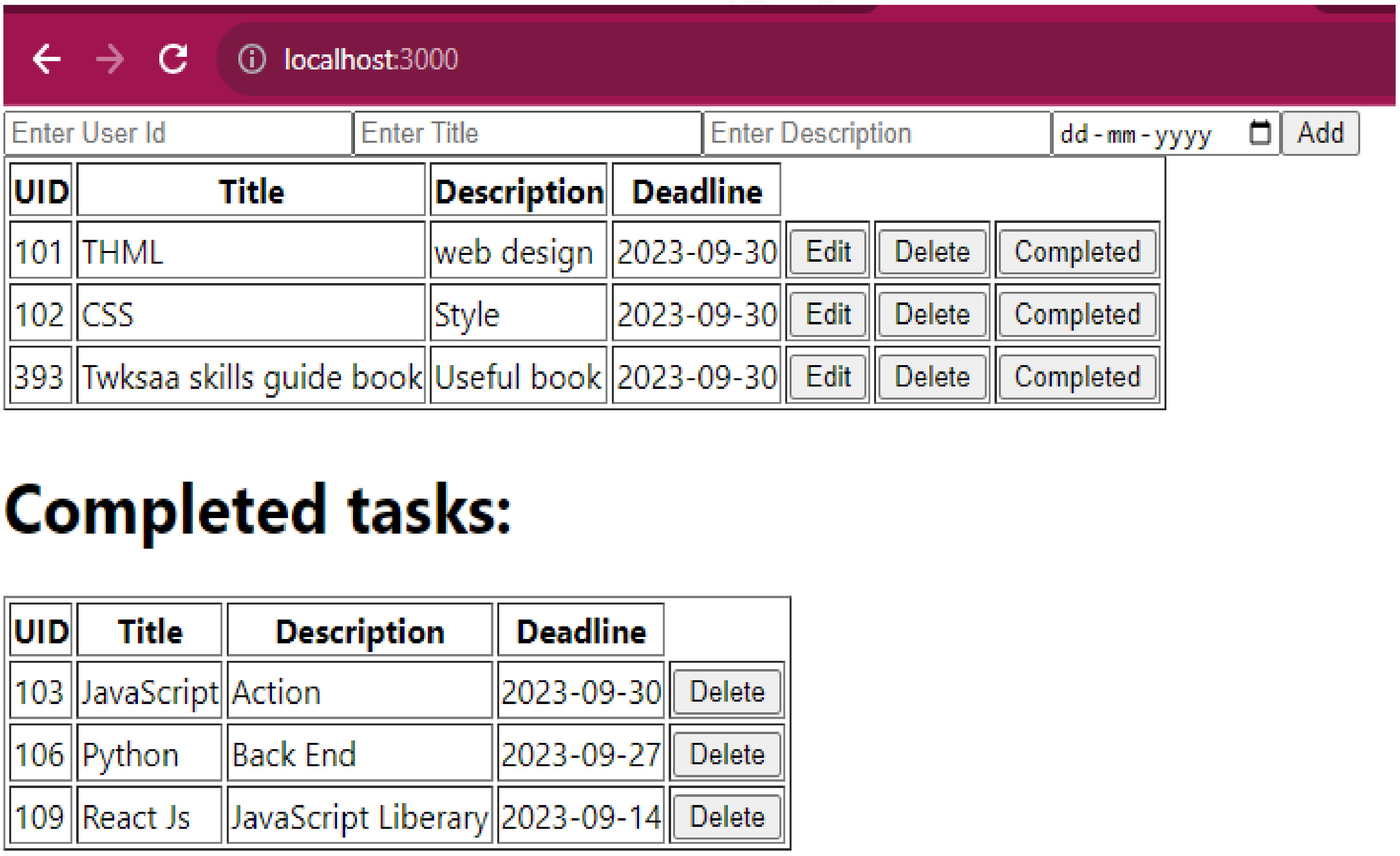
Name	Email	Phno	Branch
Sangam Kumar	sangamkr393@gamil.com	9988776655	CSE
Satyam Kumar	satyamkr339@gmail.com	8877667755	ECE
Sujeet Kumar	sujeetkr5896@gmail.com	7799886655	EEE
Shusil Kumar	shushilkr56@gamil.com	9656545452	ME
Raushni Kumari	raushnikr3453@gamil.com	9856548932	CSE
Priti Kumari	pritikumari53@gmail.com	8932542365	CSE

Mini Project-2 To do Application:

Program:

```
// Import the `useState` function from the 'react' library
import { useState } from 'react';
// Define the functional component named `App`
let App = () => {
  // Initialize state variables using the `useState` hook
  let [data, setData] = useState({ uid: '', title: '', desc: '', dedline: '' });
  let [todo, setTodo] = useState([]);
  let [ind, setInd] = useState();
  let [f, setFlag] = useState(true);
  let [comp, setComp] = useState([]);
  // Event handler function for input fields
  let fun = (e) => {
    // Update the `data` state by spreading the previous state and updating the changed field
    setData({ ...data, [e.target.name]: e.target.value });
    console.log(data); // Log the current state (it may not reflect the update immediately due
to the asynchronous nature of `useState`)
  };// Event handler function for adding a task
  let add = () => {
    // Add the current `data` state to the `todo` array and clear the input fields
    setTodo([...todo, data]);
    setData({ uid: '', title: '', desc: '', dedline: '' });
  };// Event handler function for deleting a task from the `todo` array
  let del = (index) => {
    // Remove the task at the specified index and update the `todo` state
    todo.splice(index, 1);
    setTodo([...todo]);
  };// Event handler function for moving a task to the completed tasks list
  let com = (index) => {
    // Add the task at the specified index in `todo` to the `comp` array and delete it from `todo`
    setComp([...comp, todo[index]]);
    del(index);
  };// Event handler function for editing a task
  let edit = (index) => {
    // Set the `data` state to the task at the specified index in `todo`, set the index, and switch to edit mode
    setData(todo[index]);
    setInd(index);
    setFlag(false);
  };// Event handler function for updating an edited task
  let upd = () => {
    // Replace the task at the specified index in `todo` with the updated `data`, and reset the state for editing
    todo[ind] = data;
    setTodo([...todo]);
    setFlag(true);
    setData({ uid: '', title: '', desc: '', dedline: '' });
  };// Return the JSX (user interface) for the component
  return (
    <div>
      {/* Input fields for task details */}
      <input type="text" placeholder='Enter User Id' name="uid" value={data.uid}
onChange={fun} />
      <input type="text" placeholder='Enter Title' name="title" value={data.title}
onChange={fun} />
      <input type="text" placeholder='Enter Description' name="desc" value={data.desc}
onChange={fun} />
      <input type="date" name="dedline" value={data.dedline} onChange={fun} />
      {/* Conditional rendering of buttons based on whether the component is in add or edit mode */}
      {f && <button onClick={add}>Add</button>}
      {!f && <button onClick={upd}>Update</button>}
      {/* Display the to-do list in a table */}
      <div>
        <table border={1}>
          <tr>
            <th>UID</th>
            <th>Title</th>
            <th>Description</th>
            <th>Deadline</th>
          </tr>
          {/* Map through the `todo` array to display tasks */}
          {todo.map((item, index) => {
            return (
              <tr>
                <td>{item.uid}</td>
                <td>{item.title}</td>
                <td>{item.desc}</td>
                <td>{item.dedline}</td>
                <td><button onClick={() => edit(index)}>Edit</button></td>
                <td><button onClick={() => del(index)}>Delete</button></td>
                <td><button onClick={() => com(index)}>Completed</button></td>
              </tr>
            );
          })}
        </table>
      </div>
      {/* Display the completed tasks list in a table */}
      <div>
        <h1>Completed tasks:</h1>
        <table border={1}>
          <tr>
            <th>UID</th>
            <th>Title</th>
            <th>Description</th>
            <th>Deadline</th>
          </tr>
          {/* Map through the `comp` array to display completed tasks */}
          {comp.map((item, index) => {
            return (
              <tr>
                <td>{item.uid}</td>
                <td>{item.title}</td>
                <td>{item.desc}</td>
                <td>{item.dedline}</td>
                <td><button onClick={() => com(index)}>Delete</button></td>
              </tr>
            );
          })}
        </table>
      </div>
    </div>
  );
};
export default App;
```

Result:



localhost:3000

Enter User Id | Enter Title | Enter Description | dd-mm-yyyy | Add

UID	Title	Description	Deadline			
101	THML	web design	2023-09-30	Edit	Delete	Completed
102	CSS	Style	2023-09-30	Edit	Delete	Completed
393	Twksaa skills guide book	Useful book	2023-09-30	Edit	Delete	Completed

Completed tasks:

UID	Title	Description	Deadline	
103	JavaScript	Action	2023-09-30	Delete
106	Python	Back End	2023-09-27	Delete
109	React Js	JavaScript Liberary	2023-09-14	Delete

❖ passing state as a prop for child component:

1. Define the Parent Component:

- First, you'll need a parent component that holds the state you want to pass to the child component. The state can be defined using the `useState` hook or within a class component's state.

Example using a functional component and the `useState` hook:

```
import React, { useState } from 'react';
function ParentComponent() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <ChildComponent count={count} />
    </div>
  );
}
```

2. Create the Child Component:

- Next, create the child component that will receive the state as a prop. In this example, `count` is passed as a prop to `ChildComponent`.

```
import React from 'react';
function ChildComponent(props) {
  return (
    <div>
      <p>Count: {props.count}</p>
    </div>
  );
}
```

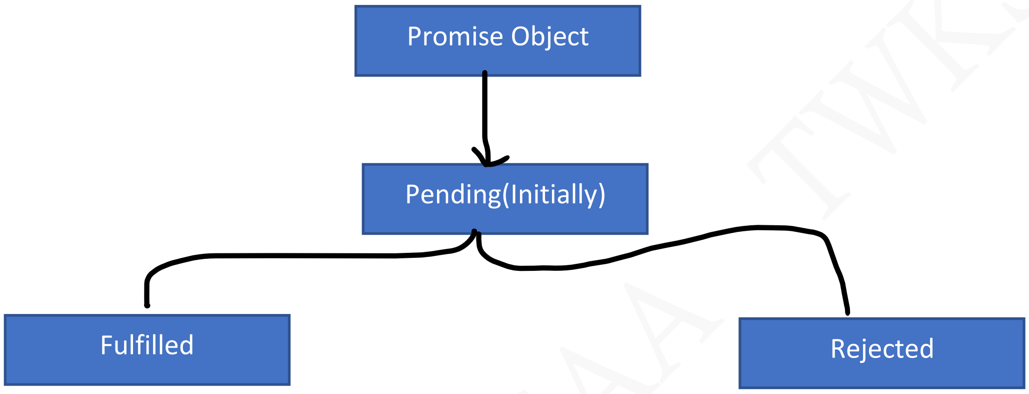
3. Rendering the Parent Component:

- To make this work, you need to render the `ParentComponent` in your main application file (e.g., `index.js` or `App.js`).

```
import React from 'react';
import ReactDOM from 'react-dom';
import ParentComponent from './ParentComponent';
ReactDOM.render(
  <React.StrictMode>
    <ParentComponent />
  </React.StrictMode>,
  document.getElementById('root')
);
```

❖ State lifting:

- If you want to maintain a common state for the group of components, we will maintain state in its parent component it is call state lifting.
- To implement lifecycle method in functional components useEffect will use it will take two arguments:
    1. Call back function(): which it need to execute
    2. Dependency arrayk
- If the dependency array is empty useEffect will create only one. At the time of component initial rendering
- In dependency array if you provide some variables when that variable are updated every time use Effect will execute



Syntax:
Promiseobj.then((res)=>{ }).catch((err)={})
Note: to mange parent component state through its child component, we will pass state updation as props

❖ state lifting in React:

1. State in React Components:

- In React, components can have their own state, which is a JavaScript object that holds data specific to that component. State allows components to manage and update their data independently.

2. Parent and Child Components:

- In a typical React application, you often have a hierarchy of components, with some components being children of others. Data may need to be shared between these parent and child components.

3. When to Use State Lifting:

- State lifting becomes necessary when you have a piece of state in a child component that needs to be accessed or modified by a parent or other sibling components. Instead of managing this state solely within the child component, you move it up to a common ancestor component (typically a parent) to make it accessible to other parts of your application.

4. Lifting State Up:

a. Identify the state that needs to be shared or modified by multiple components.
b. Move this state up to the nearest common ancestor component that needs access to it.

c. This parent component becomes the "source of truth" for the shared state.

5. Passing State as Props:

- After lifting the state, you pass it down to the child component(s) that need access to it as props. The child component(s) can then read and possibly modify this state by invoking functions provided as props by the parent.

6. Updating State:

- When a child component needs to update the state, it should call a function passed down from the parent component as a prop. This function, defined in the parent component, will update the state in the parent, which will then trigger a re-render of both the parent and any affected child components.

7. Benefits of State Lifting:

- Centralized State: State lifting helps in centralizing the management of shared state, making it easier to understand and maintain.
- Data Flow Control: It provides better control over the flow of data between components.
- Reusable Components: By lifting state, you can create more reusable components because the state is managed by a parent, and child components can focus on rendering and behavior

Example:

➢ **app.js npm install axios**

```
import axios from "axios"
import { useEffect, useState } from "react"
import Row from "./Row"
let App=()=>{
  let [data,setData]=useState([])
  useEffect(()=>{
    axios.get("https://jsonplaceholder.typicode.com/posts").then((res)=>{
      setData(res.data)
    })
  },[])
  let del=(ind)=>{
    data.splice(ind,1)
    setData([...data])
  }
  return(
    <div>
      <table border={1}>
        <tr><th>uid</th><th>Id</th><th>title</th><th>body</th></tr>
        {
          data.map((item,index)=><Row data={item} del={del} i={index}/>)
        }
      </table>

    </div>
  )
}
export default App
```

Row.js:

```
let Row=(props)=>{
   let data=props.data
   return(
      <tr>
         <td>{data.userId}</td>
         <td>{data.id}</td>
         <td>{data.title}</td>
         <td>{data.body}</td>
         <td><button onClick={()=>props.del(props.i)}>Delete</button></td>
      </tr>
   )
}
export default Row
```

Data access through Api in React js:

Example:

➢ **App.js**

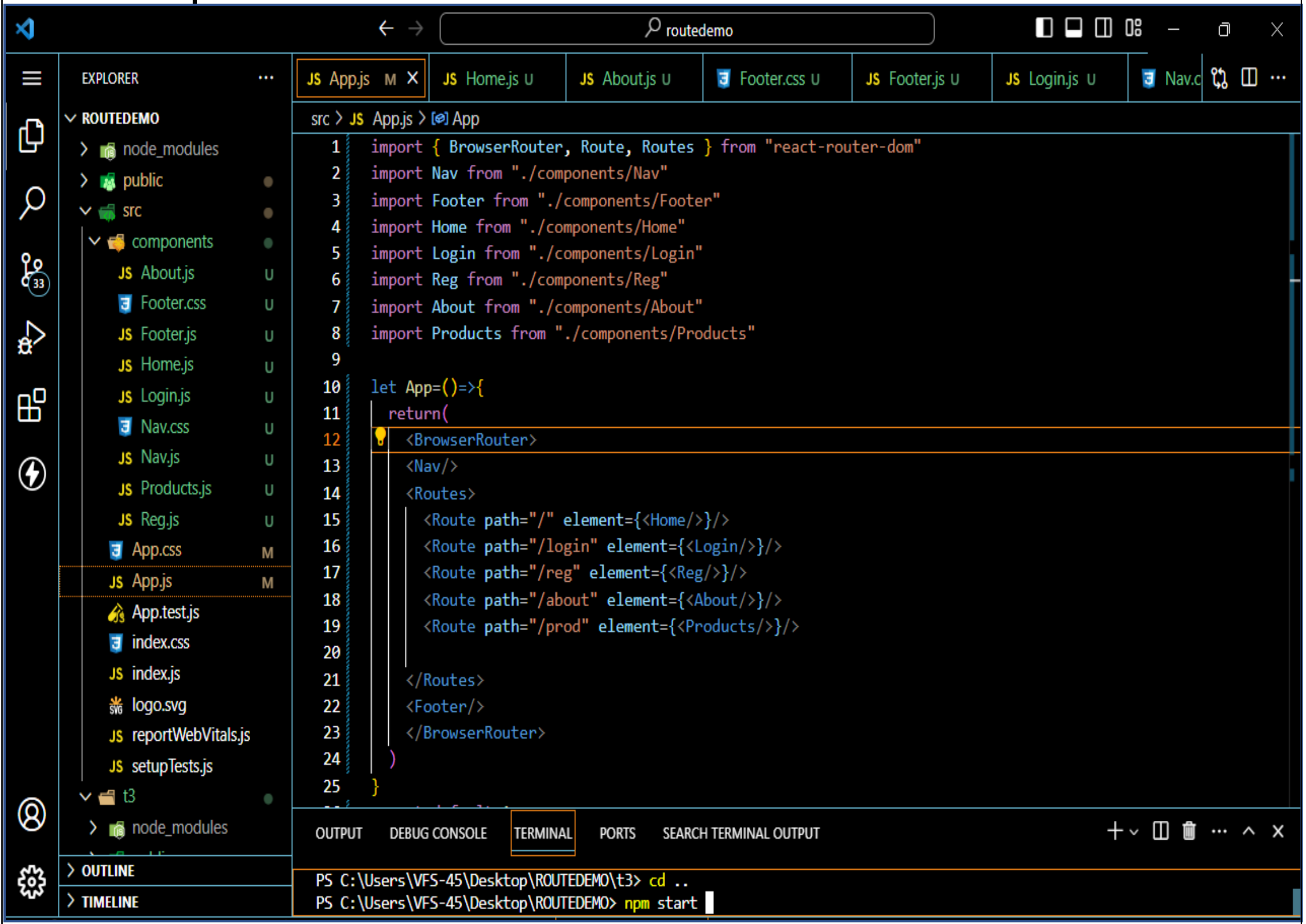
```
import axios from "axios"
// npm i axios    install axios
import { useEffect, useState } from "react"
import Row from "./Row"
let App=()=>{
  let [data,setData]=useState([])
  useEffect(()=>{
    axios.get("https://jsonplaceholder.typicode.com/posts").then((res)=>{
      setData(res.data)
    })
  },[])
  let del=(ind)=>{
    data.splice(ind,1)
    setData([...data])
  }
  return(
    <div>
      <table border={1}>
        <tr><th>uid</th><th>Id</th><th>title</th><th>body</th></tr>
        {
          data.map((item,index)=><Row data={item} del={del} i={index}/>)
        }
      </table>

    </div>
  )
}
export default App
```

➢ **Row.js**

```
let Row=(props)=>{
   let data=props.data
   return(
      <tr>
         <td>{data.userId}</td>
         <td>{data.id}</td>
         <td>{data.title}</td>
         <td>{data.body}</td>
         <td><button onClick={()=>props.del(props.i)}>Delete</button></td>
      </tr>
   )}
export default Row
```

Route in React:

Example-1:



```
import { BrowserRouter, Route, Routes } from "react-router-dom"
import Nav from "./components/Nav"
import Footer from "./components/Footer"
import Home from "./components/Home"
import Login from "./components/Login"
import Reg from "./components/Reg"
import About from "./components/About"
import Products from "./components/Products"

let App=()=>{
  return(
    <BrowserRouter>
    <Nav/>
    <Routes>
      <Route path="/" element={<Home/>}/>
      <Route path="/login" element={<Login/>}/>
      <Route path="/reg" element={<Reg/>}/>
      <Route path="/about" element={<About/>}/>
      <Route path="/prod" element={<Products/>}/>

    </Routes>
    <Footer/>
    </BrowserRouter>
  )
}
```

```
PS C:\Users\VFS-45\Desktop\ROUTEDEMO\t3> cd ..
PS C:\Users\VFS-45\Desktop\ROUTEDEMO> npm start
```

Note add CDN link in index.css `font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New', monospace;`

➢ **App.js:**

```
import { BrowserRouter, Route, Routes } from "react-router-dom"
import Nav from "./components/Nav"
import Footer from "./components/Footer"
import Home from "./components/Home"
import Login from "./components/Login"
import Reg from "./components/Reg"
import About from "./components/About"
import Products from "./components/Products"
let App=()=>{
 return(
  <BrowserRouter>
  <Nav/>
  <Routes>
```

```
      <Route path="/" element={<Home/>}/>
      <Route path="/login" element={<Login/>}/>
      <Route path="/reg" element={<Reg/>}/>
      <Route path="/about" element={<About/>}/>
      <Route path="/prod" element={<Products/>}/>
     </Routes>
     <Footer/>
     </BrowserRouter>
   )
  }
  export default App
```

- **Footer.js**

```
import './Footer.css'
let Footer=()=>{
  return(
    <div className="ft">@copyright reserved for twksaa.org</div>
  )
}
export default Footer
```

- **Footer.css:**

```
.ft{
  width: 100%;
  line-height: 30px;
  font-size: 20px;
  background-color: blue;
  color: white;
  text-align: center;
  font-style: italic;
}
```

- **About.js:**

```
let About=()=>{
  return(<div>About page</div>)
}
export default About
```

- **Home.js:**

```
let Home=()=>{
  return(
    <div>Home page</div>
  )
}
export default Home
```

- **Login.js:**

```
let Login=()=>{
  return(
    <div>Login page</div>
    )}
export default Login
```

- **Nav.css:**

```
nav{
    width: 100%;
    height: 15vh;
    background-color: aqua;
    display: flex;
    justify-content: space-evenly;
    align-items: center;
}
Nav a{
    text-decoration: none;
    font-size: 20px;
    color: chocolate;
}
```

- **Nav.js:**

```
import { Link } from "react-router-dom"
import "./Nav.css"
let Nav=()=>{
    return(
        <nav>
            <Link to="/">Home <i class="fa-solid fa-house"></i></Link>
            <Link to="/login">Login</Link>
            <Link to="/about">About</Link>
            <Link to="/reg">Register</Link>
            <Link to="/prod">Products</Link>
        </nav>
    )}
export default Nav
```

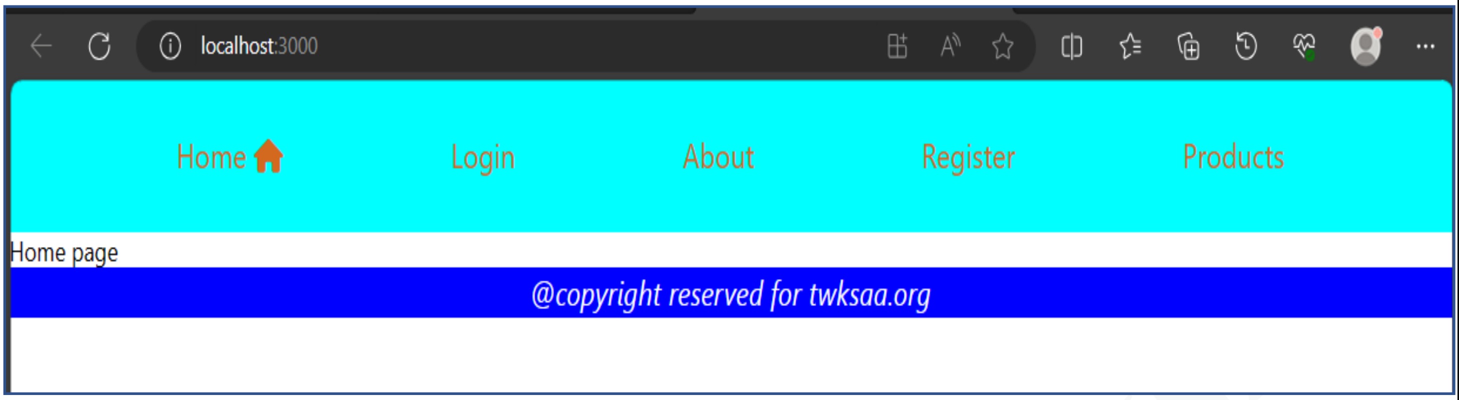
- **Product.js:**

```
let Products=()=>{
    return(
        <div>Products page</div>
    )
}
export default Products
```

- **Reg.js**

```
let Reg=()=>{
    return(
        <div>Reg page</div>
    )
}
export default Reg
```

Output:


localhost:3000

Home Login About Register Products

Home page

@copyright reserved for twksaa.org

Example-2:

Note: Install react-router-dom: for Example > npm i react-router-dom

Install axios: for Example > npm i axios

Install create-react-app: npm install -g create-react-app

Create you react app now: create-react-app appname

➢ **App.js:**

```
import { BrowserRouter, Route, Routes } from "react-router-dom"
import Nav from "./components/Nav"
import Footer from "./components/Footer"
import Home from "./components/Home"
import Login from "./components/Login"
import Reg from "./components/Reg"
import Product from "./components/Product"
let App=()=>{
  return(
    <BrowserRouter>
    <Nav/>
    <Routes>
      <Route path="/" element={<Home/>}/>
      <Route path="/login" element={<Login/>}/>
      <Route path="/reg" element={<Reg/>}/>
      <Route path="/prod" element={<Product/>}/>
    </Routes>
    <Footer/>
    </BrowserRouter>
  )}
export default App
```

➢ **Footer.css**

```
.ft{
    width: 100%;
    line-height: 30px;
    font-size: 20px;
    background-color: rgb(32, 32, 32);
    color: white;
    text-align: center;
    font-style: italic;
}
```

➢ **Footer.js:**

```
import './Footer.css'
let Footer=()=>{
    return(
        <footer className="ft">
            @copyright reserved to DKSRA.COM
        </footer>
    )}
export default Footer
```

➢ **Home.css**

```
.baner{
    width: 100%;
    height: 80vh;
    position: relative;
}
.baner img{
    width: 100%;
    height: 100%;
}
.baner h1{
    position: absolute;
    top:20px;
    left: 100px;
    color: red;
}
.coursecard{
    display: flex;
    width: 80%;margin-left: 10%;
    padding: 30px 0px;
    background-color: bisque;
    flex-wrap: wrap;
    justify-content: space-evenly;
    gap:10px;color: white;
    font-weight: bold;
    text-align: center;
}
.coursecard div{
    width: 35%;
    height: 200px;
    background-color: blue;
}
.coursecard h1{width: 100%;
    text-align: center;}
```

➢ Home.js

```
import { useEffect, useState } from "react"
import './Home.css'
let Home=()=>{
   let arr=["https://images.pexels.com/photos/443446/pexels-photo-
443446.jpeg?cs=srgb&dl=daylight-forest-glossy-443446.jpg&fm=jpg",
"https://th.bing.com/th/id/R.be5169fe358cddf9251f1007b97d4cc8?rik=%2bCN4%2bGOBkw
1ESw&riu=http%3a%2f%2fwww.pixelstalk.net%2fwp-
content%2fuploads%2f2016%2f07%2fPeaceful-Desktop-
Backgrounds.jpeg&ehk=23WIQbDrBOqACC%2bGzzRj9h%2fhZ4iGgWR0kDGAEM%2bHGjk%3
d&risl=&pid=ImgRaw&r=0",
"https://th.bing.com/th/id/R.9b81663a3934071f8de09681cec0684e?rik=DEV7m%2bUbPpm
L2A&riu=http%3a%2f%2fwww.pixelstalk.net%2fwp-
content%2fuploads%2f2016%2f08%2fCool-Nature-Background-Images-
4523x2590.jpg&ehk=XxAKNdbLqeV84jakVaOYrmZWai83%2bRlxqGS7WF2x2%2bo%3d&risl=
1&pid=ImgRaw&r=0",
"https://th.bing.com/th/id/R.057b31be70cc73b9d0c829205f90e02f?rik=PxrTE7PkNVZoRg&r
iu=http%3a%2f%2fwww.pixelstalk.net%2fwp-content%2fuploads%2f2016%2f08%2fCool-
nature-backgrounds-hd-resolution-
1920x1080.jpg&ehk=hQzPW7PZMRmNSlTgkm1V0lR4FopZ7yGKJtUMRedf8wM%3d&risl=&pi
d=ImgRaw&r=0",
"https://wallpapercave.com/wp/wp2555019.jpg",
"https://www.pixelstalk.net/wp-content/uploads/2016/06/Free-Images-Wallpaper-HD-
Background.jpg"]
let txt=["Sangam Kumar","Shusil Kumar","Satyam Kumar","Sujeet Kumar","Raushni
Kumari","T6"]
let [i,setI]=useState(0)
let fun=()=>{
      setI((i)=>(i+1)%6)
}
useEffect(()=>{
   let iid=setInterval(fun,2000)
   return ()=>clearInterval(iid)
},[])
   return(
   <>
      <div className="baner">
         <img src={arr[i]}/>
         <h1>{txt[i]}</h1>
      </div>
      <div className="coursecard">
         <h1>Courses we offer</h1>
         <div>Python Full Stack</div>
         <div>HTML</div>
         <div>CSS</div>
         <div>JAVA SRCIPT</div>
         <div>REACT JS</div>
```

```
            <div>Java Full stack</div>
            <div>.Net Full Stack</div>
        </div> </> )
    }
    export default Home
```

- **Login.js:**

```
    let Login=()=>{
        return(
            <div>
                Login page
            </div>
        )}
    export default Login
```

- **Nav.css:**

```
    nav{
        width: 100%;height: 15vh;
        background-color: blue;
        display: flex;
        justify-content: space-evenly;align-items: center;}
    nav a{
        text-decoration: none;font-size: 20px;
        color: white;
        font-weight: bold;}
```

- **Nav.js:**

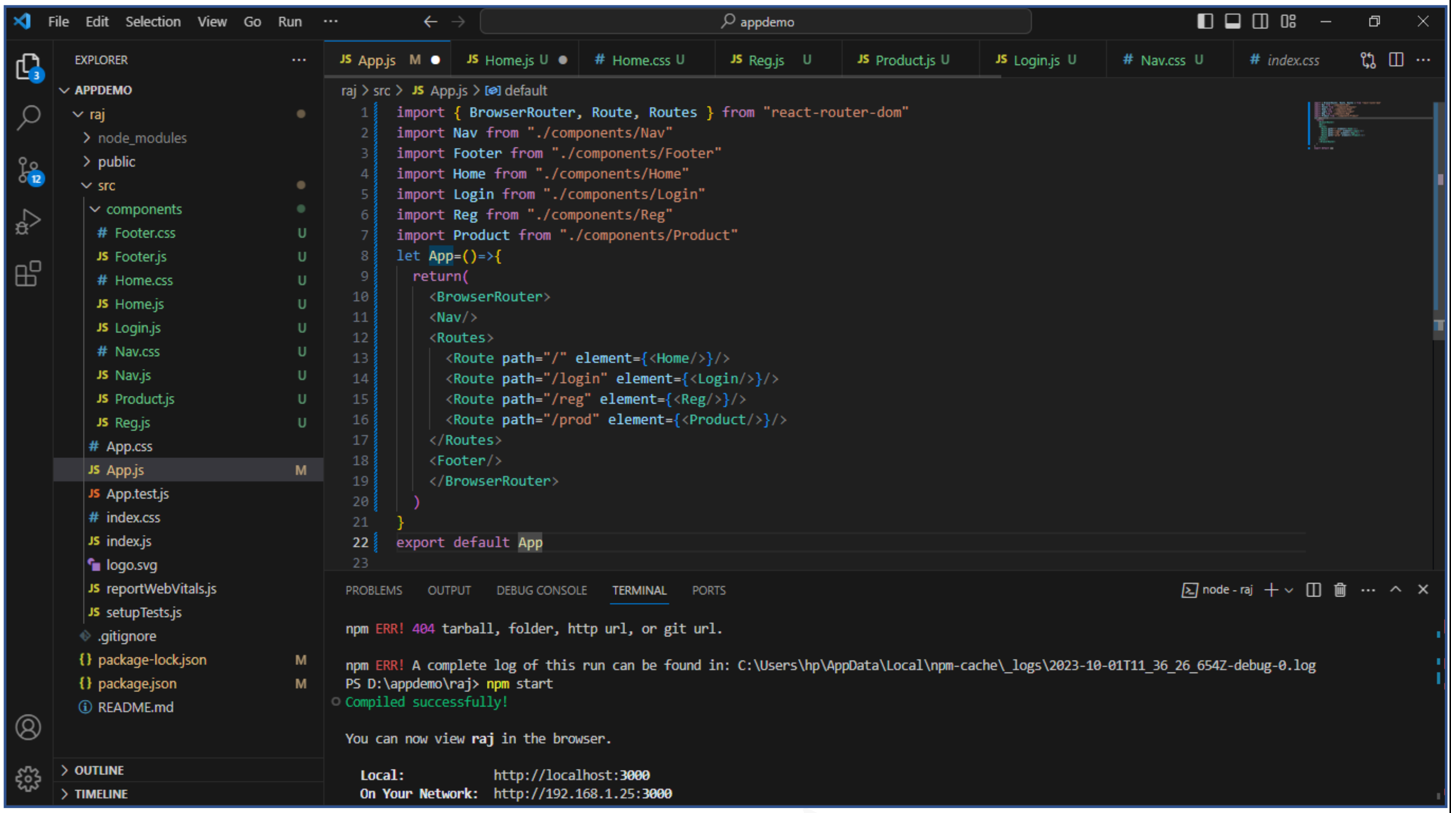
```
    import { Link } from "react-router-dom"
    import "./Nav.css"
    let Nav=()=>{
        return(
            <nav>
              <Link to="/">Home</Link> <Link to="/login">Login</Link>
              <Link to="/reg">Register</Link>
              <Link to="/prod">Products</Link>
            </nav>
        )}
    export default Nav
```

- **Product. js:**

```
    let Product=()=>{
        return(
            <div>products page</div>
        )}
    export default Product
```

- **Reg.js**

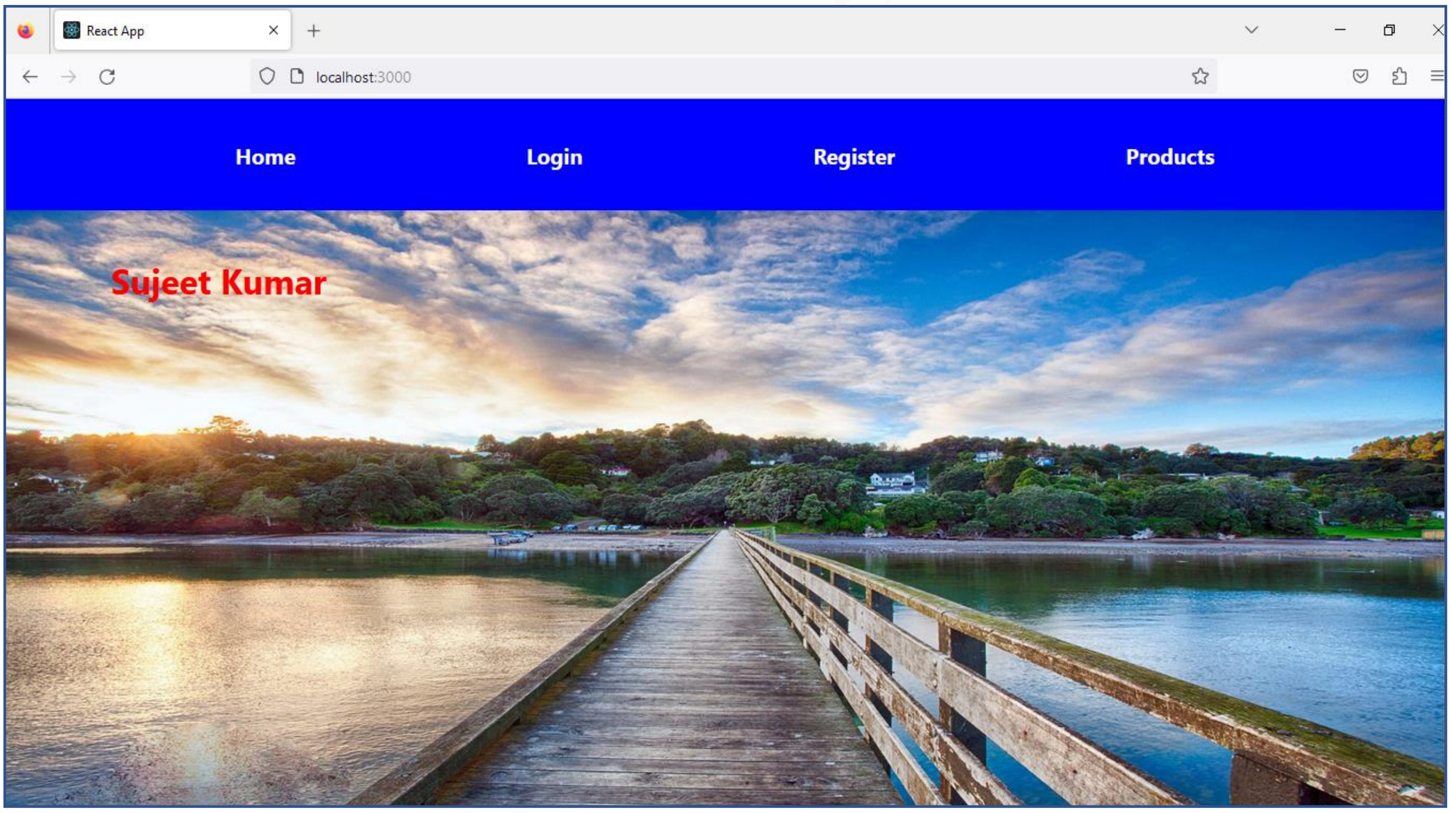
```
    let Reg=()=>{
        return(<div>Reg page</div>)
    }
    export default Reg
```



```
import { BrowserRouter, Route, Routes } from "react-router-dom"
import Nav from "./components/Nav"
import Footer from "./components/Footer"
import Home from "./components/Home"
import Login from "./components/Login"
import Reg from "./components/Reg"
import Product from "./components/Product"
let App=()=>{
  return(
    <BrowserRouter>
    <Nav/>
    <Routes>
      <Route path="/" element={<Home/>}/>
      <Route path="/login" element={<Login/>}/>
      <Route path="/reg" element={<Reg/>}/>
      <Route path="/prod" element={<Product/>}/>
    </Routes>
    <Footer/>
    </BrowserRouter>
  )
}
export default App
```

```
npm ERR! 404 tarball, folder, http url, or git url.

npm ERR! A complete log of this run can be found in: C:\Users\hp\AppData\Local\npm-cache\_logs\2023-10-01T11_36_26_654Z-debug-0.log
PS D:\appdemo\raj> npm start
Compiled successfully!

You can now view raj in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.25:3000
```

Output:



Mini Project on Ecommerce (product add in cart)

❖ **Install axios: npm install axios**

➢ **App.js:**

```
import { BrowserRouter,Route,Routes } from "react-router-dom"
import Home from "./Home"
import Cart from "./Cart"
import Nav from "./Nav"
import './App.css'
import { useState } from "react"
import Gc from "./Gc"
let App=()=>{
 let [cart,setCart]=useState([])
 let [ctotal,setCtotal]=useState(0)
 let addcart=(item)=>{
  let res=cart.filter((prd)=>prd.id==item.id)
  if(res.length==0)
  {
  item={...item,"qty":1,"total":item.price}
  setCart([...cart,item])
  setCtotal(ctotal+item.price)
  }
  else{
   inc(item.id)
  }
 }
 let inc=(id)=>{
  setCart(cart.map((item)=>{
   if(item.id==id)
   {
    setCtotal(ctotal+item.price)
    return {...item,"qty":item.qty+1,"total":item.total+item.price}
   }
   else{
    return item
   }
  }))
 }
 let dec=(id)=>{
  setCart(cart.map((item)=>{
   if(item.id==id && item.qty>1)
   {
    setCtotal(ctotal-item.price)
    return {...item,"qty":item.qty-1,"total":item.total-item.price}
   }
   else{
    return item
```

```
      }
    }))
  }
  let del=(ind)=>{
    setCtotal(ctotal-cart[ind].total)
    cart.splice(ind,1)
    setCart([...cart])
  }
  let obj={"cart":cart,"addcart":addcart,"inc":inc,
  "dec":dec,"del":del,"ctotal":ctotal}
  return(
    <BrowserRouter>
    <Gc.Provider value={obj}>
    <Nav/>
    <Routes>
      <Route path="/" element={<Home/>}/>
      <Route path="/cart" element={<Cart/>}/>
    </Routes>
    </Gc.Provider>
    </BrowserRouter>
  )
}
export default App
```

- **App.css**

```
*{
  margin: 0px;
  padding: 0px;
  outline: 0px;
}
```

- **Cart.js:**

```
import { useContext } from "react"
import Gc from "./Gc"
import './Cart.css'
let Cart=()=>{
    let obj=useContext(Gc)
    return(
      <div>
        <div className="con">
        {
            obj.cart.map((item,index)=>{
                return(
                  <div className="card">
                    <div className="img">
                      <img src={item.images[0]}/>
                      </div>
                      <h2>Name:{item.title}</h2>
                      <h2>Desc:{item.description}</h2>
```

```
                        <h2>Price:{item.price}</h2>
                        <h2>Brand:{item.brand}</h2>
                        <h2>Cat:{item.category}</h2>
                      <h2><button onClick={()=>obj.dec(item.id)}>-</button>
                        {item.qty}
                      <button onClick={()=>obj.inc(item.id)}>+</button></h2>
                        <h2>Total:{item.total}</h2>

                      <button onClick={()=>obj.del(index)}>delcart</button>
                    </div> )
                })}
            </div>
           {obj.cart.length>0&& <div>GT:{obj.ctotal}</div>}
            </div> )}
export default Cart
```

- **Cart.css:**

```
.con{
   width: 100%;
   display: flex;
   justify-content: space-evenly;
   flex-wrap: wrap;
   gap:30px;
   padding: 30px 0px;
   background-color: aliceblue;
}
.card{
   border: 2px solid gray;
   width: 40%;
   height: 250px;
   display: flex;
   flex-direction: column;
   justify-content: space-evenly;
   padding: 10px;
}
.img{
   width: 100px;
   height: 100px;
}
.img img{
   width: 100%;
   height: 100%;
}
.card h2{
   font-size: 12px;
   color:darkcyan
}
```

- **Home.js:**

```
import axios from "axios"
import { useContext, useEffect, useState } from "react"
import './Home.css'
import Gc from "./Gc"
let Home=()=>{
  let [prod,setProd]=useState([])
  let obj=useContext(Gc)
  useEffect(()=>{
    axios.get("https://dummyjson.com/products").then((res)=>{
      setProd(res.data.products)
    })
  },[])
  return(<div>
    <h1>Products:</h1>
    <div className="con">
      {
        prod.map((item)=>{
          return(
            <div className="card">
              <div className="img">
                <img src={item.images[0]}/>
                </div>
                <h2>Name:{item.title}</h2>
                <h2>Desc:{item.description}</h2>
                <h2>Price:{item.price}</h2>
                <h2>Brand:{item.brand}</h2>
                <h2>Cat:{item.category}</h2>
                <button onClick={()=>obj.addcart(item)}>Addcart</button>
            </div>)
        })}
    </div>
  </div>)}
export default Home
```

- **Home.css**

```
.con{
  width: 100%;
  display: flex;
  justify-content: space-evenly;
  flex-wrap: wrap;
  gap:30px;
  padding: 30px 0px;
  background-color: aliceblue;}
.card{
  border: 2px solid gray;
  width: 40%;
  height: 250px;
  display: flex;
  flex-direction: column;
  justify-content: space-evenly;
  padding: 10px;
}
.img{
  width: 100px;
  height: 100px;
}
.img img{
  width: 100%;
  height: 100%;
}
.card h2{
  font-size: 12px;
  color:darkcyan
}
```

➢ **Gc.js**

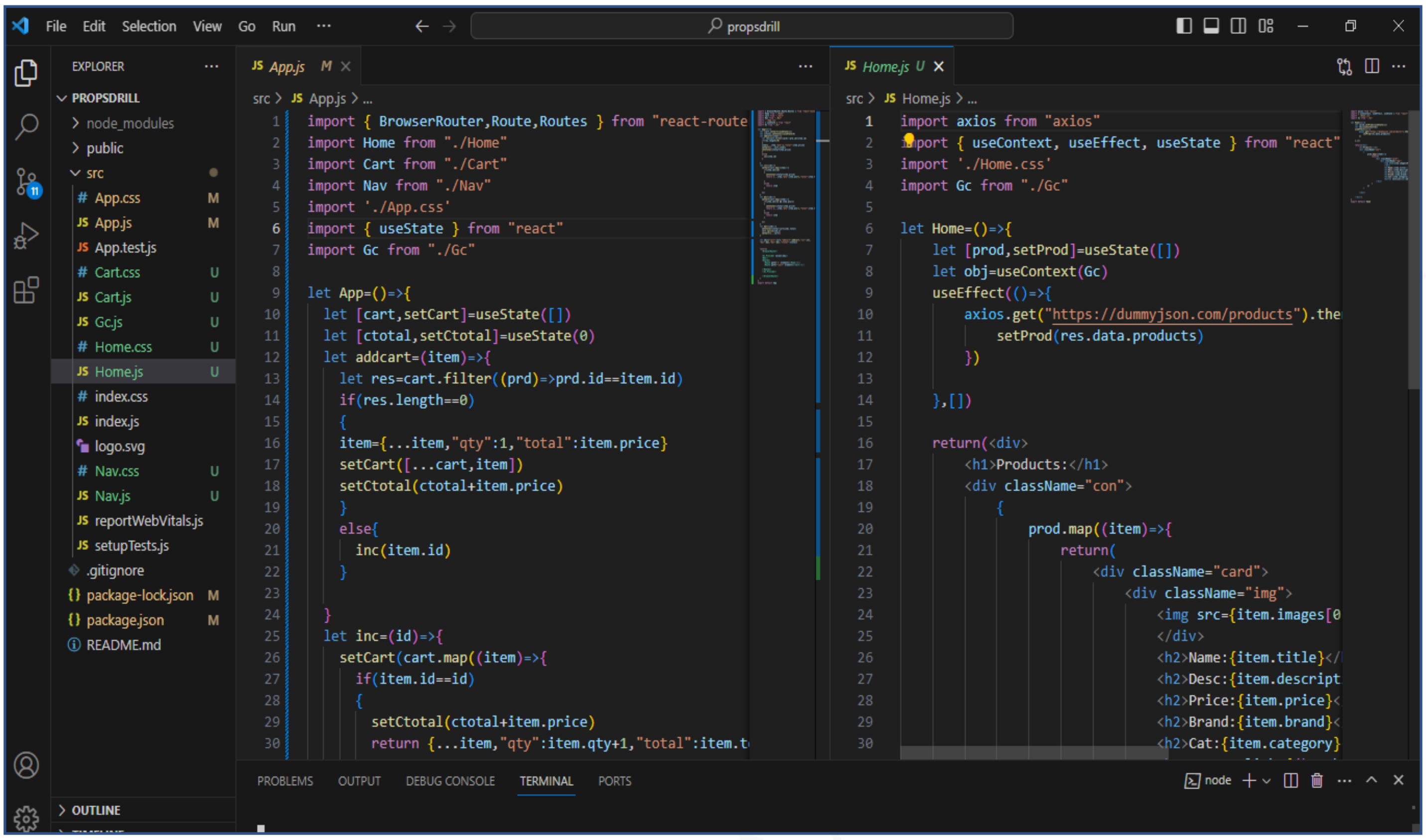
```
import { createContext } from "react";
let Gc=createContext({})
export default Gc
```

➢ **Nav.js:**

```
import {Link} from 'react-router-dom'
import './Nav.css'
import { useContext } from 'react'
import Gc from './Gc'
let Nav=()=>{
  let obj=useContext(Gc)
  return(<nav>
    <Link to="/">Home</Link>
    <Link to="/cart">Cart<button>{obj.cart.length}</button></Link>
  </nav>)
}
export default Nav
```

➢ **Nav.css:**

```
nav{
  width: 100%;
  height: 10vh;
  background: gray;
  display: flex;
  gap:30px;
  padding: 0px 20px;
  align-items: center;}
nav a{
  font-size: 18px;
  color: blue;}
```

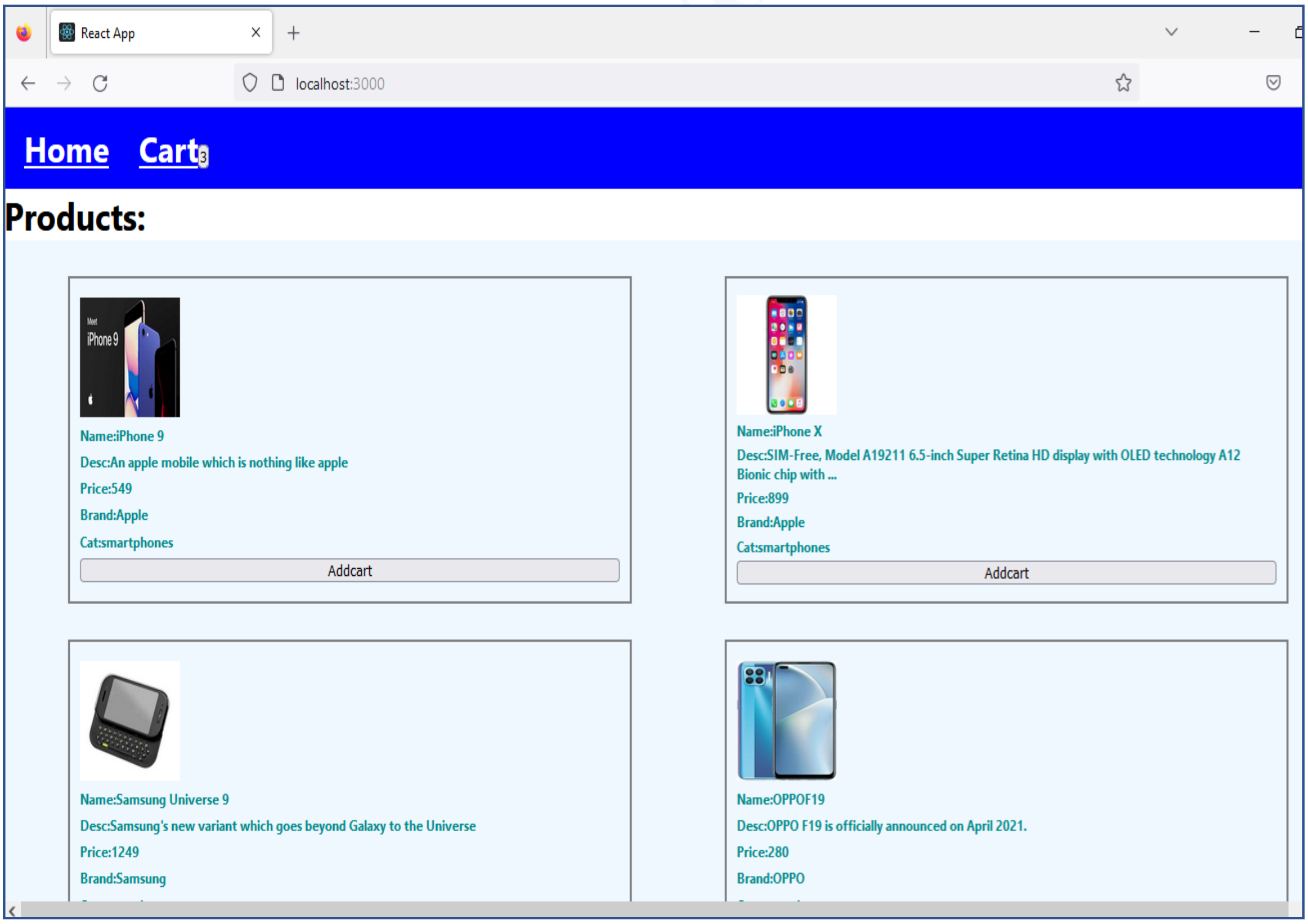


App.js

```
import { BrowserRouter,Route,Routes } from "react-route
import Home from "./Home"
import Cart from "./Cart"
import Nav from "./Nav"
import './App.css'
import { useState } from "react"
import Gc from "./Gc"

let App=()=>{
  let [cart,setCart]=useState([])
  let [ctotal,setCtotal]=useState(0)
  let addcart=(item)=>{
    let res=cart.filter((prd)=>prd.id==item.id)
    if(res.length==0)
    {
    item={...item,"qty":1,"total":item.price}
    setCart([...cart,item])
    setCtotal(ctotal+item.price)
    }
    else{
      inc(item.id)
    }

  }
  let inc=(id)=>{
    setCart(cart.map((item)=>{
      if(item.id==id)
      {
        setCtotal(ctotal+item.price)
        return {...item,"qty":item.qty+1,"total":item.t
```

Home.js

```
import axios from "axios"
import { useContext, useEffect, useState } from "react"
import './Home.css'
import Gc from "./Gc"

let Home=()=>{
    let [prod,setProd]=useState([])
    let obj=useContext(Gc)
    useEffect(()=>{
        axios.get("https://dummyjson.com/products").the
            setProd(res.data.products)
        })

    },[])

    return(<div>
        <h1>Products:</h1>
        <div className="con">
            {
                prod.map((item)=>{
                    return(
                        <div className="card">
                            <div className="img">
                                <img src={item.images[0
                                </div>
                                <h2>Name:{item.title}</
                                <h2>Desc:{item.descript
                                <h2>Price:{item.price}<
                                <h2>Brand:{item.brand}<
                                <h2>Cat:{item.category}
```



Home Cart3

Products:

Name:iPhone 9
Desc:An apple mobile which is nothing like apple
Price:549
Brand:Apple
Cat:smartphones
Addcart

Name:iPhone X
Desc:SIM-Free, Model A19211 6.5-inch Super Retina HD display with OLED technology A12 Bionic chip with ...
Price:899
Brand:Apple
Cat:smartphones
Addcart

Name:Samsung Universe 9
Desc:Samsung's new variant which goes beyond Galaxy to the Universe
Price:1249
Brand:Samsung

Name:OPPOF19
Desc:OPPO F19 is officially announced on April 2021.
Price:280
Brand:OPPO

Add user, Display user, Edit User Update user and delete user application React js:

- **App.js**

```
import { BrowserRouter,Route,Routes } from "react-router-dom"
import { useState } from "react"
import Adduser from "./Adduser"
import Nav from "./Nav"
import Deluser from "./Deluser"
import Dpusers from "./Dpusers"
import Edituser from "./Edituser"
import Gc from "./Gc"
import "./App.css"
let App=()=>{
  let [data,setData]=useState([])
  let addUser=(user)=>{
    setData([...data,user])
  }
  let delUser=(ind)=>{
    data.splice(ind,1)
    setData([...data])
  }
  let editUser=(user,ind)=>{
    data[ind]=user
    setData([...data])

  }
  let obj={"data":data,"fun":addUser,"edit":editUser,"del":delUser}
  return(
    <BrowserRouter>
    <Nav/>
    <Gc.Provider value={obj}>
    <Routes>
      <Route path="/" element={<Dpusers />}/>
      <Route path="/add" element={<Adduser />}/>
      <Route path="/edit" element={<Edituser/>}/>
      <Route path="/del" element={<Deluser />}/>
    </Routes>
    </Gc.Provider>
    </BrowserRouter>
  )

}
export default App
```

- **Deleteuser.js**

```
import { useContext } from "react"
import Gc from "./Gc"
let Deluser=()=>{
   let props=useContext(Gc)
   return(<div>
      {
         props.data.map((user,ind)=>{
            return(<li>
               {user}<button onClick={()=>props.del(ind)}>Delete</button>
            </li>)
         })
      }
   </div>)
}
export default Deluser
```

- **Displayuser.js**

```
import { useContext } from "react"
import Gc from "./Gc"
let Dpusers=()=>{
   let props=useContext(Gc)
   return(
      <div>
        {props.data.map((user)=>{
         return <li>{user}</li>
        })}
      </div>
   )
}
export default Dpusers
```

Edituser.js

```
import { useContext, useState } from "react"
import Gc from "./Gc"

let Edituser=()=>{
   let [user,setUser]=useState("")
   let [index,setIndex]=useState()
   let props=useContext(Gc)
   let edit=(name,ind)=>{
      setUser(name)
      setIndex(ind)

   }
   let upd=()=>{
      props.edit(user,index)
      setUser("")
```

```
    }
    let fun=(e)=>{
      setUser(e.target.value)
    }
    return(
      <div>
        <input type="text" value={user} onChange={fun}/>
        <button onClick={upd}>update</button>
        <div>
       {
        props.data.map((user,ind)=>{
    return(<li>{user} <button onClick={()=>edit(user,ind)}>edit</button></li>)
        })

       }
       </div>
      </div>
    )
  }
  export default Edituser;
```

- Gc.js

```
  import { createContext } from "react";
  let Gc=createContext({})
  export default Gc
```

➢ Data access from form:

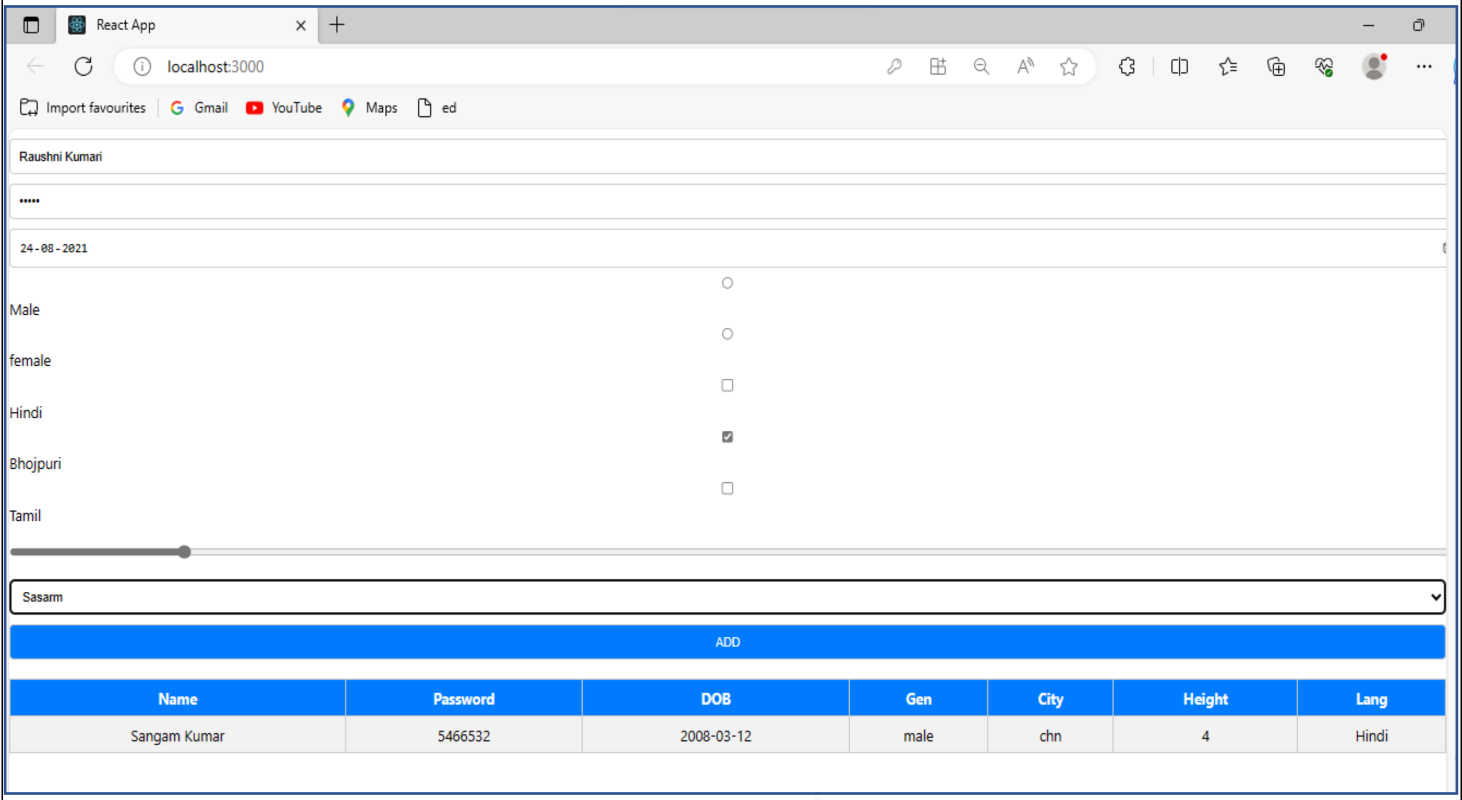
App.js

```
import { useState } from "react"
import "./App.css"
let App=()=>{
 let [data,setData]=useState({})
 let [lang,setLang]=useState([])
 let [users,setUsers]=useState([])

 let fun=(e)=>{

   setData({...data,[e.target.name]:e.target.value})
 }
 let fun1=(e)=>{
   if(e.target.checked)
   {
     setLang([...lang,e.target.value])
   }
   else{
     lang.splice(lang.indexOf(e.target.value),1)
     setLang([...lang])

   }

 }
 let add=()=>{
setUsers([...users,{...data,"lang":[...lang]}])
setData({"name":"","pwd":"","dob":"","city":"","ht":5,"gen":""})
setLang([])
 }
 return(
   <div>
     <div>
     <input type="text" name="name" onChange={fun} value={data.name} placeholder="Enter
your name"/>
     <input type="password" name="pwd" onChange={fun} value={data.pwd} placeholder="Enter
your password"/>
     <input type="date" name="dob" onChange={fun} value={data.dob}/>
<input type="radio" name="gen" onChange={fun} value="male"
checked={data.gen==='male'} />Male
     <input type="radio" name="gen" onChange={fun} value="female"
      checked={data.gen==='female'}/>female

     <input type="checkbox" value="Hindi" onChange={fun1}/>Hindi
     <input type="checkbox" value="Bhojpuri" onChange={fun1}/>Bhojpuri
     <input type="checkbox" value="Tamil" onChange={fun1}/>Tamil
```

```
    <input type="range" min="1" max="10" name="ht" onChange={fun} value={data.ht}/>
    <select onChange={fun} name="city" value={data.city}>
      <option value="" disabled selected>Select</option>
      <option value=" Patna ">Patna</option>
      <option value=" Sasaram ">Sasaram</option>
      <option value="Baheri">Baheri</option>

    </select>
    <button onClick={add}>ADD</button>

    </div>
    <table border={1}>
      <tr>
        <th>Name</th>
        <th>Password</th>
        <th>DOB</th>
        <th>Gen</th>
        <th>City</th>
        <th>Height</th>
        <th>Lang</th>

      </tr>
      {
        users.map((item)=>{
          return(<tr>
            <td>{item.name}</td>
            <td>{item.pwd}</td>
            <td>{item.dob}</td>
            <td>{item.gen}</td>
            <td>{item.city}</td>
            <td>{item.ht}</td>
            <td>{item.lang.toString()}</td>
          </tr>)
        })
      }
    </table>

  </div>
 )

}
export default App
```

- **App.css**

```
.container {
  width: 100%;
  margin: 0 auto;
  padding: 20px;
  background-color: #f5f5f5;
  border: 1px solid #ccc;
  border-radius: 5px;
 }
  input,
 select,
 button {
  display: block;
  margin: 10px 0;
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
 }
  button {
  background-color: #007bff;
  color: #fff;
  cursor: pointer;
 }
  table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
 }
  table, th, td {
  border: 1px solid #ccc;
 }
  th, td {
  padding: 8px;
  text-align: center;
 }
  th {
  background-color: #007bff;
  color: #fff;
 }

 tr:nth-child(even) {
  background-color: #f2f2f2;
 }
 tr:hover {
  background-color: #ddd;
 }
```



Raushni Kumari

•••••

24-08-2021

Male

female

Hindi

Bhojpuri [✓]

Tamil

Sasarm

ADD

Name	Password	DOB	Gen	City	Height	Lang
Sangam Kumar	5466532	2008-03-12	male	chn	4	Hindi

How to use material UI and Ant Design from used pre define react js component

➢ App.js

```
import TextRating from "./TextRating"
import X from "./X"
import Y from "./Y"

let App=()=>{
 return(
  <div>
   <TextRating/>
   <X/>
   <div style={{"width":"100%","height":"500px"}}>
   <Y/>
   </div>
  </div>
 )

}
export default App
```

➢ X.js

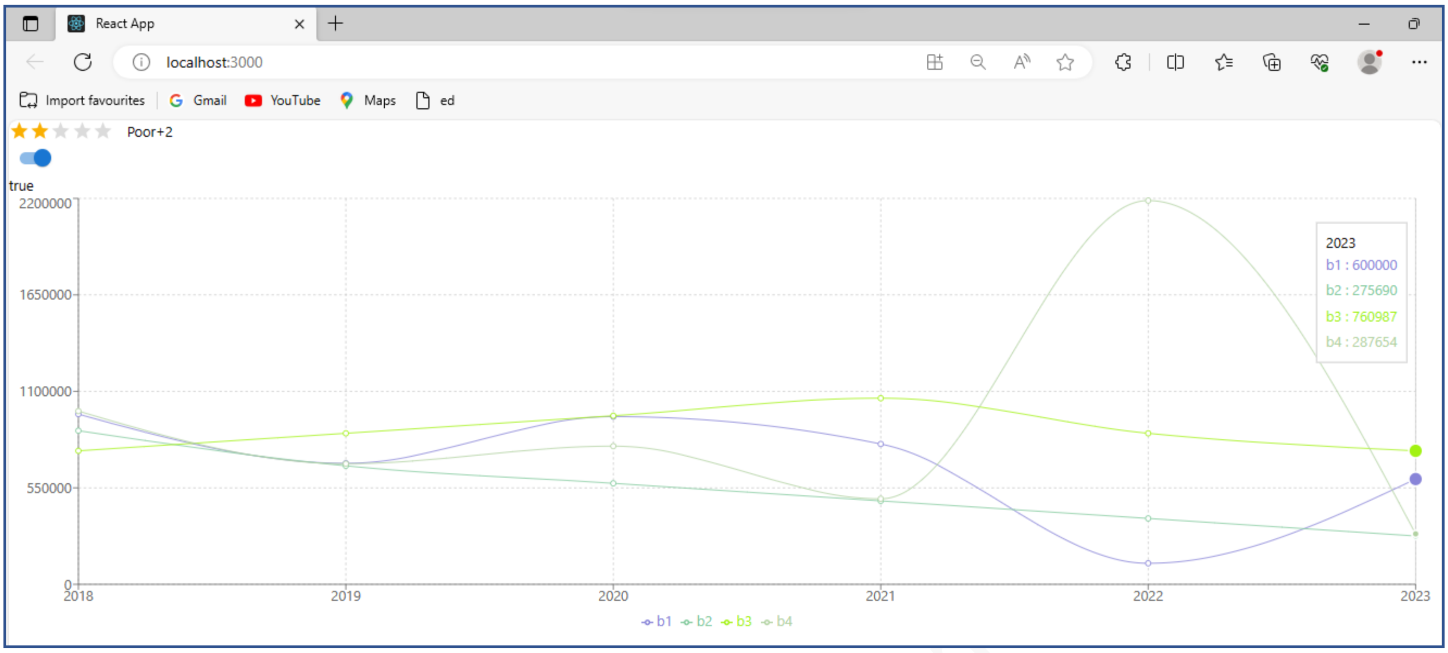
```
import * as React from 'react';
import Switch from '@mui/material/Switch';
export default function X() {
 const [checked, setChecked] = React.useState(true);

 const handleChange = (event) => {
  setChecked(event.target.checked);
 };

 return (<div>
  <Switch
   checked={checked}
   onChange={handleChange}
   inputProps={{ 'aria-label': 'controlled' }}
  />
  <div>{checked+""}</div>
  </div>
 );
}
```

Y.js

```
import React, { PureComponent } from 'react';
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer } from
'recharts';
const data = [{"year":2018,"b1":970000,"b2":875690,"b3":760987,"b4":987654},
{"year":2019,"b1":690000,"b2":675690,"b3":860987,"b4":687654},
{"year":2020,"b1":956000,"b2":575690,"b3":960987,"b4":787654},
{"year":2021,"b1":800000,"b2":475690,"b3":1060987,"b4":487654},
{"year":2022,"b1":120000,"b2":375690,"b3":860987,"b4":2187654},
{"year":2023,"b1":600000,"b2":275690,"b3":760987,"b4":287654}

];

export default class Y extends PureComponent {


  render() {
    return (
      <ResponsiveContainer width="100%" height="100%">
        <LineChart
          width={500}
          height={300}
          data={data}
          margin={{
            top: 5,
            right: 30,
            left: 20,
            bottom: 5,
          }}>
          <CartesianGrid strokeDasharray="3 3" />
          <XAxis dataKey="year" />
          <YAxis />
          <Tooltip />
          <Legend />
          <Line type="monotone" dataKey="b1" stroke="#8884d8" activeDot={{ r: 8 }} />
          <Line type="monotone" dataKey="b2" stroke="#82ca9d" />
          <Line type="monotone" dataKey="b3" stroke="#a2f212" activeDot={{ r: 8 }} />
          <Line type="monotone" dataKey="b4" stroke="#b2d2a2" />

        </LineChart>
      </ResponsiveContainer>
    );
  }
}
```

TextRating.js

```
import * as React from 'react';
import Rating from '@mui/material/Rating';
import Box from '@mui/material/Box';
import StarIcon from '@mui/icons-material/Star';
const labels = {
  0.5: 'Useless',
  1: 'Useless+',
  1.5: 'Poor',
  2: 'Poor+',
  2.5: 'Ok',
  3: 'Ok+',
  3.5: 'Good',
  4: 'Good+',
  4.5: 'Excellent',
  5: 'Excellent+',};
function getLabelText(value) {
  return `${value} Star${value !== 1 ? 's' : ''}, ${labels[value]}`;
}
export default function TextRating() {
  const [value, setValue] = React.useState(2);
  const [hover, setHover] = React.useState(-1);
  return (
    <Box
      sx={{
        width: 200,
        display: 'flex',
        alignItems: 'center',
      }}>
      <Rating
        name="hover-feedback"
        value={value}
        precision={0.5}
        getLabelText={getLabelText}
        onChange={(event, newValue) => {
          setValue(newValue);
        }}
        onChangeActive={(event, newHover) => {
          setHover(newHover);
        }}
        emptyIcon={<StarIcon style={{ opacity: 0.55 }} fontSize="inherit" />}/>
      {value !== null && (
        <Box sx={{ ml: 2 }}>{labels[hover !== -1 ? hover : value]}</Box>
      )}
      <div>{value}</div>
    </Box>
  );}
```



```
2023
b1 : 600000
b2 : 275690
b3 : 760987
b4 : 287654
```

important question and answer:

1. What is React.js?
   - React.js is an open-source JavaScript library for building user interfaces.
2. What is JSX in React.js?
   - JSX is a syntax extension for JavaScript that allows you to write HTML-like code in your JavaScript files.
3. What are the key features of React.js?
   - React.js features a virtual DOM, component-based architecture, and one-way data binding.
4. What are components in React.js?
   - Components are the building blocks of a React application, representing UI elements.
5. What is the difference between state and props?
   - State is used for data that should be saved and modified by a component, while props are used to pass data from parent to child components.
6. How do you create a functional component in React?
   - You can create a functional component by defining a JavaScript function that returns JSX.
7. How do you create a class component in React?
   - You can create a class component by extending the `React.Component` class and defining a `render` method.
8. What is the purpose of the `render()` method in React components?
   - The `render()` method is used to describe what the component's UI should look like.
9. What is the virtual DOM in React?
   - The virtual DOM is a lightweight copy of the actual DOM, used to improve rendering performance.
10. What is the significance of keys in React lists?
    - Keys are used to uniquely identify elements in a list and help React efficiently update the DOM.
11. What is the use of React Router?
    - React Router is used for adding navigation and routing to React applications.
12. How do you handle user input in React forms?
    - You can manage form input by using controlled components and handling the `onChange` event.
13. What is the purpose of `setState()` in React?
    - `setState()` is used to update the state of a React component, triggering re-rendering.
14. What are props in React Router?
    - Props are parameters that are passed to React components, allowing you to customize component behavior.
15. How do you pass data from a parent component to a child component?
    - Data is passed from parent to child components by setting props in the child component.
16. What is React context?
    - React context provides a way to share data across components without passing props manually.
17. What is conditional rendering in React?
    - Conditional rendering involves showing or hiding components based on certain conditions.
18. How do you perform side effects in React?
    - Side effects are managed using the `useEffect` hook in functional components.
19. What is Redux and why is it used in React?
    - Redux is a state management library used to manage application state in a predictable and centralized manner.
20. What are actions and reducers in Redux?

- Actions are plain JavaScript objects that describe what happened, and reducers specify how the application's state changes.

21. How do you connect a React component to the Redux store?
   - You can use the `connect` function from the `react-redux` library to connect a component to the store.
22. What is the purpose of the `mapStateToProps` and `mapDispatchToProps` functions in React Redux?
   - `mapStateToProps` maps the state from the store to component props, while `mapDispatchToProps` maps action creators to component props.
23. What is the difference between React and React Native?
   - React is used for building web applications, while React Native is used for building mobile applications for iOS and Android.
24. How do you handle navigation in a React Native app?
   - Navigation in React Native can be managed using libraries like React Navigation.
25. What is the purpose of Redux Thunk in Redux?
   - Redux Thunk is a middleware for handling asynchronous actions in Redux.
26. What is React Hook Form in React.js?
   - React Hook Form is a library for managing forms in React using hooks.
27. What is the significance of the `keyExtractor` function in React Native?
   - `keyExtractor` is used to extract unique keys for items in a FlatList or a SectionList component in React Native.
28. What is a Higher Order Component (HOC) in React?*
   - A Higher Order Component is a pattern for sharing behavior among multiple React components.
29. What are the benefits of using functional components over class components?
   - Functional components are simpler, have better performance, and support hooks for managing state and side effects.
30. What are hooks in React and why are they used?
   - Hooks are functions that allow functional components to manage state and side effects, improving code reuse and component logic.
31. How does React handle security vulnerabilities like Cross-Site Scripting (XSS)?
   - React provides built-in protections to prevent XSS attacks, such as escaping dynamic content by default.
32. What is the purpose of the `useMemo` hook in React?
   - `useMemo` is used to memoize expensive calculations and optimize component performance.
33. What is React Portals and when would you use them?
   - Portals allow rendering components outside the current DOM hierarchy, useful for modals, tooltips, and other overlays.
34. What is the significance of the `Fragment` component in React?
   - The `Fragment` component is a wrapper that doesn't create extra DOM elements, useful for grouping elements.
35. How do you implement server-side rendering (SSR) in React applications?
   - You can use libraries like Next.js to enable SSR in React applications.
36. What is code splitting in React and why is it important?
   - Code splitting is the process of splitting your code into smaller, more manageable pieces, improving load times and performance.
37. How do you handle errors in React components?
   - You can use error boundaries and the `componentDidCatch` lifecycle method to handle errors gracefully.

38. What is the difference between `ref` and `key` in React?
   - `ref` is used to access the underlying DOM element or React component, while `key` is used for reconciliation in lists.
39. How do you update the state of a component in React?
   You should never modify the state directly; instead, use the `setState` method to update the state.
40. What is the purpose of the `children` prop in React components?
   The `children` prop is a special prop that allows you to pass arbitrary JSX content between opening and closing tags.
41. What are controlled components in React forms?
   Controlled components have their value controlled by React state, making it easy to handle form input.
42. What is the purpose of the `useContext` hook in React?
   The `useContext` hook allows functional components to access the context without a consumer component.
43. What are the benefits of using Redux for state management in React applications?
   Redux provides a centralized, predictable, and easily testable way to manage state in complex

❖ Research(अनुसंधान):

- अनुसंधान एक प्रणालीकरण कार्य होता है जिसमें विशेष विषय या विषय की नई ज्ञान एवं समझ को प्राप्त करने के लिए सिद्धांतिक जांच और अध्ययन किया जाता है। इसकी प्रक्रिया में डेटा का संग्रह और विश्लेषण, निष्कर्ष निकालना और विशेष क्षेत्र में मौजूदा ज्ञान में योगदान किया जाता है। अनुसंधान के माध्यम से विज्ञान, प्रोधोगिकी, चिकित्सा, सामाजिक विज्ञान, मानविकी, और अन्य क्षेत्रों में विकास किया जाता है। अनुसंधान की प्रक्रिया में अनुसंधान प्रश्न या कल्पनाएँ तैयार की जाती हैं, एक अनुसंधान योजना डिज़ाइन की जाती है, डेटा का संग्रह किया जाता है, विश्लेषण किया जाता है, निष्कर्ष निकाला जाता है और परिणामों को उचित दर्शाने के लिए समाप्ति तक पहुंचाया जाता है।

❖ Innovation(नवीनीकरण): -

- Innovation (इनोवेशन) एक विशेषता या नई विचारधारा की उत्पत्ति या नवीनीकरण है। यह नए और आधुनिक विचारों, तकनीकों, उत्पादों, प्रक्रियाओं, सेवाओं या संगठनात्मक ढंगों का सृजन करने की प्रक्रिया है जिससे समस्याओं का समाधान, प्रतिस्पर्धा में अग्रणी होने, और उपयोगकर्ताओं के अनुकूलता में सुधार किया जा सकता है।

❖ Discovery (आविष्कार):

- Discovery का अर्थ होता है "खोज" या "आविष्कार"। यह एक विशेषता है जो किसी नए ज्ञान, अविष्कार, या तत्व की खोज करने की प्रक्रिया को संदर्भित करता है। खोज विज्ञान, इतिहास, भूगोल, तकनीक, या किसी अन्य क्षेत्र में हो सकती है। इस प्रक्रिया में, व्यक्ति या समूह नए और अज्ञात ज्ञान को खोजकर समझने का प्रयास करते हैं और इससे मानव सभ्यता और विज्ञान-तकनीकी के विकास में योगदान देते हैं।

Note: अनुसंधान विशेषता या विषय पर नई ज्ञान के प्राप्ति के लिए सिस्टमैटिक अध्ययन है, जबकि आविष्कार नए और अज्ञात ज्ञान की खोज है।

TWKSAA RID MISSION

(Research) अनुसंधान करने के महत्वपूर्ण कारण:	(Innovation) नवीनीकरण करने के महत्वपूर्ण कारण:	(Discovery) खोज करने के महत्वपूर्ण कारण:
1. नई ज्ञान की प्राप्ति	1. प्रगति के लिए	1. नए ज्ञान की प्राप्ति
2. समस्याओं का समाधान	2. परिवर्तन के लिए	2. ज्ञान के विकास में योगदान
3. तकनीकी और व्यापार में उन्नति	3. उत्पादन में सुधार	3. अविष्कारों की खोज
4. विकास को बढ़ावा देना	4. प्रतिस्पर्धा में अग्रणी होने के लिए	4. समस्याओं का समाधान
5. सामाजिक प्रगति	5. समाज को लाभ	5. समाज के उन्नति का माध्यम
6. देश विज्ञान और प्रौद्योगिकी का विकास	6. देश विज्ञान और प्रौद्योगिकी के विकास।	6. देश विज्ञान और तकनीक के विकास

➢ जो लोग रिसर्च, इनोवेशन और डिस्कवरी करते हैं उन लोगों को ही हमें अपना नायक, प्रतीक एवं आदर्श मानना चाहिए क्योंकि ये लोग हमारे समाज, देश एवं विज्ञान के क्षेत्र में प्रगति, विकास और समस्याओं के समाधान में महत्वपूर्ण भूमिका निभाते हैं।

मैं राजेश प्रसाद एक वीणा उठाया हूँ Research, Innovation and Discovery का जिसका मुख्य उदेश्य हैं आने वाले समय में सबसे पहले New(RID, PMS & TLR) की खोज, प्रकाशन एवं उपयोग भारत की इस पावन धरती से ही हो |

“अगर आप भी Research, Innovation and Discovery के क्षेत्र में रूचि रखतें हैं एवं अपनी प्रतिभा से दुनियां को कुछ नया देना चाहतें तो हमारे इस त्वक्सा रीड मिशन (TWKSAA RID MISSION) से जरुर जुड़ें” |

- **राजेश प्रसाद**