**What is routing in react?**

**Routing** in React refers to the process of managing **navigation** and rendering different components based on the URL or user actions.

In web applications, routing allows users to navigate between different pages or views **without reloading the entire page.**

React doesn't come with built-in routing capabilities, so developers often use third-party libraries like **React Router** to handle routing in their applications.

React Router is a popular library for handling routing in React applications.

It provides a **<BrowserRouter>** component that wraps your application and enables declarative routing using **<Route>** components.

These **<Route>** components define which component should be rendered based on the current URL path.

**How to install React Router?**

**npm install react@19 react-dom@19 react-router-dom@6**

**To see which version was installed**

**npm list react-router-dom**

**App.js**

import React from "react";

import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

// Example pages

const Home = () => <h2>Home Page</h2>;

const About = () => <h2>About Page</h2>;

const Contact = () => <h2>Contact Page</h2>;

const App = () => {

return (

<Router>

<div>

<nav>

<ul>

<li><Link to="/">Home</Link></li>

<li><Link to="/about">About</Link></li>

<li><Link to="/contact">Contact</Link></li>

</ul>

</nav>

<Routes>

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

<Route path="/about" element={<About />} />

<Route path="/contact" element={<Contact />} />

</Routes>

</div>

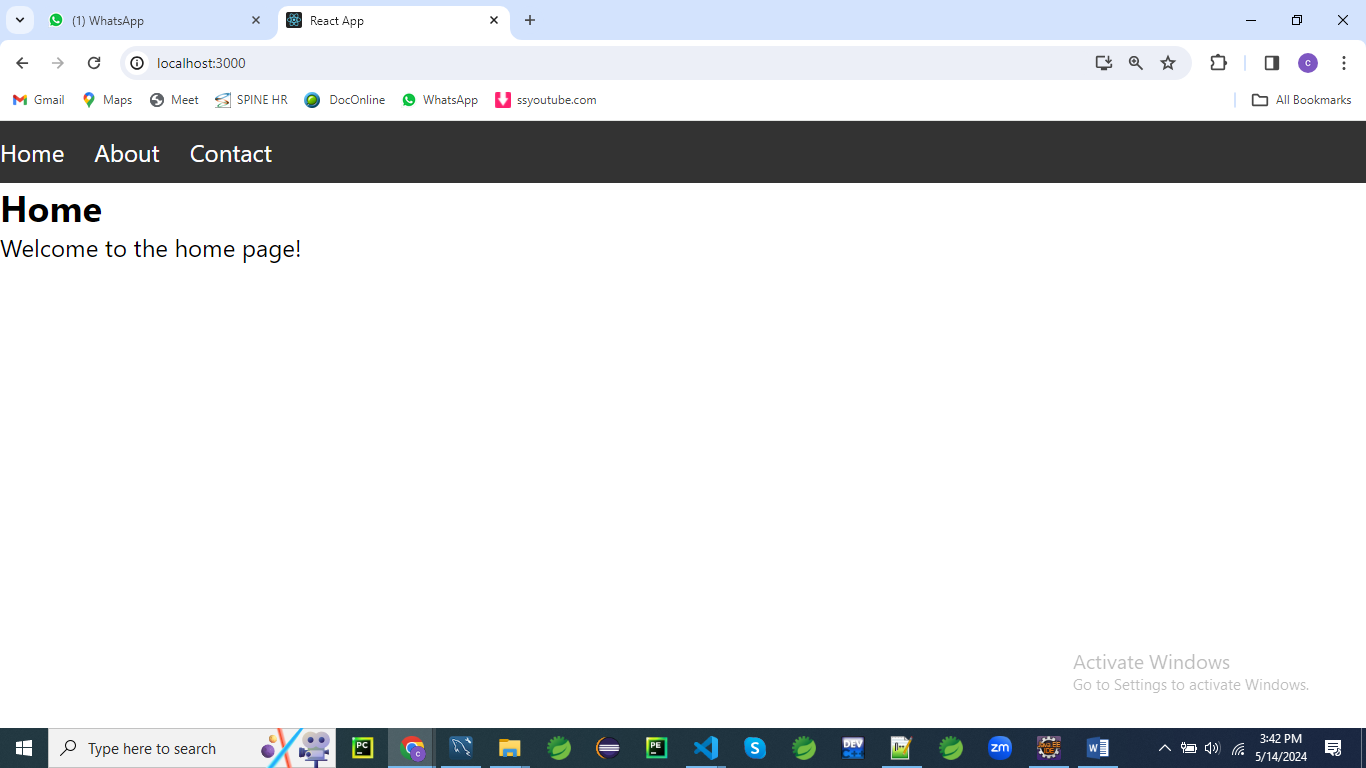
</Router>

);

};

export default App;

**Example 01**



First, make sure you have React and React Router installed in your project as described earlier.

Then, you can create the following file structure:

/src

/components

Home.js

About.js

Contact.js

App.js

Create a folder named **components** under **src** folder

**Home.js**

// components/Home.js

import React from 'react';

const Home = () => {

return (

<div>

<h2>Home</h2>

<p>Welcome to the home page!</p>

</div>

);

};

export default Home;

**About.js**

// components/About.js

import React from 'react';

const About = () => {

return (

<div>

<h2>About</h2>

<p>About us page content goes here...</p>

</div>

);

};

export default About;

**Contact.js**

// components/Contact.js

import React from 'react';

const Contact = () => {

return (

<div>

<h2>Contact</h2>

<p>Contact us page content goes here...</p>

</div>

);

};

export default Contact;

styles.css

/\* styles.css \*/

/\* Reset default browser styles \*/

body, h1, h2, p, ul, li {

margin: 0;

padding: 0;

}

/\* Style the navigation bar \*/

nav {

background-color: #333;

color: #fff;

padding: 10px 0;

}

nav ul {

list-style-type: none;

}

nav ul li {

display: inline;

margin-right: 20px;

}

nav ul li a {

color: #fff;

text-decoration: none;

}

nav ul li a:hover {

text-decoration: underline;

}

/\* Style the content container \*/

.container {

max-width: 800px;

margin: 20px auto;

padding: 0 20px;

}

/\* Style the header \*/

header {

text-align: center;

margin-bottom: 20px;

}

header h1 {

font-size: 2em;

margin-bottom: 10px;

}

/\* Style the main content \*/

.main-content {

background-color: #f9f9f9;

padding: 20px;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

Then, in your **App.js** file, you can set up routing using React Router:

// App.js

import React from 'react';

import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';

import Home from './components/Home';

import About from './components/About';

import Contact from './components/Contact';

import './styles.css'; // Import the CSS file

const App = () => {

  return (

    <Router>

      <div>

        <nav>

          <ul>

            <li>

              <Link to="/">Home</Link>

            </li>

            <li>

              <Link to="/about">About</Link>

            </li>

            <li>

              <Link to="/contact">Contact</Link>

            </li>

          </ul>

        </nav>

        <div className="container">

          <header>

            <h1>My React App</h1>

          </header>

          <div className="main-content">

            <Routes>

            <Route exact path="/" component={<Home/>} />

            <Route path="/about" component={<About/>} />

            <Route path="/contact" component={<Contact/>} />

            </Routes>

          </div>

        </div>

      </div>

    </Router>

  );

};

export default App;

**Explanation:**

**import React from 'react';**

**import { BrowserRouter as Router, Route, Link } from 'react-router-dom';**

**import Home from './components/Home';**

**import About from './components/About';**

**import Contact from './components/Contact';**

**import './styles.css'; // Import the CSS file**

In this section, we're importing necessary modules and components for our React application.

We're importing **React** from the 'react' library and components such as **BrowserRouter**, **Route**, and **Link** from the 'react-router-dom' library.

Additionally, we're importing three custom components (**Home**, **About**, and **Contact**) from their respective files, and we're importing a CSS file named **styles.css** for styling.

**const App = () => {**

**return (**

**<Router>**

**<div>**

**<nav>**

**<ul>**

**<li>**

**<Link to="/">Home</Link>**

**</li>**

**<li>**

**<Link to="/about">About</Link>**

**</li>**

**<li>**

**<Link to="/contact">Contact</Link>**

**</li>**

**</ul>**

**</nav>**

**<div className="container">**

**<header>**

**<h1>My React App</h1>**

**</header>**

**<div className="main-content">**

**<Route exact path="/" component={<Home/>} />**

**<Route path="/about" component={<About/>} />**

**<Route path="/contact" component={<Contact/>} />**

**</div>**

**</div>**

**</div>**

**</Router>**

**);**

**};**

In the **App** component, we're defining the structure of our React application.

* We wrap our entire application with the **<Router>** component to enable routing functionality.
* Inside the router, we have a **<div>** element containing the navigation bar (**<nav>**) and main content area (**<div className="container">**).
* The navigation bar contains an unordered list (**<ul>**) with list items (**<li>**), each containing a **<Link>** component from React Router. These links navigate to different routes specified by the **to** prop.
* The main content area (**<div className="container">**) contains a header (**<header>**) with an **<h1>** element displaying the title of the application.
* Below the header, we have a **<div>** element with the class name **main-content**. This is where the content for each route will be rendered.
* Inside this **<div>**, we have three **<Route>** components. Each **<Route>** component specifies a path and the component to render when that path matches the current URL. The **exact** prop is used for the home route to ensure that it matches exactly the root URL (**/**).

export default App;

Finally, we export the **App** component so it can be imported and used in other parts of our application.

In summary, this code sets up a basic React application with routing functionality using React Router.

It consists of a navigation bar and main content area, with different components rendered based on the current URL path.

The application structure is defined in the **App** component, which is exported as the default component.

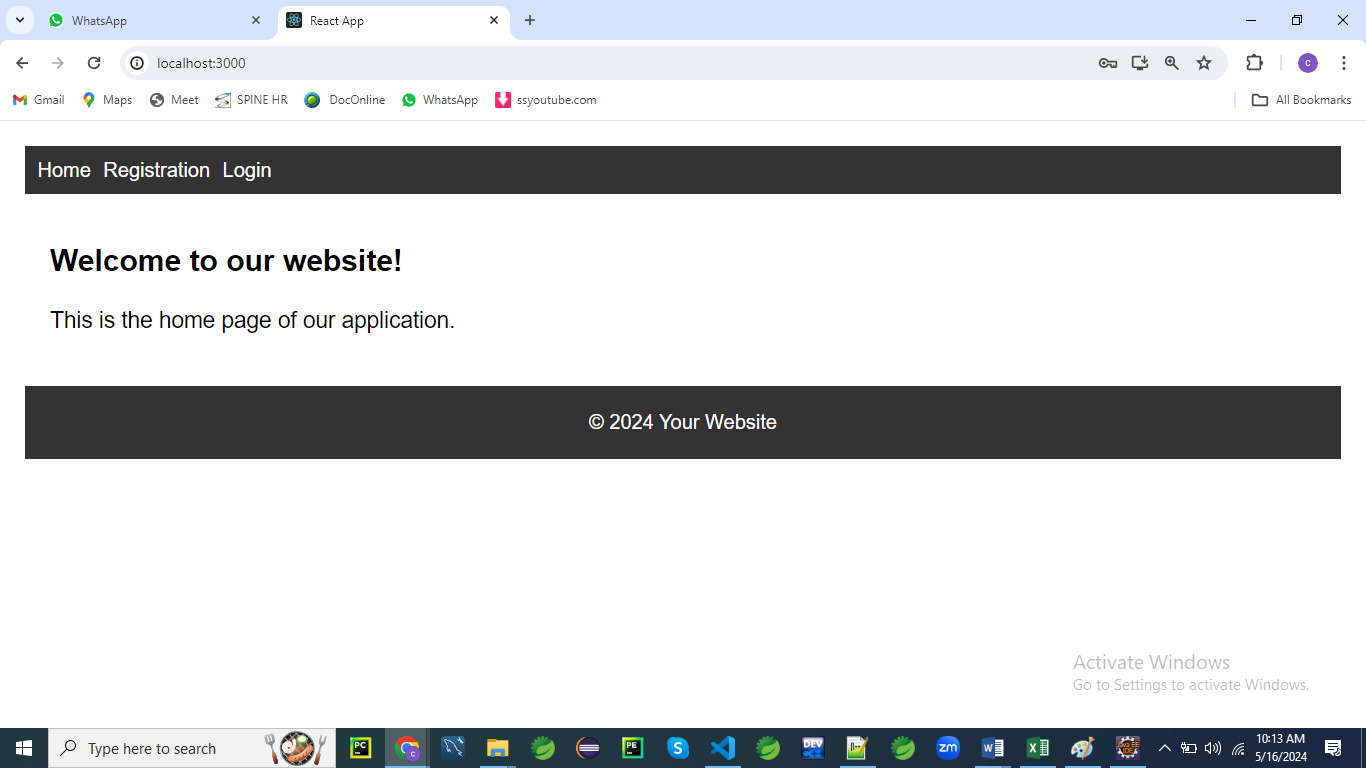
**Example 02**

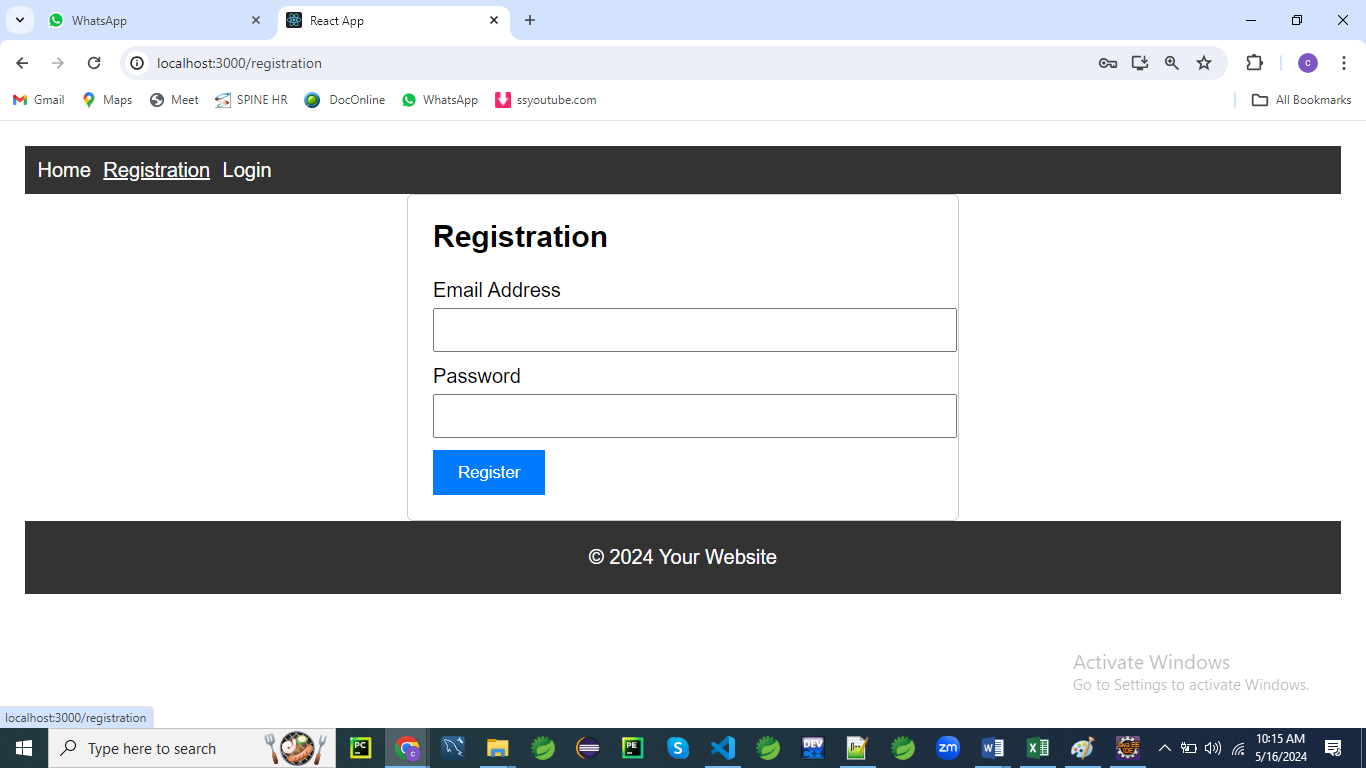
Let's create a React application with multiple routes, navigation, form validation, and CSS styling.

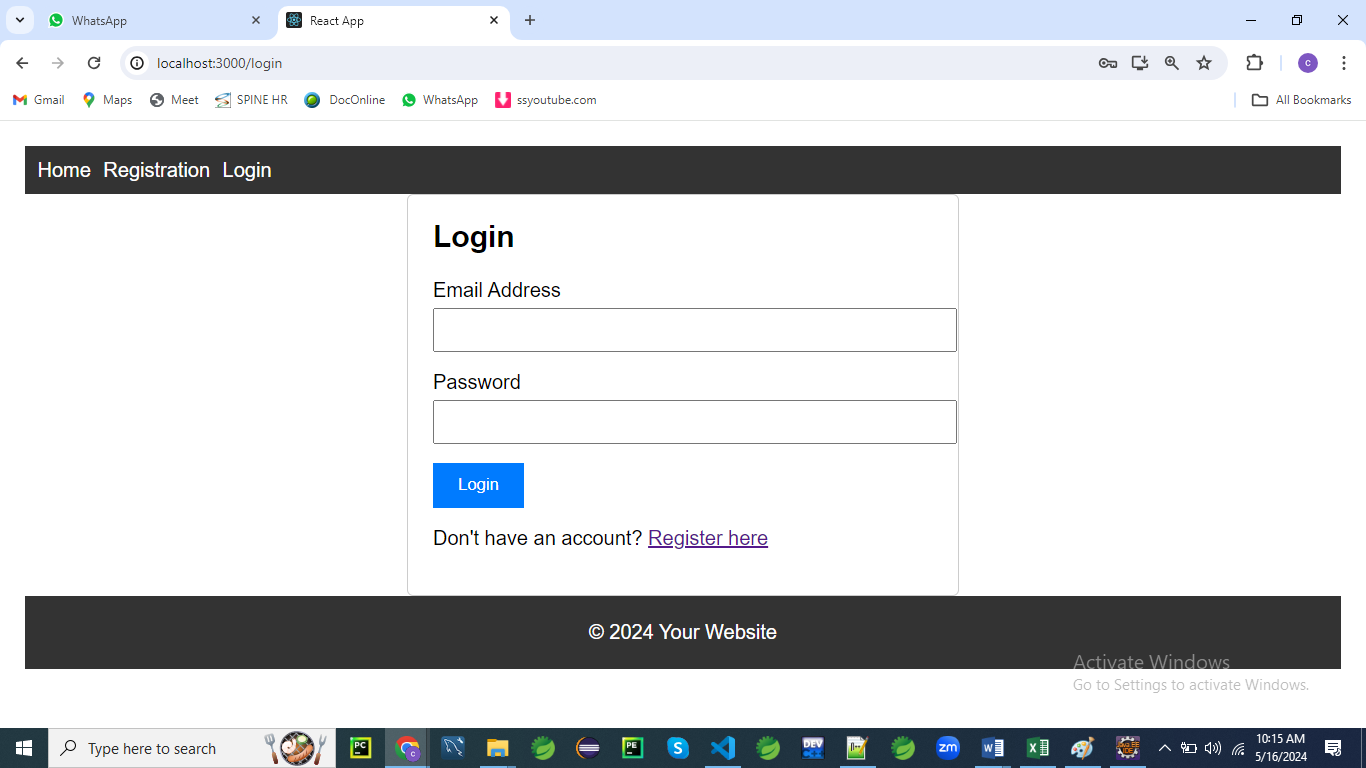
We'll build a simple registration form with the following features:

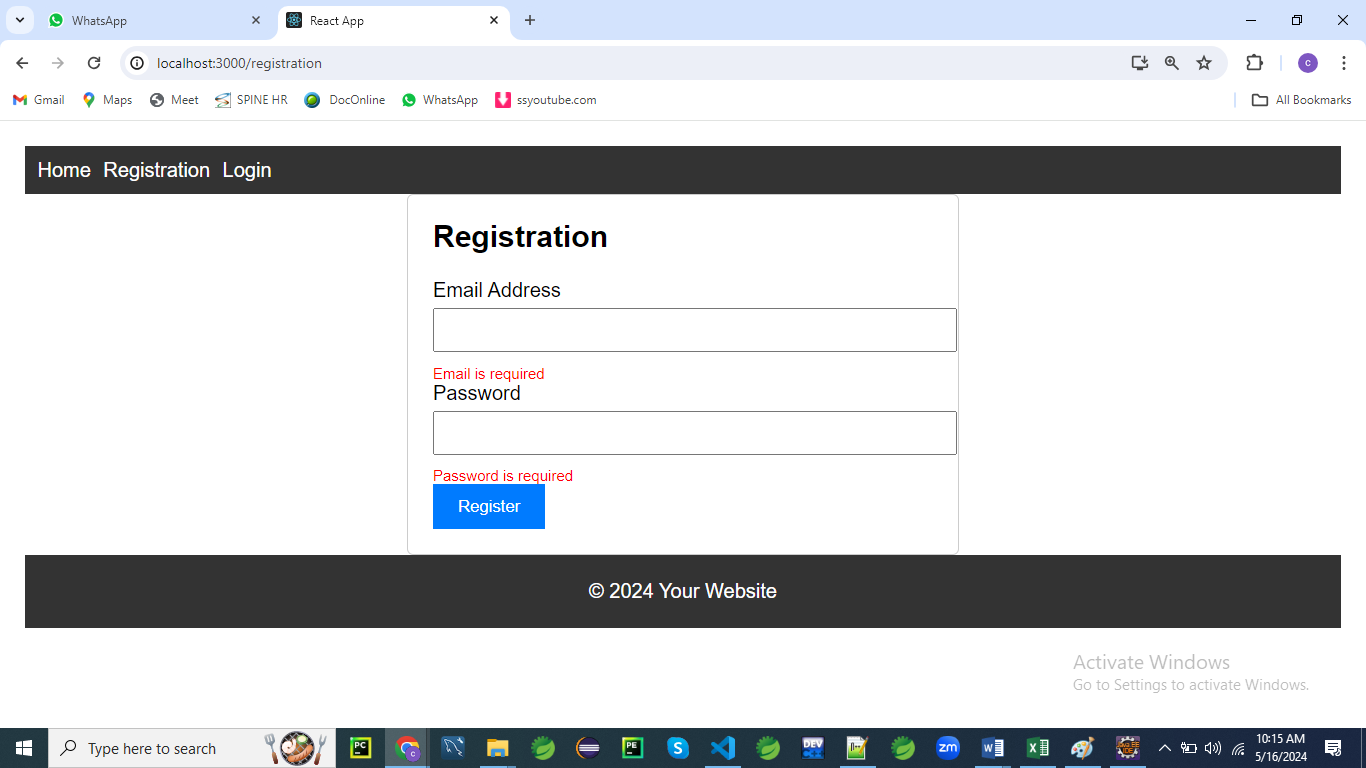
1. **Home Page**: Displaying a welcome message.
2. **Registration Page**: A form for users to register with validation for email and password fields.
3. **Login Page**: A form for users to login.

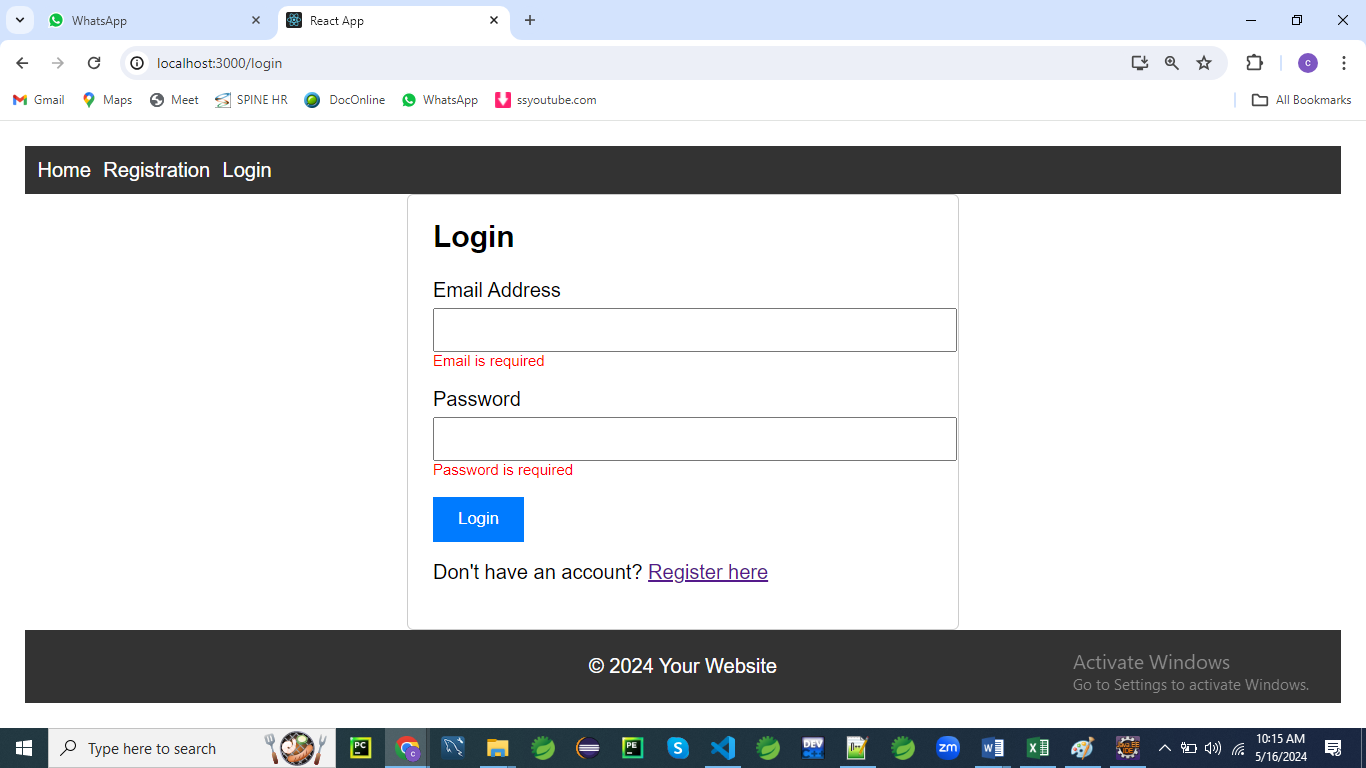
Define routes for your application using the **BrowserRouter**, **Route** components provided by React Router.

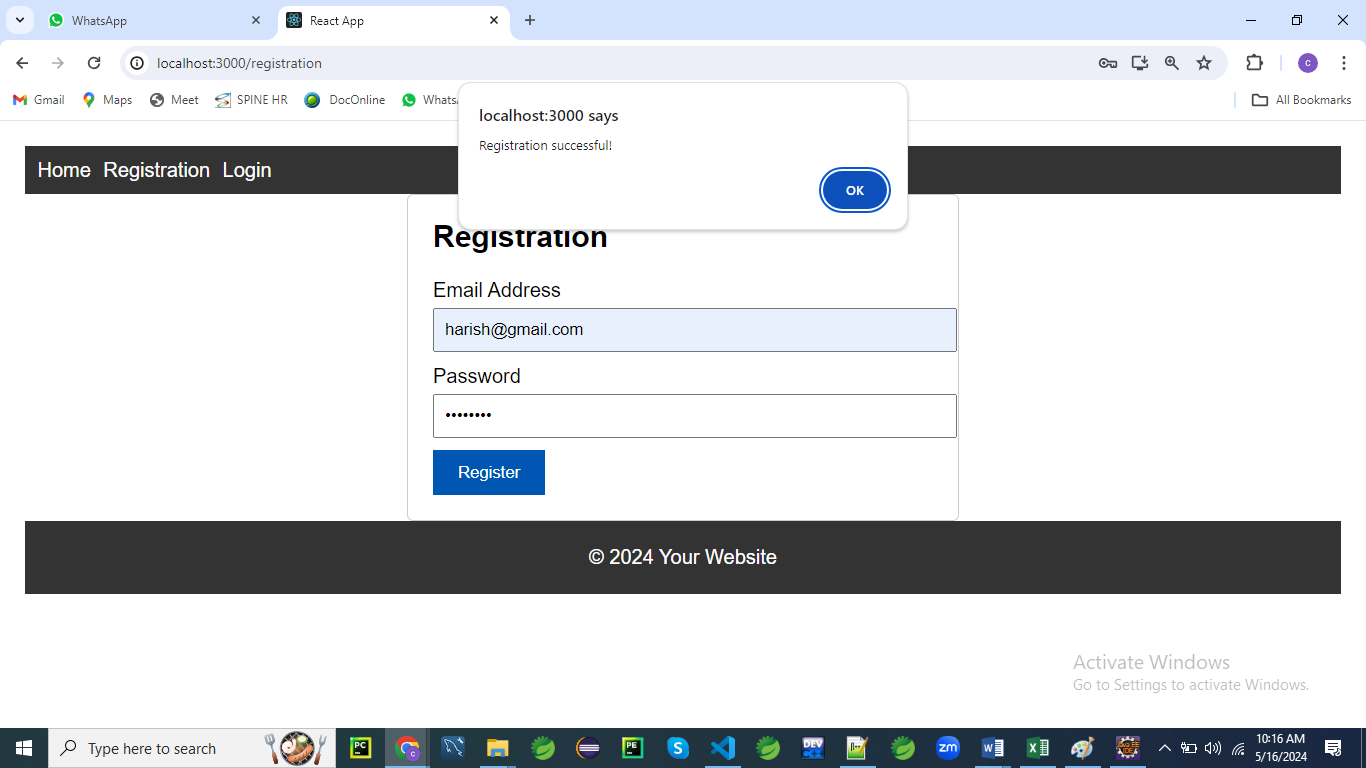


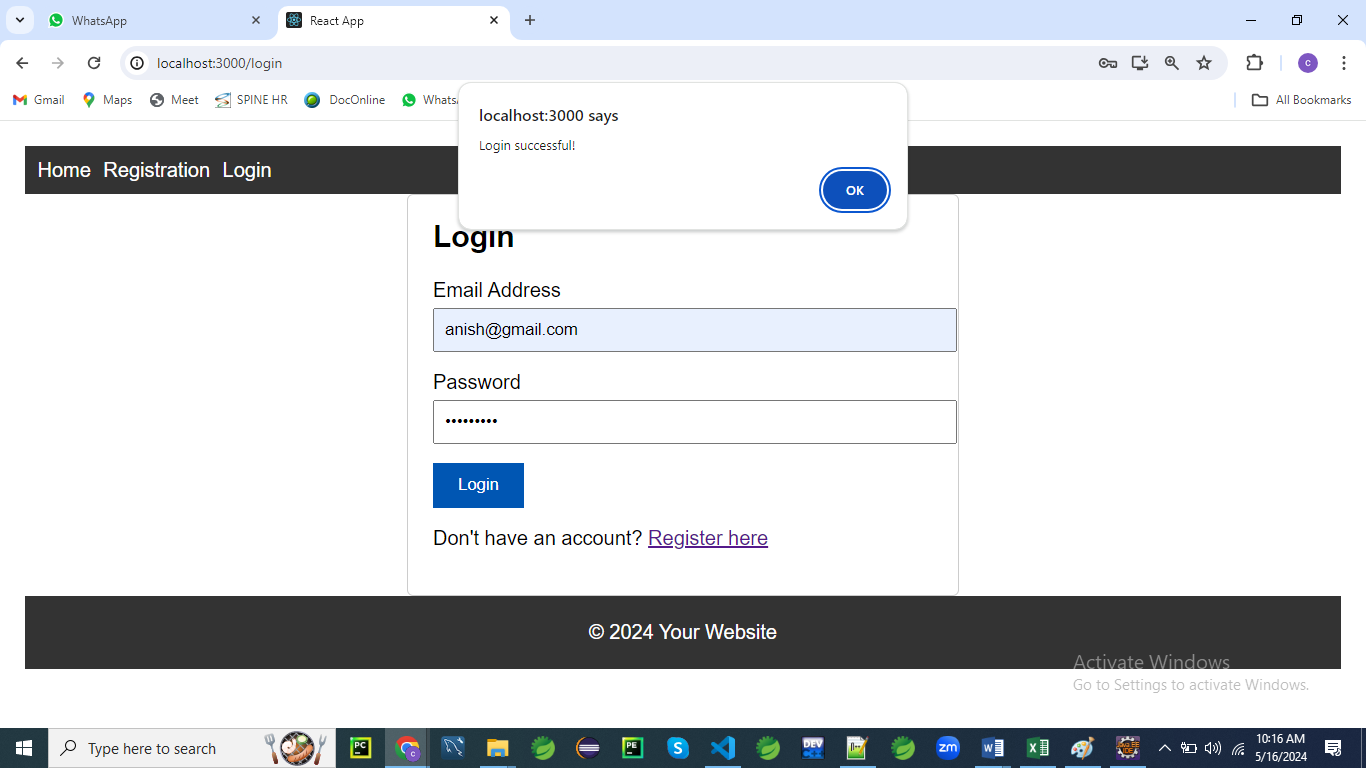












/src

/components

Home.js

Home.css

Registration.js

Registration.css

Login.js

Login.css

Navigation.js

Navigation.css

App.js

App.css

App.js

// App.js

import React from 'react';

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

import Home from './components/Home';

import Registration from './components/Registration';

import Login from './components/Login';

import Navigation from './components/Navigation';

import './App.css'; // Import the global CSS file

const App = () => {

  return (

    <Router>

      <div className="container">

        <Navigation />

        <Routes>

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

          <Route path="/registration" element={<Registration />} />

          <Route path="/Login" element={<Login />} />

        </Routes>

        <footer>

          &copy; 2024 Your Website

        </footer>

      </div>

    </Router>

  );

};

export default App;

Here’s an explanation of the provided code snippet:

1. **Imports**:
   * **import React from 'react';:** Imports the React library, which is necessary for using JSX and creating React components.
   * **import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';:** Imports routing components from react-router-dom for handling navigation within the application.
   * **import Home from './components/Home'**;: Imports the Home component, which will be displayed at the root route.
   * **import Registration from './components/Registration';:** Imports the Registration component, used for the registration route.
   * **import Login from './components/Login';:** Imports the Login component, used for the login route.
   * **import Navigation from './components/Navigation';:** Imports the Navigation component, which provides navigation links or a menu.
   * **import './App.css';:** Imports a CSS file to apply global styles to the application.
2. **Functional Component**: **const App = () => { ... }** defines a functional component named App. This is the main component that sets up the routing and layout for the application.
3. **Router Setup**:
   * **<Router>:** Wraps the entire application to provide routing functionality.
   * Inside <Router>, the **<div className="container">** element acts as a wrapper for layout and styling.
4. **Navigation and Routing**:
   * **<Navigation />:** Renders the navigation component, typically containing links to navigate between different routes.
   * **<Routes>:** Defines the different routes in the application.
     + **<Route path="/" element={<Home />} />:** Maps the root path / to the Home component.
     + **<Route path="/registration" element={<Registration />} />:** Maps the /registration path to the Registration component.
     + **<Route path="/Login" element={<Login />} />:** Maps the /Login path to the Login component. Note that the path should generally be in lowercase (/login) for consistency.
5. **Footer**:
   * **<footer>:** Adds a footer to the bottom of the page with the text “© 2024 Your Website”. This is used for copyright information or other footer content.

This setup provides a basic structure for a React application with routing, a navigation bar, and global styling.

/\* App.css \*/

/\* Global styles \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

.container {

max-width: 1200px;

margin: 0 auto;

padding: 20px;

}

/\* Navigation styles \*/

nav {

background-color: #333;

color: #fff;

padding: 10px;

}

nav ul {

list-style-type: none;

margin: 0;

padding: 0;

}

nav ul li {

display: inline;

margin-right: 10px;

}

nav ul li a {

color: #fff;

text-decoration: none;

}

/\* Footer styles \*/

footer {

background-color: #333;

color: #fff;

text-align: center;

padding: 20px 0;

}

**[**

**useState() in React**

The useState() hook in React allows functional components to **manage state**. It provides a state variable and a function to update that state.

**1. Basic Example**

**Import useState and Use It in a Component**

import React, { useState } from "react";

function Counter() {

const [count, setCount] = useState(0); // Initial state is 0

return (

<div>

<h2>Count: {count}</h2>

<button onClick={() => setCount(count + 1)}>Increase</button>

<button onClick={() => setCount(count - 1)}>Decrease</button>

</div>

);

}

export default Counter;

**Explanation:**

* useState(0) initializes count with 0.
* setCount(count + 1) updates the state.
* The component **re-renders** whenever the state changes.

**2. Using useState() with Strings**

import React, { useState } from "react";

function Username() {

const [name, setName] = useState("Guest");

return (

<div>

<h2>Welcome, {name}</h2>

<input

type="text"

placeholder="Enter your name"

onChange={(e) => setName(e.target.value)}

/>

</div>

);

}

export default Username;

**How It Works:**

* The initial state is "Guest".
* When the user types in the input field, setName() updates the state.

**3. Using useState() with Objects**

If you want to manage multiple values, store them in an **object**:

import React, { useState } from "react";

function UserProfile() {

const [user, setUser] = useState({ name: "", age: "" });

return (

<div>

<input

type="text"

placeholder="Name"

value={user.name}

onChange={(e) => setUser({ ...user, name: e.target.value })}

/>

<input

type="number"

placeholder="Age"

value={user.age}

onChange={(e) => setUser({ ...user, age: e.target.value })}

/>

<h2>

Name: {user.name}, Age: {user.age}

</h2>

</div>

);

}

export default UserProfile;

**Key Points:**

* useState({ name: "", age: "" }) initializes state with an object.
* setUser({ ...user, name: e.target.value }) updates only one property while keeping the rest.

**4. Using useState() with Arrays**

If managing **lists**, store them in an array:

import React, { useState } from "react";

function TodoList() {

const [tasks, setTasks] = useState([]);

const addTask = () => {

setTasks([...tasks, `Task ${tasks.length + 1}`]);

};

return (

<div>

<button onClick={addTask}>Add Task</button>

<ul>

{tasks.map((task, index) => (

<li key={index}>{task}</li>

))}

</ul>

</div>

);

}

export default TodoList;

**How It Works:**

* useState([]) initializes an empty array.
* setTasks([...tasks, newTask]) adds a new task without losing the previous ones.

**5. Functional Updates in useState()**

For state updates based on the **previous state**, use a callback function:

const [count, setCount] = useState(0);

const increment = () => {

setCount(prevCount => prevCount + 1);

};

This ensures React always gets the latest state.

**Common Issues & Solutions**

| **Issue** | **Solution** |
| --- | --- |
| State doesn't update immediately | React updates state asynchronously. Use console.log inside useEffect() to debug. |
| Losing previous object properties | Use the spread operator (...prevState) to retain previous values. |
| Updating arrays directly | Instead of tasks.push(newTask), use setTasks([...tasks, newTask]). |

**Final Notes**

* useState() **only works inside functional components**.
* State updates **cause re-renders**.
* Always **use the setter function** (setState) instead of modifying state directly.

**]**

**[**

The useNavigate() hook is used in **React Router**  to navigate programmatically within your application.

It replaces the older useHistory() from React Router v5.

**How to Use useNavigate()**

**1. Import useNavigate**

First, ensure that you import it from react-router-dom:

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

**2. Basic Example**

You can use useNavigate() inside a functional component:

import React from "react";

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

function Home() {

const navigate = useNavigate();

const goToAbout = () => {

navigate("/about"); // Navigates to the About page

};

return (

<div>

<h1>Home Page</h1>

<button onClick={goToAbout}>Go to About</button>

</div>

);

}

export default Home;

**3. Navigate with Parameters**

You can pass **dynamic parameters** using navigate():

const navigate = useNavigate();

navigate(`/profile/${userId}`);

**Example:**

navigate("/profile/123");

This navigates to /profile/123.

**4. Navigate with State**

You can also send **data** (state) to the next route:

navigate("/dashboard", { state: { user: "John Doe" } });

To **retrieve** the state in the new component:

import { useLocation } from "react-router-dom";

function Dashboard() {

const location = useLocation();

console.log(location.state); // { user: "John Doe" }

return <h1>Welcome, {location.state?.user}</h1>;

}

**5. Navigate Back and Forward**

You can **go back** or **forward** like this:

navigate(-1); // Go back one step

navigate(1); // Go forward one step

**Make Sure You Have BrowserRouter**

For useNavigate() to work, your app must be wrapped in <BrowserRouter>:

import { BrowserRouter, Routes, Route } from "react-router-dom";

import Home from "./Home";

import About from "./About";

function App() {

return (

<BrowserRouter>

<Routes>

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

<Route path="/about" element={<About />} />

</Routes>

</BrowserRouter>

);

}

export default App;

**If useNavigate Is Not Working**

1. Ensure **react-router-dom is installed**:

npm install react-router-dom

1. Make sure your component is **inside <BrowserRouter>**.
2. **Check for incorrect paths** in your navigate() calls.
3. Restart your development server if the issue persists:

npm start

**]**

**[**

The **useLocation**() hook in **React Router** is used to access the current location object, which contains details about the URL, including the pathname, search parameters, and state.

**How to Use useLocation()**

**1. Import useLocation**

First, ensure that you import it from react-router-dom:

import { useLocation } from "react-router-dom";

**2. Basic Example**

You can use useLocation() inside a functional component:

import React from "react";

import { useLocation } from "react-router-dom";

function CurrentPage() {

const location = useLocation();

return (

<div>

<h2>Current Path: {location.pathname}</h2>

</div>

);

}

export default CurrentPage;

* If the URL is http://localhost:3000/about, it will display:

Current Path: /about

**3. Accessing Search Query Parameters**

You can extract **query parameters** from location.search:

import { useLocation } from "react-router-dom";

function QueryParams() {

const location = useLocation();

const params = new URLSearchParams(location.search);

const name = params.get("name");

return <h2>User Name: {name}</h2>;

}

export default QueryParams;

* If the URL is http://localhost:3000/profile?name=John, it will display:

User Name: John

**4. Accessing State Data Passed from useNavigate()**

If you navigate with useNavigate() and pass **state**, you can retrieve it using useLocation().

**Sending State Data**

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

function Home() {

const navigate = useNavigate();

return (

<button onClick={() => navigate("/dashboard", { state: { user: "John Doe" } })}>

Go to Dashboard

</button>

);

}

export default Home;

**Receiving State Data**

import { useLocation } from "react-router-dom";

function Dashboard() {

const location = useLocation();

const user = location.state?.user || "Guest";

return <h2>Welcome, {user}</h2>;

}

export default Dashboard;

* If the user clicks the button, they will be taken to /dashboard, and the page will display:

Welcome, John Doe

**5. Example with useEffect() to Track Route Changes**

If you need to track route changes, you can use useEffect():

import { useLocation } from "react-router-dom";

import { useEffect } from "react";

function RouteTracker() {

const location = useLocation();

useEffect(() => {

console.log("Current route:", location.pathname);

}, [location]);

return <h2>Current Route: {location.pathname}</h2>;

}

export default RouteTracker;

Every time the route changes, it logs the new pathname.

**Make Sure You Have BrowserRouter**

For useLocation() to work, your app **must** be wrapped in <BrowserRouter>:

import { BrowserRouter, Routes, Route } from "react-router-dom";

import Home from "./Home";

import Dashboard from "./Dashboard";

function App() {

return (

<BrowserRouter>

<Routes>

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

<Route path="/dashboard" element={<Dashboard />} />

</Routes>

</BrowserRouter>

);

}

export default App;

**If useLocation() Is Not Working**

1. **Ensure react-router-dom is installed**:

npm install react-router-dom

1. **Wrap your app in <BrowserRouter>**.
2. **Check for incorrect paths** when navigating.
3. **Restart your dev server** if issues persist:

npm start

**]**

Create components for each page of your application.

// Registration.js

// Registration.js

import React, { useState } from 'react';

import { Navigate, useNavigate  } from 'react-router-dom';

import './Registration.css'; // Import CSS file

const Registration = () => {

  const navigate = useNavigate();

  const [formData, setFormData] = useState({

    email: '',

    password: '',

  });

  const [errors, setErrors] = useState({});

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData({ ...formData, [name]: value });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    const validationErrors = validateForm(formData);

    if (Object.keys(validationErrors).length === 0) {

      // Form is valid, proceed with registration

      alert('Registration successful!');

      navigate('/'); // Redirect to home page

    } else {

      // Form is invalid, display errors

      setErrors(validationErrors);

    }

  };

  const validateForm = (data) => {

    const errors = {};

    // Basic email validation

    if (!data.email) {

      errors.email = 'Email is required';

    } else if (!isValidEmail(data.email)) {

      errors.email = 'Invalid email address';

    }

    // Password validation

    if (!data.password) {

      errors.password = 'Password is required';

    } else if (data.password.length < 8) {

      errors.password = 'Password must be at least 8 characters long';

    }

    return errors;

  };

  const isValidEmail = (email) => {

    // Basic email validation regex

    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    return regex.test(email);

  };

  return (

    <div className="form-container">

      <h2>Registration</h2>

      <form onSubmit={handleSubmit}>

        <div>

          <label htmlFor="email">Email Address</label>

          <input type="email" id="email" name="email" value={formData.email} onChange={handleChange} />

          {errors.email && <div className="error">{errors.email}</div>}

        </div>

        <div>

          <label htmlFor="password">Password</label>

          <input type="password" id="password" name="password" value={formData.password} onChange={handleChange} />

          {errors.password && <div className="error">{errors.password}</div>}

        </div>

        <button type="submit">Register</button>

      </form>

    </div>

  );

};

export default Registration;

Here’s an explanation of the provided Registration component code in 5 points:

1. **Imports**:
   * **import React, { useState } from 'react';:** Imports React and the useState hook to manage component state.
   * **import { Navigate, useNavigate } from 'react-router-dom';:** Imports navigation-related functions for handling routing. useNavigate is used here to programmatically navigate to different routes.
   * **import './Registration.css';:** Imports a CSS file for styling the registration form.
2. **State Management**:
   * **const [formData, setFormData] = useState({ email: '', password: '' });:** Initializes state to store form input values (email and password).
   * **const [errors, setErrors] = useState({});:** Initializes state to store validation error messages.
3. **Event Handlers**:
   * handleChange: Updates the formData state when the user types into form fields. It extracts the name and value from the event target and updates the corresponding field in the formData state.
   * **handleSubmit**: Prevents the default form submission, validates the form data, and either displays errors or shows a success message and redirects to the home page if the form is valid.
4. **Validation Functions**:
   * **validateForm**: Validates the form data and returns an object with error messages if any validation rules are not met. It checks for required fields and validates the email format and password length.
   * **isValidEmail:** Uses a regular expression to validate the email format. Returns true if the email matches the pattern and false otherwise.
5. **JSX Rendering**:
   * The return statement renders the registration form inside a div with the class form-container.
   * Includes <h2>Registration</h2> for the form title.
   * The form fields for email and password are accompanied by labels and conditional error messages.
   * The button with type="submit" triggers the handleSubmit function when clicked, submitting the form.

This component manages form data, validates input, and provides feedback to the user. It also handles navigation using useNavigate to redirect users upon successful registration.

**Registration.css file**

/\* Registration.css \*/

.form-container {

max-width: 400px;

margin: 0 auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

}

.form-container h2 {

margin-top: 0;

}

.form-container label {

display: block;

margin-bottom: 5px;

}

.form-container input[type='email'],

.form-container input[type='password'] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

}

.form-container div.error {

color: red;

font-size: 12px;

}

.form-container button {

background-color: #007bff;

color: #fff;

border: none;

padding: 10px 20px;

cursor: pointer;

}

.form-container button:hover {

background-color: #0056b3;

}

Home.css

/\* Home.css \*/

.container {

max-width: 800px;

margin: 0 auto;

padding: 20px;

}

.title {

font-size: 24px;

margin-bottom: 20px;

}

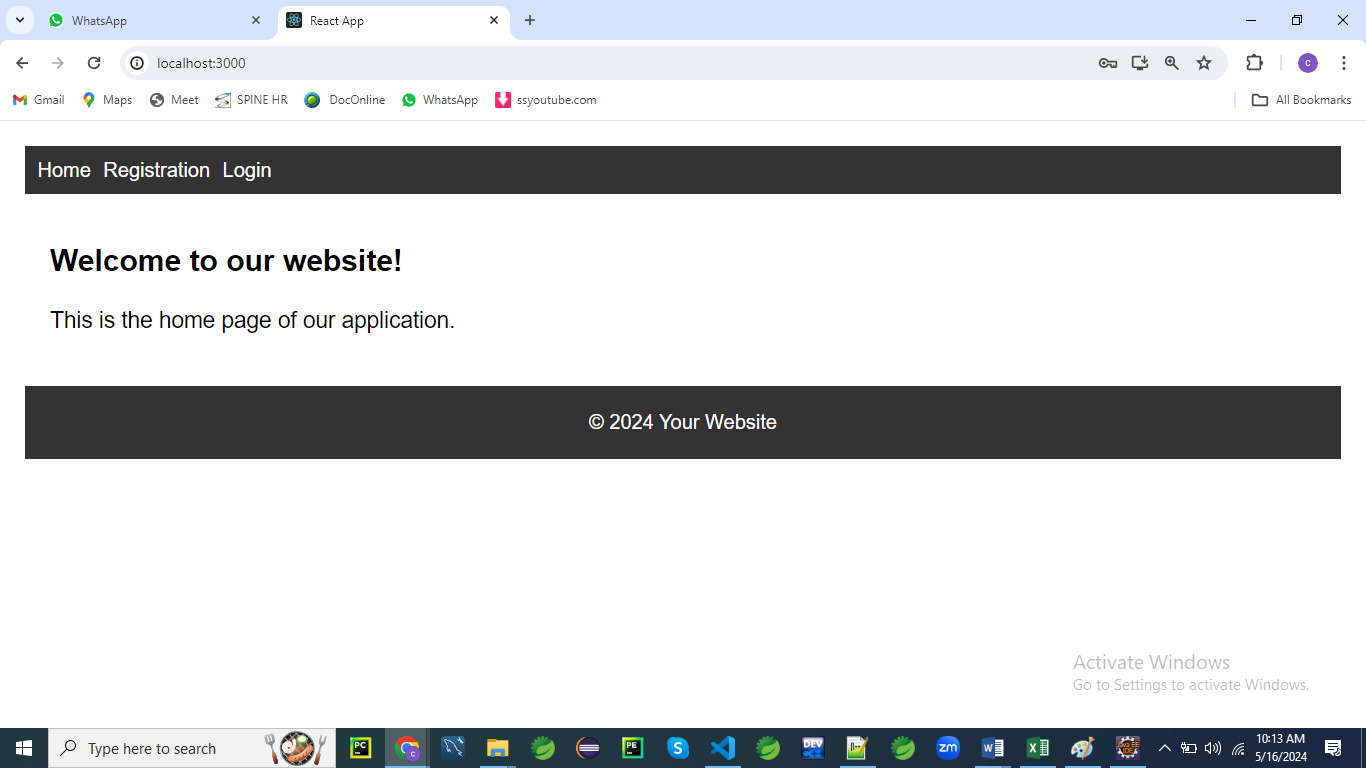
.content {

font-size: 18px;

line-height: 1.6;

}

Home.js



// Home.js

import React from 'react';

import './Home.css'; // Import the CSS file

const Home = () => {

  return (

    <div className="container">

      <h2 className="title">Welcome to our website!</h2>

      <p className="content">This is the home page of our application.</p>

    </div>

  );

};

export default Home;

Here’s an explanation of the Home component code:

1. **Imports**:
   * import React from 'react';: Imports the React library, which is necessary to create and use React components.
   * import './Home.css';: Imports a CSS file for styling the Home component, ensuring the component's layout and design are defined according to the styles in this file.
2. **Functional Component**:
   * const Home = () => { ... }: Defines a functional component named Home. This is a JavaScript function that returns JSX, which React will render to the screen.
3. **JSX Structure**:
   * <div className="container">: Creates a container div with the CSS class container. This class is used for applying styles defined in the Home.css file.
   * <h2 className="title">Welcome to our website!</h2>: Renders an <h2> heading with the class title, which is styled according to the Home.css file. It provides a welcoming message to users.
   * <p className="content">This is the home page of our application.</p>: Renders a paragraph with the class content, describing the home page of the application. The content is styled with the content class from the CSS file.
4. **Styling**:
   * The classes container, title, and content are defined in the Home.css file. These classes are used to apply specific styles to the div, h2, and p elements respectively, allowing for a consistent look and feel.
5. **Export Statement**:
   * export default Home;: Exports the Home component as the default export from this module. This allows other parts of the application to import and use the Home component, typically in routing or other component compositions.

This component serves as the home page of the application, providing a welcome message and a brief description. It leverages CSS for styling to maintain a clean and organized presentation.

Login.css

/\* Login.css \*/

.login-container {

max-width: 400px;

margin: 0 auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

}

.login-container h2 {

margin-top: 0;

}

.form-group {

margin-bottom: 15px;

}

.form-group label {

display: block;

margin-bottom: 5px;

}

.form-group input[type='email'],

.form-group input[type='password'] {

width: 100%;

padding: 8px;

}

.error {

color: red;

font-size: 12px;

}

button {

background-color: #007bff;

color: #fff;

border: none;

padding: 10px 20px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

p {

margin-top: 15px;

}

Login.js

// Login.js

import React, { useState } from 'react';

import { Navigate, useNavigate, Link  } from 'react-router-dom';

import './Login.css'; // Import the CSS file

const Login = () => {

  const [formData, setFormData] = useState({

    email: '',

    password: '',

  });

  const [errors, setErrors] = useState({});

  const navigate = useNavigate();

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData({ ...formData, [name]: value });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    const validationErrors = validateForm(formData);

    if (Object.keys(validationErrors).length === 0) {

      // Form is valid, perform login logic (not implemented in this example)

      alert('Login successful!');

      navigate("/")

    } else {

      // Form is invalid, display errors

      setErrors(validationErrors);

    }

  };

  const validateForm = (data) => {

    const errors = {};

    // Basic email validation

    if (!data.email) {

      errors.email = 'Email is required';

    } else if (!isValidEmail(data.email)) {

      errors.email = 'Invalid email address';

    }

    // Password validation

    if (!data.password) {

      errors.password = 'Password is required';

    } else if (data.password.length < 8) {

      errors.password = 'Password must be at least 8 characters long';

    }

    return errors;

  };

  const isValidEmail = (email) => {

    // Basic email validation regex

    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    return regex.test(email);

  };

  return (

    <div className="login-container">

      <h2>Login</h2>

      <form onSubmit={handleSubmit}>

        <div className="form-group">

          <label htmlFor="email">Email Address</label>

          <input type="email" id="email" name="email" value={formData.email} onChange={handleChange} />

          {errors.email && <div className="error">{errors.email}</div>}

        </div>

        <div className="form-group">

          <label htmlFor="password">Password</label>

          <input type="password" id="password" name="password" value={formData.password} onChange={handleChange} />

          {errors.password && <div className="error">{errors.password}</div>}

        </div>

        <button type="submit">Login</button>

      </form>

      <p>Don't have an account? <Link to="/registration">Register here</Link></p>

    </div>

  );

};

export default Login;

Here’s an explanation of the Login component code:

1. **Imports**:
   * import React, { useState } from 'react';: Imports React and the useState hook for managing component state.
   * import { useNavigate, Link } from 'react-router-dom';: Imports navigation and routing components from react-router-dom. useNavigate is used to programmatically navigate, and Link is used to create navigable links.
   * import './Login.css';: Imports a CSS file for styling the Login component.
2. **State Management**:
   * const [formData, setFormData] = useState({ email: '', password: '' });: Initializes state to store the form input values for email and password.
   * const [errors, setErrors] = useState({});: Initializes state to store validation error messages.
3. **Event Handlers**:
   * handleChange: Updates the formData state when the user types into form fields. It updates the corresponding field in the state based on the input's name attribute.
   * handleSubmit: Prevents the default form submission behavior, validates the form data, and either shows an alert and redirects to the home page if valid, or sets error messages if invalid.
4. **Validation Functions**:
   * validateForm: Validates the form data, returning an object with error messages for invalid fields. It checks for required fields and validates the email format and password length.
   * isValidEmail: Uses a regular expression to validate the email format. Returns true if the email matches the pattern and false otherwise.
5. **JSX Rendering**:
   * The return statement renders the login form within a div with the class login-container.
   * Includes <h2>Login</h2> for the form title.
   * The form fields for email and password are wrapped in divs with the class form-group, each having a label and conditional error messages.
   * The button with type="submit" triggers the handleSubmit function when clicked.
   * A <p> tag with a Link component provides a navigable link to the registration page for users who don’t have an account.

This component manages login functionality, including form data handling, validation, and navigation. It also includes a link to the registration page for users who need to sign up.

Navigation.css

/\* Navigation.css \*/

nav {

    background-color: #333;

    color: #fff;

    padding: 10px;

  }

  nav ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

  }

  nav ul li {

    display: inline;

    margin-right: 10px;

  }

  nav ul li a {

    color: #fff;

    text-decoration: none;

  }

  nav ul li a:hover {

    text-decoration: underline;

  }

Navigation.js

// Navigation.js

import React from 'react';

import { Link } from 'react-router-dom';

import './Navigation.css'; // Import the CSS file

const Navigation = () => {

  return (

    <nav>

      <ul>

        <li><Link to="/">Home</Link></li>

        <li><Link to="/registration">Registration</Link></li>

        <li><Link to="/login">Login</Link></li>

      </ul>

    </nav>

  );

};

export default Navigation;

Here’s an explanation of the Navigation component code:

1. **Imports**:
   * import React from 'react';: Imports the React library, which is necessary for creating and using React components.
   * import { Link } from 'react-router-dom';: Imports the Link component from react-router-dom to create navigable links that enable routing within the application.
   * import './Navigation.css';: Imports a CSS file for styling the Navigation component.
2. **Functional Component**:
   * const Navigation = () => { ... }: Defines a functional component named Navigation. This component returns JSX that represents the navigation menu.
3. **Navigation Structure**:
   * <nav>: The nav element represents a section of a page that contains navigation links. It helps in organizing the navigation menu semantically.
   * <ul>: An unordered list element is used to group the navigation links. It provides a list structure for the links.
   * <li>: Each list item represents an individual navigation link within the list.
4. **Links**:
   * <Link to="/">Home</Link>: Creates a navigable link to the root path (/). Clicking this link will navigate to the Home page.
   * <Link to="/registration">Registration</Link>: Creates a navigable link to the registration page (/registration).
   * <Link to="/login">Login</Link>: Creates a navigable link to the login page (/login).
5. **CSS Styling**:
   * The Navigation component relies on styles defined in Navigation.css to style the navigation menu. This CSS file can include rules for the nav, ul, and li elements to ensure the navigation bar is visually appealing and functional.

This component provides a basic navigation bar with links to the Home, Registration, and Login pages, allowing users to easily navigate through different sections of the application.

**Comments**

We've done a fantastic job setting up the structure for our React application!

It looks like we have everything organized neatly, including separate components for each page, CSS files for styling, and a clear navigation structure.

Here's a summary of what we've achieved:

1. **App.js**: We've set up our main component where we define routes using **BrowserRouter** and **Routes** from React Router.
2. **We**'ve also included the **Navigation** component to provide navigation links throughout the application.
3. **Components**: We have separate components for each page of our application:
   * **Home.js**: Displays a welcome message on the home page.
   * **Registration.js**: Implements a registration form with form validation for email and password fields.
   * **Login.js**: Implements a login form with form validation for email and password fields.
4. **CSS Styling**:
   * We've provided CSS files for styling each component and the global styles for the entire application.
   * Each CSS file contains styles specific to the corresponding component, ensuring modularity and maintainability.
5. **Navigation**:
   * **We**'ve created a **Navigation** component to provide navigation links to different pages of our application.
   * This component is included in the main layout and ensures consistent navigation throughout the app.

Overall, our application structure is well-organized and follows best practices for React development.

We've separated concerns by creating reusable components, applied CSS styling effectively, and implemented navigation using React Router.