React JS Notes

FSD -Central Training Team

ABES Engineering College, Ghaziabad

A. What is React JS

React.js is a popular open-source JavaScript library for building user interfaces, primarily for single-page applications (SPA). It allows developers to build web applications that can update and render efficiently in response to data changes. React is maintained by Facebook and a community of developers.

Benefits of Using React.js

- **Declarative UI**: React allows you to build user interfaces by describing what the UI should look like based on the application's state.
- **Reusable Components**: Components are independent, reusable pieces of code that make development easier and more efficient.
- **Virtual DOM**: React's virtual DOM ensures that changes are efficiently applied to the real DOM, improving performance.
- **Rich Ecosystem**: React has a large ecosystem of libraries and tools to extend its capabilities, from routing (React Router) to state management (Redux, MobX).
- **Community and Support**: React is maintained by Facebook and has a large, active developer community.

B. Rendering Approach

Traditional Website:

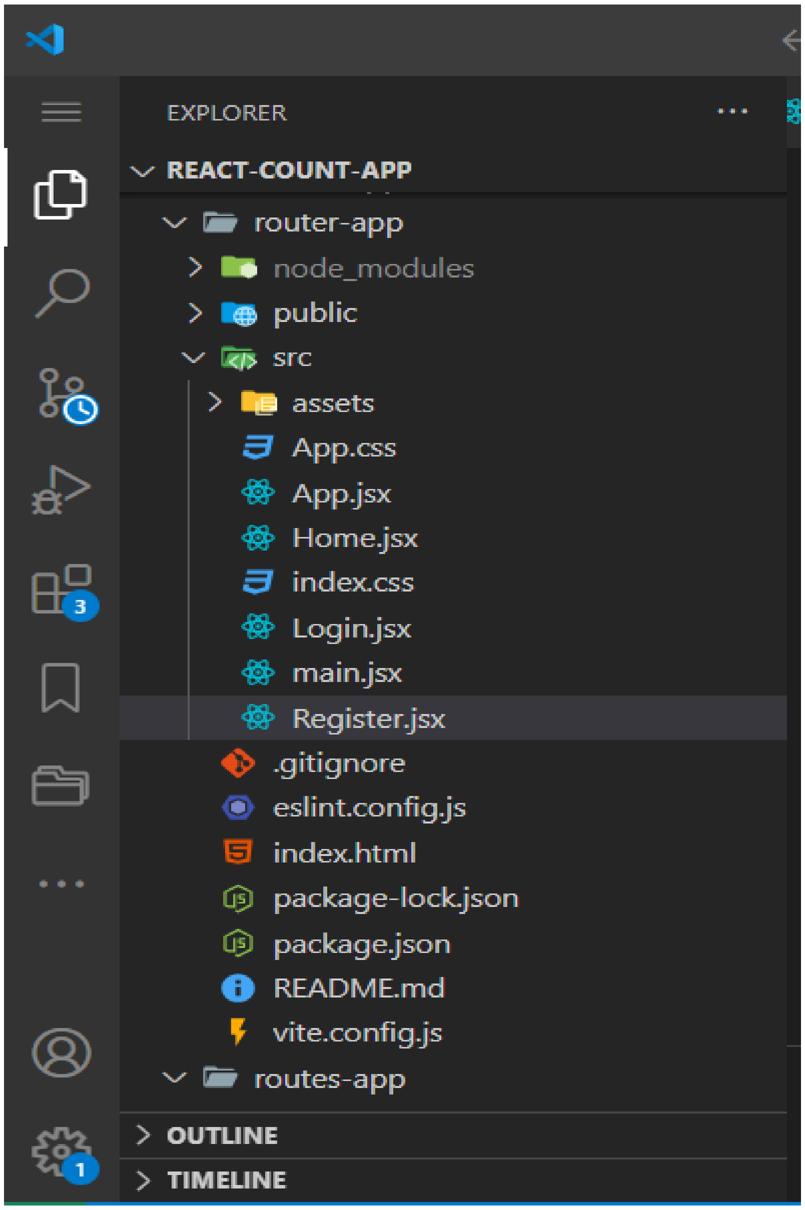
- Server-Side Rendering (SSR): Each time a user navigates to a new page or performs an action, the browser sends a request to the server.
- Full-Page Reloads: The server responds by rendering a full HTML page and sending it back, which the browser displays. This process can be slow as each request triggers a complete page reload.
- HTML-Driven UI: Pages are static or dynamically generated on the server and sent to the client.

React.js (SPA):

- Client-Side Rendering (CSR): The initial load fetches a minimal HTML page and a JavaScript bundle. After this, React takes over rendering on the client side.
- Single-Page Application (SPA): React manipulates the DOM dynamically, updating only the parts of the page that need changing, without reloading the entire page. This creates a smoother, faster user experience.
- Component-Based UI: The UI is broken down into reusable components, allowing for efficient updates and maintenance.

C. Basic Folder Structure

When you create a new React project with Vite, the default structure looks something like this:



```
EXPLORER
REACT-COUNT-APP
  router-app
    node_modules
    public
    src
      assets
      App.css
      App.jsx
      Home.jsx
      index.css
      Login.jsx
      main.jsx
      Register.jsx
    .gitignore
    eslint.config.js
    index.html
    package-lock.json
    package.json
    README.md
    vite.config.js
  routes-app
OUTLINE
TIMELINE
```

Key Folders and Files

node_modules

This folder contains all the dependencies and packages installed via npm. It’s managed automatically by npm, so you generally don’t need to interact with it directly.

public

The public folder holds static assets that are accessible directly in the application, such as images or icons. Vite will serve files from this folder as-is, meaning they won't be processed or bundled.

src

The src folder is where the main code for your application lives. Here's a breakdown of the core files and subfolders:

assets: This folder is typically used for storing images, fonts, and other static assets specific to your application. Vite uses a more efficient way to handle assets, allowing you to import them directly in your JavaScript/JSX files.

components: This folder holds reusable React components that you'll use throughout your application. For example, you might have a Header.jsx, Footer.jsx, or Button.jsx component here.

App.jsx: This is the main component that serves as the root of your application. You can think of it as the container for your entire app, where you structure routes or render top-level components.

App.css: A CSS file used for styling the App.jsx component or for global styles. You can organize additional CSS files here or in individual component folders as needed.

index.css: This is the global CSS file for base styles or reset CSS.

main.jsx: This is the entry point of the application. It's responsible for rendering the App component into the DOM using ReactDOM.createRoot(). Here, you'll also set up the context providers or routing libraries if needed.

.gitignore

This file specifies which files and folders should be ignored by Git. Common entries include node_modules, environment variable files, and build outputs.

index.html

Vite provides a direct entry point for the application using this index.html file. Unlike typical React setups (like create-react-app), Vite allows for a more flexible index.html, which is used to inject the root of the app directly.

package.json

This file contains metadata about your project, including its name, version, dependencies, and scripts. Important scripts include:

dev: Starts the development server.
build: Creates an optimized production build.
preview: Previews the production build.

package-lock.json

The package-lock.json file is automatically generated when you run npm install in a Node.js project, including projects created with React and Vite.

The package-lock.json file helps ensure consistent dependency management across different environments and among team members. Its main roles are to:

- Lock Dependency Versions: It locks the exact versions of each package and its dependencies. This ensures that every time you or someone else installs the project, the exact same dependency versions are used, avoiding unexpected issues caused by version updates.

- Improve Installation Speed: package-lock.json allows npm to skip version resolution during installation, making it faster since npm knows exactly which versions to install

NOTE: `package.json` specifies the dependencies a project requires, `package-lock.json` provides an exact dependency tree.

vite.config.js

This configuration file is where you can customize Vite’s behavior. For example, you might specify a different public base path, define environment variables, or configure plugins (like React or TypeScript support).

README.md

The README provides an overview of your project, instructions for setup, usage, and any other relevant details for future developers or users.

Environment Variables: Vite uses .env files for environment variables, such as .env or .env.production.

Routing and State Management: If your app grows, you might add folders for features like routing (e.g., react-router-dom) or state management (e.g., Redux, Context API).

D. Key Concepts of React.js

1. **Component-Based Architecture**
    - **Components**: React applications are made up of components, which are small, reusable pieces of code that represent parts of the user interface (UI). Each component can have its own state and logic.
    - **Functional Components**: These are simple JavaScript functions that accept props (inputs) and return JSX (UI elements).

Example of a simple **functional component**:

```
import React from "react";
const Greeting = ({ name }) => {
  return <h1>Hello, {name}!</h1>;
};
export default Greeting;
```

2. **JSX (JavaScript XML)**
    - **JSX** is a syntax extension that allows writing HTML-like code inside JavaScript. It makes it easier to write UI logic alongside JavaScript logic.
    - JSX is not a necessity but is commonly used in React applications to improve readability.

Example of JSX:

```
const element = <h1>Hello, world!</h1>;
```

3. **Virtual DOM**
    - React uses a **virtual DOM**, which is an in-memory representation of the real DOM. When the state of a component changes, React updates the virtual DOM first, then efficiently updates only the parts of the actual DOM that have changed.
    - This improves performance by minimizing expensive DOM manipulations.
4. **One-Way Data Binding**
    - React enforces a unidirectional data flow. The data flows from parent components to child components through **props** (properties). This makes tracking data changes and debugging easier.

5. **State Management**
    - React components can have **state**, which is an object that holds data that may change over time. The state is managed using the useState hook in functional components or the setState method in class components.

6. **React Hooks**
    - **Hooks** are special functions that allow functional components to use React features such as state and lifecycle methods.

7. **Props (Properties)**

Props are the way to pass data from parent components to child components. Props are read-only and cannot be modified by the child component.

Props (Properties)

- **Props** are the way to pass data from parent components to child components. Props are read-only and cannot be modified by the child component. In React, props (short for "properties") are used to pass data from parent components to child components.
- There are several different ways to pass and use props in React, ranging from basic prop passing to more advanced techniques involving destructuring, default props, and using functions as props. Below are various methods to work with props in React:

Basic Prop Passing

This is the simplest way to pass props. In the parent component, you pass the props as attributes to the child component.

Example:

```
// Parent Component
const Parent = () => {
  return <Child name="John" age={30} />;
};

// Child Component
const Child = (props) => {
  return (
    <div>
      <h1>Name: {props.name}</h1>
      <p>Age: {props.age}</p>
    </div>
  );
};
```

Destructuring Props

Instead of accessing props with props.name, you can destructure the props for cleaner code.

Example:

```
// Parent Component
const Parent = () => {
  return <Child name="John" age={30} />;
};

// Child Component with Destructuring
const Child = ({ name, age }) => {
  return (
    <div>
      <h1>Name: {name}</h1>
      <p>Age: {age}</p>
    </div>
  );
};
```

Passing Objects as Props

You can pass an entire object as a prop, which is helpful when passing multiple related values.

Example:

```
// Parent Component
const Parent = () => {
  const user = { name: "John", age: 30 };
  return <Child user={user} />;
};

// Child Component
const Child = (props) => {
  return (
    <div>
      <h1>Name: {props.user.name}</h1>
      <p>Age: {props.user.age}</p>
    </div>
  );
};
```

You can also destructure the object inside the child component:

```
const Child = ({ user: { name, age } }) => {
  return (
    <div>
      <h1>Name: {name}</h1>
      <p>Age: {age}</p>
    </div>
  );
};
```

4. Default Props

React allows you to define default props in case the parent component does not pass certain props.

Example:

```
// Child Component
const Child = ({ name, age }) => {
  return (
    <div>
      <h1>Name: {name}</h1>
      <p>Age: {age}</p>
    </div>
  );
};

// Default Props
Child.defaultProps = {
  name: "Default Name",
  age: 18,
};

// Parent Component
const Parent = () => {
  return <Child />;
};
```

If the Parent does not pass any props, the Child component will use the default values.

5. Passing Functions as Props (Callback Props)

You can pass a function from the parent component to the child component, which the child can call.

Example:

```
// Parent Component
const Parent = () => {
  const handleClick = () => {
    alert("Button clicked!");
  };
  return <Child onButtonClick={handleClick} />;
};

// Child Component
const Child = ({ onButtonClick }) => {
  return <button onClick={onButtonClick}>Click Me</button>;
};
```

In this example, clicking the button in the Child component will trigger the handleClick function in the Parent.

Props with JSX Spread Operator

The spread operator (...) can be used to pass multiple props at once, especially when passing an object or many props.

Example:

```
const Parent = () => {
  const user = { name: "John", age: 30, occupation: "Developer" };
  return <Child {...user} />;
};

// Child Component
const Child = ({ name, age, occupation }) => {
  return (
    <div>
      <h1>Name: {name}</h1>
      <p>Age: {age}</p>
      <p>Occupation: {occupation}</p>
    </div>
  );
};
```

In this case, all properties of the user object are passed to the Child component as individual props.

Conditional Rendering with Props

Props can be used to conditionally render elements in a child component based on the data passed from the parent.

Example:

```
const Child = ({ isLoggedIn }) => {
  return (
    <div>
      {isLoggedIn ? <p>Welcome back!</p> : <p>Please log in.</p>}
    </div>
  );
};

// Parent Component
const Parent = () => {
  return <Child isLoggedIn={true} />;
};
```

In this example, the child component conditionally renders based on the isLoggedIn prop.

React Hooks

React Hooks are used to hook into the features like state and lifecycle methods of React Component. Generally, Hooks are special functions that provide access to state in the React Application.

Hooks were introduced in the 16.8 version of React. Hooks give access to states for functional components while creating a React application. It allows you to use state and other React features without writing a class.

Common React Hooks

1. **useState**: Manages state in functional components.
2. **useEffect**: Handles side effects like fetching data or updating the DOM.
3. **useContext**: Accesses React context values without wrapping components in a Consumer.
4. **useReducer**: A more complex state management alternative to useState.
5. **useRef**: Accesses DOM elements or persists mutable values.
6. **useMemo**: Optimizes performance by memoizing computed values.
7. **useCallback**: Memoizes functions to prevent unnecessary re-renders.
8. **useNavigate** (React Router): Redirects users to different routes in your app.

1. useState

The useState hook is used to manage state in functional components. It initializes a state variable and provides a function to update it.

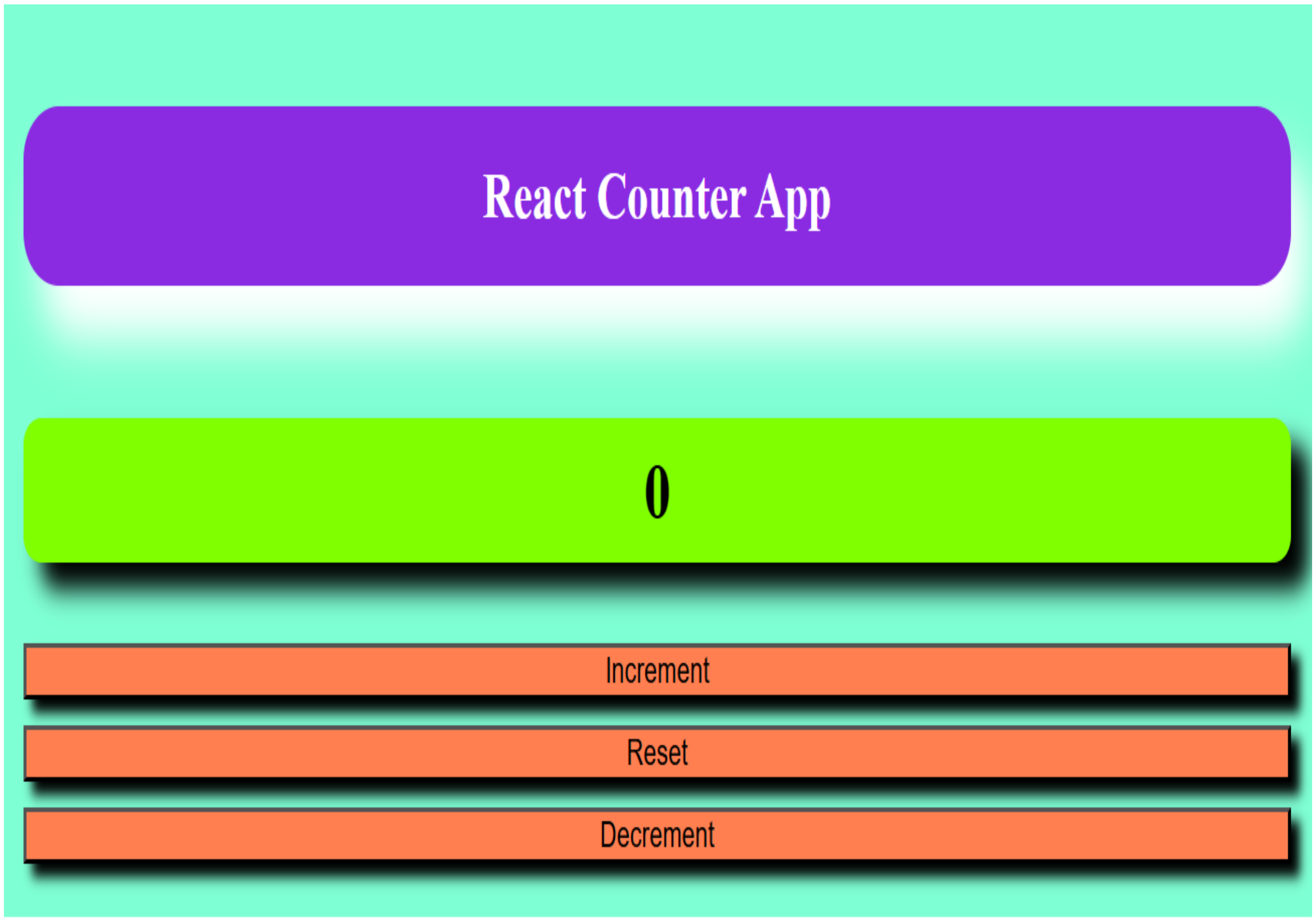
Syntax:

```
const [state, setState] = useState(initialValue);
```

Parameters:

- initialValue: The starting value of the state (e.g., 0, "", {}).
- Returns:
  - state: The current value.
  - setState: A function to update the state.

Example Counter App



CounterApp.jsx

```
import React, { useState } from "react";
import "./CounterApp.css";
const CounterApp = () => {
  const [count, setCount] = useState(0);
  return (
    <div className="disp">
      <h2 className="app">React Counter App</h2>
      <h1 className="header">{count}</h1>
      <button
        className="btn"
        onClick={() => {
          setCount(count + 1);
        }}
      >
        Increment
      </button>
      <button
        className="btn"
        onClick={() => {
          setCount(0);
        }}
      >
        Reset
      </button>
      <button
        className="btn"
        onClick={() => {
          setCount(count - 1);
        }}
      >
        Decrement
      </button>
```

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

CounterApp.css

```
.disp {
  display: flex;
  flex-direction: column;
  background-color: aquamarine;
  padding: 20px;
  gap: 10px;
}
.header {
  background-color: chartreuse;
  font-size: 30px;
  border-radius: 10px;
  text-align: center;
  padding: 10px;
  color: black;
  box-shadow: 10px 10px 10px;
}
.btn {
  background-color: coral;
  box-shadow: 5px 5px 5px;
}
.app {
  text-align: center;
  background-color: blueviolet;
  color: white;
  padding: 20px;
  border-radius: 20px;
  box-shadow: 10px 20px 20px;
}
```

2. useEffect

The useEffect hook performs side effects in functional components. It replaces lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount.

Syntax:

```
useEffect(() => {
  // Side effect code
  return () => {
    // Cleanup code (optional)
  };
}, [dependencies]);
```

Parameters:

- A callback function to execute.
- Dependency array:
  - []: Runs only once (like componentDidMount).
  - [dep1, dep2]: Runs whenever dep1 or dep2 changes.
  - undefined: Runs after every render (use cautiously).

Example: Fetch Data

```
import React, { useState, useEffect } from "react";

const DataFetcher = () => {
  const [data, setData] = useState([]);

  useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/posts")
      .then((response) => response.json())
      .then((data) => setData(data));
  }, []); // Empty dependency array ensures it runs only once

  return (
    <div>
      <h3>Fetched Data:</h3>
      <ul>
```

```
        {data.slice(0, 5).map((item) => (
          <li key={item.id}>{item.title}</li>
        ))}
      </ul>
    </div>
  );
};

export default DataFetcher;
```

---

3. useNavigate (React Router)

The useNavigate hook, part of **React Router v6**, is used for programmatic navigation. It replaces the useHistory hook from earlier versions.

Syntax:

```
const navigate = useNavigate();
navigate(path, options);
```

Parameters:

- path: The URL to navigate to (e.g., "/profile").
- options (optional):
  - replace: If true, replaces the current history entry instead of adding a new one.

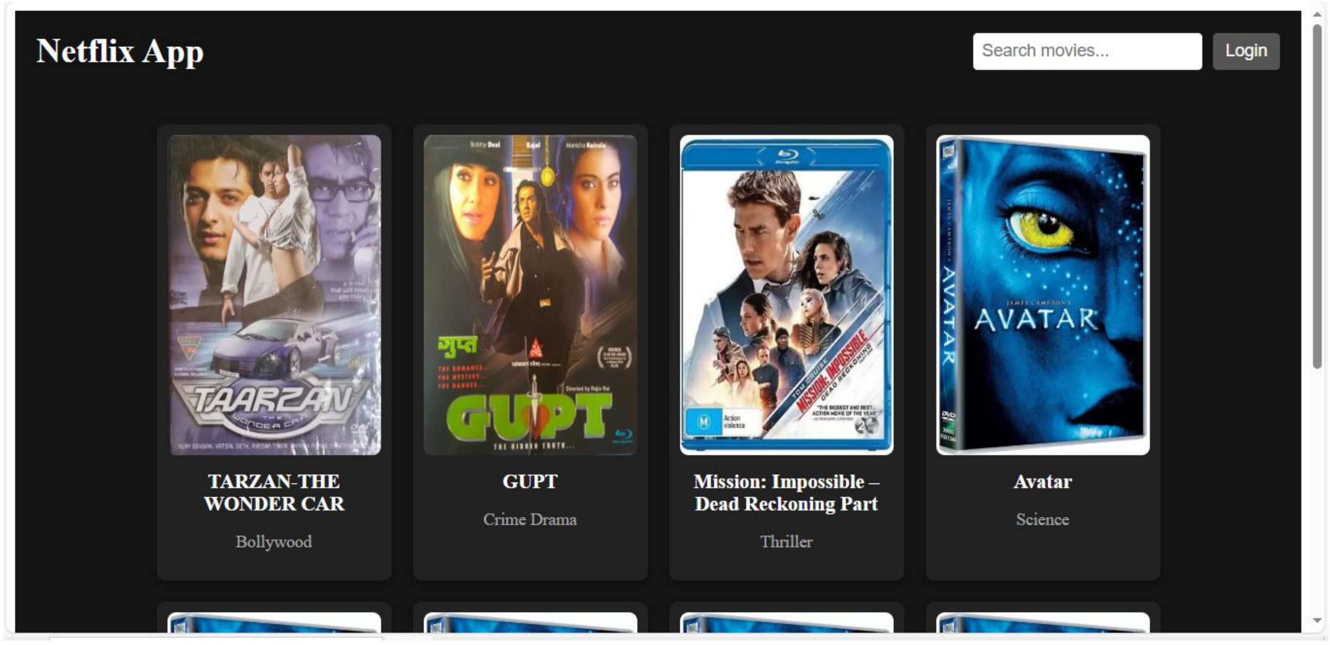
Example: Redirect on Button Click

```
import React from "react";
import { useNavigate } from "react-router-dom";

const Home = () => {
  const navigate = useNavigate();

  const goToProfile = () => {
    navigate("/profile");
  };

  return (
    <div>
      <h3>Welcome to the Home Page</h3>
      <button onClick={goToProfile}>Go to Profile</button>
    </div>
  );
};

export default Home;
```

Example: Redirect on Condition

```
import React, { useEffect } from "react";
import { useNavigate } from "react-router-dom";

const Dashboard = ({ isLoggedIn }) => {
  const navigate = useNavigate();

  useEffect(() => {
    if (!isLoggedIn) {
      navigate("/login");
    }
  }, [isLoggedIn, navigate]);

  return <h3>Welcome to the Dashboard</h3>;
};

export default Dashboard;
```

Netflix App



App.jsx

```
import React, { useState } from "react";
import MovieList from "./MovieList";
const App = () => {
  const [searchTerm, setSearchTerm] = useState("");
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const handleSearchChange = (event) => {
    setSearchTerm(event.target.value);
  };

  const toggleLogin = () => {
    setIsLoggedIn(!isLoggedIn);
  };

  return (
    <div style={{ backgroundColor: "#141414", color: "#fff", minHeight: "100vh" }}>
      <header
        style={{
          padding: "20px",
          display: "flex",
          justifyContent: "space-between",
          alignItems: "center",
        }}
      >
        <h1 style={{ margin: 0 }}>Netflix App</h1>
        <div style={{ display: "flex", alignItems: "center", gap: "10px" }}>
          <input
            type="text"
            placeholder="Search movies..."
            value={searchTerm}
            onChange={handleSearchChange}
            style={{
              padding: "8px",
              borderRadius: "4px",
              border: "none",
              outline: "none",
              fontSize: "16px",
            }}
          />
          <button
            onClick={toggleLogin}
            style={{
              backgroundColor: isLoggedIn ? "#f40612" : "#555",
              color: "#fff",
              padding: "8px 12px",
              border: "none",
              borderRadius: "4px",
              cursor: "pointer",
              fontSize: "16px",
            }}
          >
            {isLoggedIn ? "Logout" : "Login"}
          </button>
        </div>
      </header>
      <MovieList searchTerm={searchTerm} />
    </div>
   );
  };

  export default App;
```

MovieList.jsx

```
import "./MovieList.css";
import movies from "./movies";
const MovieList = () => {
 return (
  <div className="movie-list">
   {movies.map((movie) => (
    <div className="movie-card" key={movie.id}>
     <img src={movie.image} alt={movie.title} className="movie-image" />
     <h3>{movie.title}</h3>
     <p>{movie.genre}</p>
    </div>
   ))}
  </div>
 );
};

export default MovieList;
```

MovieList.css

```css
.movie-list {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  padding: 20px;
}

.movie-card {
  background-color: #222;
  border-radius: 8px;
  margin: 10px;
  padding: 10px;
  text-align: center;
  width: 200px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);
}

.movie-image {
  width: 100%;
  height: 300px;
  border-radius: 8px;
}

.movie-card h3 {
  margin: 10px 0 5px;
}

.movie-card p {
  color: #aaa;
}
```

movies.js

```
const movies = [
 {
   id: 1,
   title: "TARZAN-THE WONDER CAR",
   image:
    "https://rukminim2.flixcart.com/image/612/612/xif0q/movie/4/f/r/2009-dvd-eagle-
hindi-tarzan-the-wonder-car-original-imahfdpjpqg85j5d.jpeg?q=70",
   genre: "Bollywood",
 },
 {
   id: 2,
   title: "GUPT",
   image:
    "https://rukminim2.flixcart.com/image/612/612/xif0q/movie/d/5/q/2013-blu-ray-
eagle-english-gupt-original-imagtxsb28ygkrhy.jpeg?q=70",
   genre: "Crime Drama",
 },
 {
   id: 3,
   title: "Mission: Impossible – Dead Reckoning Part",
   image:
    "https://rukminim2.flixcart.com/image/612/612/xif0q/movie/4/s/t/2023-blu-ray-
paramount-pictures-english-mission-impossible-dead-original-
imaguw429xnwybtg.jpeg?q=70",
   genre: "Thriller",
 },
 {
   id: 4,
   title: "Avatar",
   image:
    "https://rukminim2.flixcart.com/image/612/612/jeokbrk0/movie/z/6/2/2009-dvd-
excel-home-videos-english-avatar-original-imaf3bd6zed7phes.jpeg?q=70",
   genre: "Science",
 },
 {
   id: 5,
   title: "Avatar",
   image:
    "https://rukminim2.flixcart.com/image/612/612/jeokbrk0/movie/z/6/2/2009-dvd-
excel-home-videos-english-avatar-original-imaf3bd6zed7phes.jpeg?q=70",
   genre: "Science",
 },
 {
   id: 6,
   title: "Avatar",
   image:
    "https://rukminim2.flixcart.com/image/612/612/jeokbrk0/movie/z/6/2/2009-dvd-
excel-home-videos-english-avatar-original-imaf3bd6zed7phes.jpeg?q=70",
   genre: "Science",
 },
 {
   id: 7,
   title: "Avatar",
   image:
    "https://rukminim2.flixcart.com/image/612/612/jeokbrk0/movie/z/6/2/2009-dvd-
excel-home-videos-english-avatar-original-imaf3bd6zed7phes.jpeg?q=70",
   genre: "Science",
 },
 {
   id: 8,
   title: "Avatar",
   image:
    "https://rukminim2.flixcart.com/image/612/612/jeokbrk0/movie/z/6/2/2009-dvd-
excel-home-videos-english-avatar-original-imaf3bd6zed7phes.jpeg?q=70",
   genre: "Science",
 },
];

export default movies;
```

Routes in React

In React, **Routes** are used for navigation in single-page applications (SPAs). They define how the app should render components based on the URL path. React Router, a popular library, provides tools for implementing routing functionality.

Key Concepts of React Routes

1. **<Routes>**:
   - It replaces the older <Switch> component in React Router v6.
   - Encapsulates <Route> components.
   - Renders the first <Route> that matches the URL.
2. **<Route>**:
   - Defines the path and the component to render for that path.
   - A path is matched using pattern matching.
3. **Dynamic Routes**:
   - Routes can accept dynamic segments (e.g., /user/:id).
4. **Nested Routes**:
   - Allows defining routes within other routes for a hierarchical structure.
5. **useNavigate**:
   - Enables programmatic navigation.

Installing React Router

```
npm install react-router-dom
```

Example of Routes in React

Directory Structure:

```
src/
├── App.js
├── components/
│   ├── Home.js
│   ├── About.js
│   ├── Contact.js
│   ├── User.js
├── index.js
```

1. App.js (Defining Routes)

```
import React from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Home from "./components/Home";
import About from "./components/About";
import Contact from "./components/Contact";
import User from "./components/User";

const App = () => {
  return (
    <Router>
      <div>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/contact" element={<Contact />} />
          <Route path="/user/:id" element={<User />} />
        </Routes>
      </div>
    </Router>
  );
};

export default App;
```

2. Home.js

```
import React from "react";
import { Link } from "react-router-dom";

const Home = () => {
  return (
    <div>
```

```
      <h1>Welcome to the Home Page</h1>
      <nav>
        <Link to="/about">About</Link> | <Link to="/contact">Contact</Link>
      </nav>
    </div>
  );
};

export default Home;
```

---

3. Dynamic Route Example (User.js)

```
import React from "react";
import { useParams } from "react-router-dom";

const User = () => {
  const { id } = useParams();

  return (
    <div>
      <h1>User Page</h1>
      <p>Welcome, user with ID: {id}</p>
    </div>
  );
};

export default User;
```

Features of Routes:

1. **Default Route**:
    - Example: The / path renders the Home component.

```
<Route path="/" element={<Home />} />
```

2. **Dynamic Routes**:
    - Use :parameter to define a dynamic segment.

```
<Route path="/user/:id" element={<User />} />
```

3. **Nested Routes**:
   - Define sub-routes inside a parent route.

```
<Route path="/dashboard" element={<Dashboard />}>
  <Route path="stats" element={<Stats />} />
  <Route path="settings" element={<Settings />} />
</Route>
```

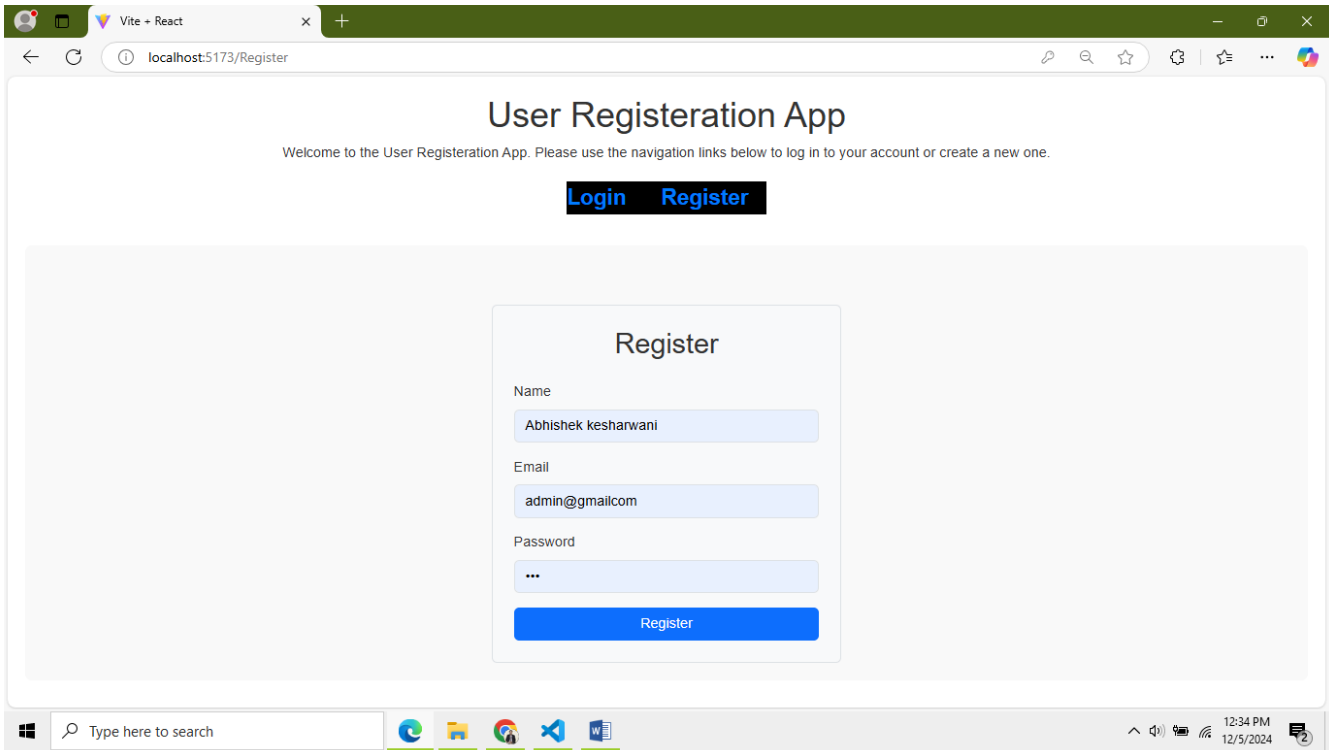
4. **Redirects**:
   - Programmatically navigate using useNavigate.

```
import { useNavigate } from "react-router-dom";

const Login = () => {
  const navigate = useNavigate();

  const handleLogin = () => {
    // Perform login logic
    navigate("/dashboard");
  };

  return <button onClick={handleLogin}>Login</button>;
};
```

Benefits of Routes in React:

- **Seamless Navigation**: Enables SPA-like navigation without reloading the page.
- **Dynamic Content**: Load components dynamically based on URL parameters.
- **Nested Routing**: Handle complex layouts with hierarchical navigation.

User Registration Project



User Registeration App

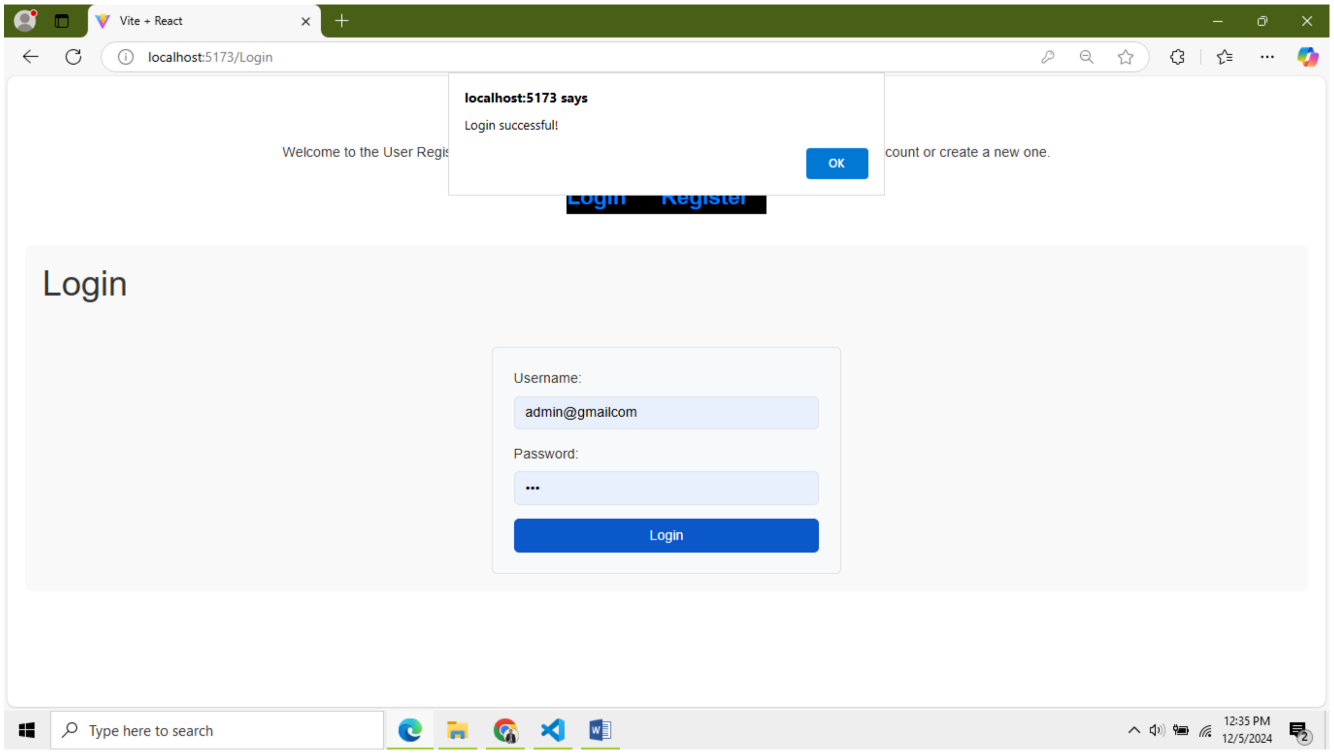
Welcome to the User Registeration App. Please use the navigation links below to log in to your account or create a new one.

Register

Name: Abhishek kesharwani

Email: admin@gmailcom

Password: •••



localhost:5173 says

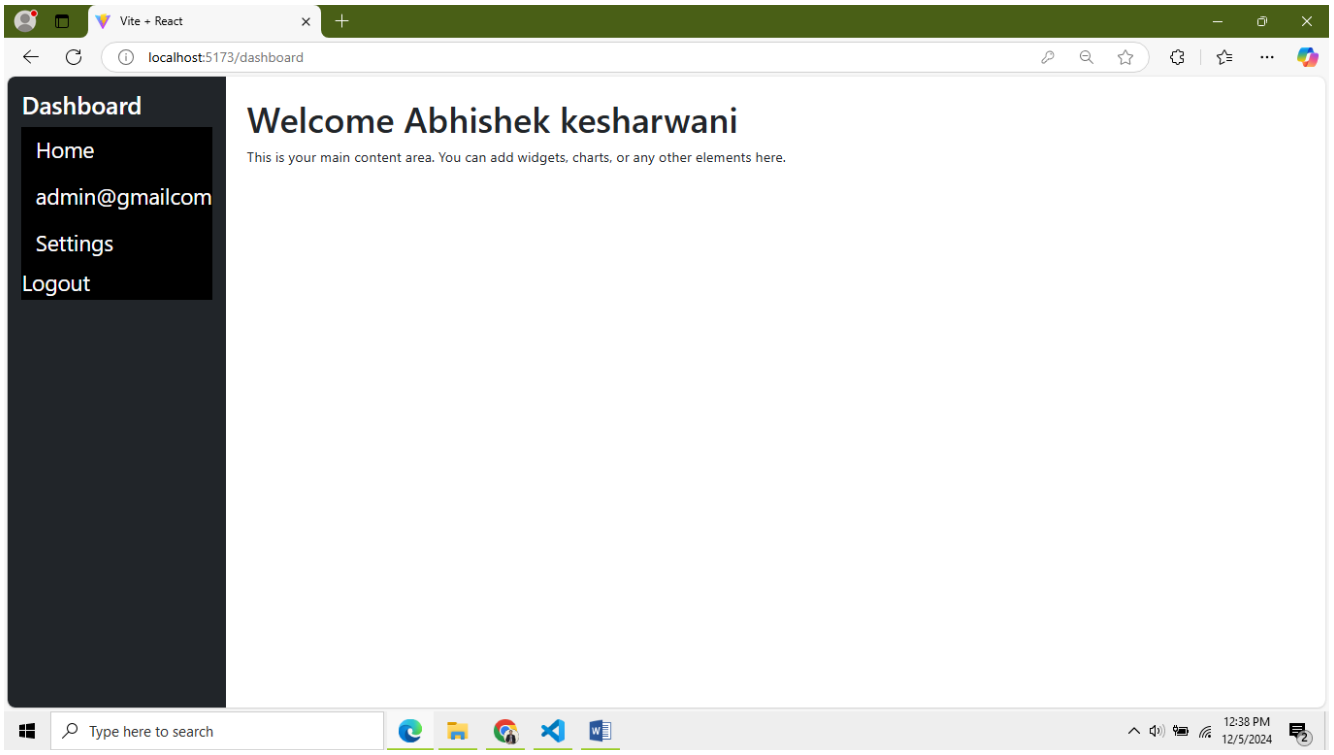
Login successful!

Welcome to the User Regis… …count or create a new one.

Login

Username: admin@gmailcom

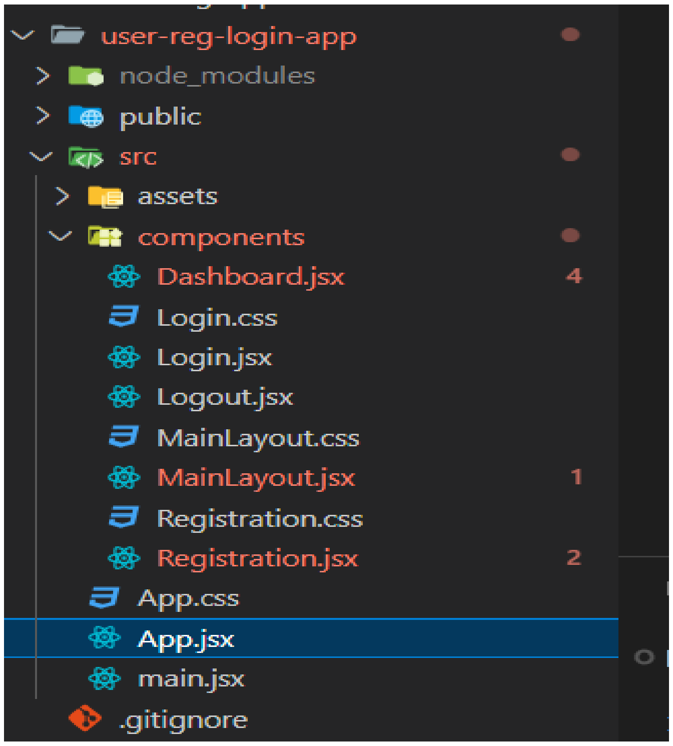
Password: •••



Welcome Abhishek kesharwani

This is your main content area. You can add widgets, charts, or any other elements here.

Project Structure



```
user-reg-login-app
  node_modules
  public
  src
    assets
    components
      Dashboard.jsx        4
      Login.css
      Login.jsx
      Logout.jsx
      MainLayout.css
      MainLayout.jsx       1
      Registration.css
      Registration.jsx     2
    App.css
    App.jsx
    main.jsx
  .gitignore
```

App.jsx

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import MainLayout from "./components/MainLayout";
import Login from "./components/Login";
import Registration from "./components/Registration";
import { useState } from "react";
import "./App.css";
import Dashboard from "./components/Dashboard";
import Logout from "./components/Logout";
const App = () => {
  const [data, setData] = useState();
  return (
    <div>
      {/* {JSON.stringify(data)} */}
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<MainLayout />}>
            <Route path="/login" element={<Login regDataLogin={data} />} />
            <Route
              path="/register"
              element={<Registration regData={setData} />}
            />
          </Route>
          <Route
            path="/dashboard"
            element={<Dashboard regDataDashboard={data} />}
          />
          <Route path="/logout" element={<Logout regDataLogout={setData} />} />
        </Routes>
      </BrowserRouter>
    </div>
  );
};
```

```
export default App;
```

MainLayout.jsx

```
import React from "react";
import { Link, Outlet } from "react-router-dom";
import "./MainLayout.css";

const MainLayout = () => {
  return (
    <div
      style={{
        fontFamily: "Arial, sans-serif",
        color: "#333",
        padding: "20px",
      }}
    >
      <header style={{ textAlign: "center", marginBottom: "20px" }}>
        <h1>User Registeration App</h1>
        <p>
          Welcome to the User Registeration App. Please use the navigation links
          below to log in to your account or create a new one.
        </p>
      </header>
      <nav style={{ marginBottom: "20px", textAlign: "center" }}>
        <ul
          style={{
            listStyle: "none",
            padding: 0,
            display: "inline-flex",
            gap: "20px",
          }}
        >
          <li>
            <Link to="/Login" style={linkStyle}>
              Login
            </Link>
          </li>
          <li>
            <Link to="/Register" style={linkStyle}>
              Register
            </Link>
          </li>
        </ul>
      </nav>
      <main
        style={{
          backgroundColor: "#f9f9f9",
          padding: "20px",
          borderRadius: "8px",
        }}
      >
        <Outlet />
      </main>
    </div>
  );
};

const linkStyle = {
  textDecoration: "none",
  color: "#007BFF",
  fontWeight: "bold",
};
```

```
export default MainLayout;
```

MainLayout.css

```
nav ul {
  background-color: black;
}
ul {
  display: flex;
  background-color: black;
  font-size: 25px;
  color: white;
  list-style-type: none;
}

ul li {
  margin-right: 20px;
  color: wheat;
}
li a {
  color: white;
  text-decoration: none;
}
```

Registration.jsx

```
import React, { useState } from "react";
import "./Registration.css";
const Registration = ({ regData }) => {
  const [name, setName] = useState();
  const [email, setEmail] = useState();
  const [password, setPassword] = useState();
  const handleregister = (e) => {
```

```
  e.preventDefault();
  const data = { name, email, password };
  regData(data);
};
return (
  <div
    className="container mt-5 p-4 border rounded bg-light"
    style={{ maxWidth: "400px" }}
  >
    <form>
      <h2 className="text-center mb-4">Register</h2>
      <div className="mb-3">
        <label htmlFor="name" className="form-label">
          Name
        </label>
        <input
          type="text"
          id="name"
          className="form-control"
          placeholder="Enter your name"
          value={name}
          onChange={(e) => setName(e.target.value)}
        />
      </div>

      <div className="mb-3">
        <label htmlFor="email" className="form-label">
          Email
        </label>
        <input
          type="email"
          id="email"
          className="form-control"
          placeholder="Enter your email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
        />
      </div>

      <div className="mb-3">
        <label htmlFor="password" className="form-label">
          Password
        </label>
        <input
          type="password"
          id="password"
          className="form-control"
          placeholder="Enter your password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />
      </div>

      <button
        type="submit"
        className="btn btn-primary w-100"
        onClick={handleregister}
      >
        Register
      </button>
    </form>
  </div>
 );
};

export default Registration;
```

Registration.css

```
.reg {
  background-color: black;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 20px;
}
.txt {
  color: blueviolet;
  font-size: 24px;
  text-align: center;

  border-radius: 8px;
  box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
.frm {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  width: 300px;
  padding: 10px;
  border-radius: 8px;
  border: 10px solid wheat;
  box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
```

Login.jsx

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import "bootstrap/dist/css/bootstrap.min.css";
const Login = ({ regDataLogin }) => {
  const [email, setEmail] = useState();
  const [password, setPassword] = useState();
  const navigate = useNavigate();
  const handlelogin = (e) => {
    e.preventDefault();
    if (regDataLogin.email === email && regDataLogin.password === password) {
      alert("Login successful!");
      navigate("/dashboard");
    } else {
      alert("Invalid credentials. Please try again.");
    }
  };
  return (
    <div className="lgn">
      <h1>Login</h1>
      <form
        className="container mt-5 p-4 border rounded bg-light"
        style={{ maxWidth: "400px" }}
      >
        <div className="mb-3">
          <label htmlFor="email" className="form-label">
            Username:
          </label>
          <input
            type="email"
            id="email"
            className="form-control"
            placeholder="Enter your email"
            onChange={(e) => setEmail(e.target.value)}
          />
        </div>
        <div className="mb-3">
          <label htmlFor="password" className="form-label">
            Password:
          </label>
          <input
            type="password"
            id="password"
            className="form-control"
            placeholder="Enter your password"
            onChange={(e) => setPassword(e.target.value)}
          />
        </div>

        <button
          type="submit"
          className="btn btn-primary w-100"
          onClick={handlelogin}
        >
          Login
        </button>
      </form>
    </div>
  );
};

export default Login;
```

Login.css

```
.lgn {
  display: flex;
  background-color: black;
}
```

DashBoard.jsx

```
import React from "react";
import "bootstrap/dist/css/bootstrap.min.css";
import { Link } from "react-router-dom";

const Dashboard = ({ regDataDashboard }) => {
  return (
    <div className="d-flex">
      {/* Sidebar */}
      <div className="bg-dark text-white p-3 vh-100" style={{ width: "250px" }}>
        <h3>Dashboard</h3>
        <ul className="nav flex-column">
          <li className="nav-item">
            <a href="#" className="nav-link text-white">
              Home
            </a>
          </li>
          <li className="nav-item">
            <a href="#" className="nav-link text-white">
              {regDataDashboard.email}
            </a>
          </li>
          <li className="nav-item">
            <a href="#" className="nav-link text-white">
              Settings
            </a>
          </li>
          <li className="nav-item">
            <Link to="/logout">Logout</Link>
          </li>
        </ul>
      </div>

      {/* Main Content */}
      <div className="p-4" style={{ flex: 1 }}>
        <h1>Welcome {regDataDashboard.name}</h1>
        <p>
          This is your main content area. You can add widgets, charts, or any
          other elements here.
        </p>
      </div>
    </div>
  );
};

export default Dashboard;
```

Logout.jsx

```
import React, { useEffect } from "react";
import { useNavigate } from "react-router-dom";

const Logout = ({ regDataLogout }) => {
  const navigate = useNavigate();

  useEffect(() => {
    regDataLogout(null);
    navigate("/");
  }, [navigate]);

  return <div>Logging out...</div>;
};

export default Logout;
```