

Task 5: Implementing Routing in a Profile Website with React Router

Objective:

Set up and master React Router for routing in a user profile website. This task aims to enhance your understanding of routing, including setting up basic routes, handling nested routes, using route parameters, and implementing navigation.

Pre-requisites:

- Basic understanding of HTML, CSS, and JavaScript
- Familiarity with a code editor like Visual Studio Code
- Basic knowledge of React

Concepts Covered:

- Installing and Setting Up React Router
- Creating Basic Routes
- Implementing Nested Routes and Route Parameters
- Adding Navigation Links and Programmatically Navigating

Concepts:

1. Installing and Setting Up React Router:

Install React Router (`react-router-dom`) and set up basic routing in your application.

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

```
// src/index.js
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router } from 'react-router-dom';
import App from './App';

ReactDOM.render(
  <Router>
    <App />
  </Router>,
  document.getElementById('root')
);
```

2. Creating Basic Routes:

Define routes for your profile website, such as Home, Profile, About, and Contact.

```
// src/App.js
import React from 'react';
import { Route, Switch } from 'react-router-dom';
import Home from './components/Home';
import Profile from './components/Profile';
import About from './components/About';
import Contact from './components/Contact';

const App = () => {
  return (
    <div>
      <Switch>
        <Route exact path="/" component={Home} />
        <Route path="/profile" component={Profile} />
        <Route path="/about" component={About} />
        <Route path="/contact" component={Contact} />
      </Switch>
    </div>
  );
};

export default App;
```

3. Implementing Nested Routes and Route Parameters:

Create a nested route for individual profile details and use route parameters to display specific profile information.

```
// src/components/Profile.js
import React from 'react';
import { Route, useRouteMatch, Link } from 'react-router-dom';
import ProfileDetail from './ProfileDetail';

const profiles = [
  { id: 1, name: 'John Doe', occupation: 'Software Engineer' },
  { id: 2, name: 'Jane Smith', occupation: 'Graphic Designer' },
  { id: 3, name: 'Sam Johnson', occupation: 'Product Manager' }
];

const Profile = () => {
  const { url, path } = useRouteMatch();

  return (
    <div>
      <h2>Profiles</h2>
      <ul>
        {profiles.map(profile => (
          <li key={profile.id}>
            <Link to={`${url}/${profile.id}`}>{profile.name}</Link>
          </li>
        ))}
      </ul>
    </div>
  );
};
```

```

        ))}
      </ul>
      <Route path={`${path}/:profileId`} component={ProfileDetail} />
    </div>
  );
};

export default Profile;

// src/components/ProfileDetail.js
import React from 'react';
import { useParams } from 'react-router-dom';

const profiles = [
  { id: 1, name: 'John Doe', occupation: 'Software Engineer' },
  { id: 2, name: 'Jane Smith', occupation: 'Graphic Designer' },
  { id: 3, name: 'Sam Johnson', occupation: 'Product Manager' }
];

const ProfileDetail = () => {
  const { profileId } = useParams();
  const profile = profiles.find(p => p.id === parseInt(profileId));

  if (!profile) {
    return <h3>Profile not found</h3>;
  }

  return (
    <div>
      <h2>{profile.name}</h2>
      <p>Occupation: {profile.occupation}</p>
    </div>
  );
};

export default ProfileDetail;

```

4. Adding Navigation Links and Programmatically Navigating:

Use the `Link` component to add navigation links in your application, and implement programmatic navigation upon certain actions, like after updating a profile.

```

// src/components/Navbar.js
import React from 'react';
import { Link } from 'react-router-dom';

const Navbar = () => {
  return (
    <nav>
      <ul>

```

```

        <li><Link to="/">Home</Link></li>
        <li><Link to="/profile">Profiles</Link></li>
        <li><Link to="/about">About</Link></li>
        <li><Link to="/contact">Contact</Link></li>
      </ul>
    </nav>
  );
};

export default Navbar;

// src/App.js
import React from 'react';
import { Route, Switch } from 'react-router-dom';
import Navbar from './components/Navbar';
import Home from './components/Home';
import Profile from './components/Profile';
import About from './components/About';
import Contact from './components/Contact';

const App = () => {
  return (
    <div>
      <Navbar />
      <Switch>
        <Route exact path="/" component={Home} />
        <Route path="/profile" component={Profile} />
        <Route path="/about" component={About} />
        <Route path="/contact" component={Contact} />
      </Switch>
    </div>
  );
};

export default App;

// src/components/ProfileDetail.js
import React from 'react';
import { useParams, useHistory } from 'react-router-dom';

const profiles = [
  { id: 1, name: 'John Doe', occupation: 'Software Engineer' },
  { id: 2, name: 'Jane Smith', occupation: 'Graphic Designer' },
  { id: 3, name: 'Sam Johnson', occupation: 'Product Manager' }
];

const ProfileDetail = () => {
  const { profileId } = useParams();
  const profile = profiles.find(p => p.id === parseInt(profileId));
  const history = useHistory();

```

```

const handleBack = () => {
  history.push('/profile');
};

if (!profile) {
  return <h3>Profile not found</h3>;
}

return (
  <div>
    <h2>{profile.name}</h2>
    <p>Occupation: {profile.occupation}</p>
    <button onClick={handleBack}>Back to Profiles</button>
  </div>
);
};

export default ProfileDetail;

```

Setup:

1. Install Visual Studio Code (VS Code):

Download and install VS Code from [Visual Studio Code](#).

2. React Developer Tools:

Install the React Developer Tools browser extension for [Chrome](#) or [Firefox](#).

3. Git:

Install Git for version control. You can download it from [Git](#).

4. Node.js and npm:

Ensure Node.js and npm are installed on your machine. You can download and install them from [Node.js](#).

5. Setting Up the React Project:

- Open your terminal.
- Navigate to the directory where you want to create your project.
- Run the following command to create a new React project:

```
npx create-react-app profile-website
```

6. Installing Additional Dependencies:

- Navigate to the project directory:

```
cd profile-website
```

- Install React Router:

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

Tasks:

Part 1: Setting Up and Basic Routing (30 minutes)

1. Installing and Setting Up React Router (15 minutes):

- Install React Router (`react-router-dom`) and set up basic routing in your application.
- Example:

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

```
// src/index.js
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router } from 'react-router-dom';
import App from './App';

ReactDOM.render(
  <Router>
    <App />
  </Router>,
  document.getElementById('root')
);
```

- Goal: Learn how to integrate React Router into a React project and understand the setup process.

2. Creating Basic Routes (15 minutes):

- Define routes for your profile website, such as Home, Profile, About, and Contact.
- Example:

```
// src/App.js
import React from 'react';
import { Route, Switch } from 'react-router-dom';
import Home from './components/Home';
import Profile from './components/Profile';
import About from './components/About';
import Contact from './components/Contact';

const App = () => {
  return (
    <div>
      <Switch>
        <Route exact path="/" component={Home} />
        <Route path="/profile" component
```

```
=>{Profile} /> );
```

```
export default App;
```
```

- Goal: Understand how to create different routes and render components based on the URL path.

## Part 2: Advanced Routing Concepts (30 minutes)

### 1. Implementing Nested Routes and Route Parameters (15 minutes):

- Create a nested route for individual profile details and use route parameters to display specific profile information.
- Example:

```
// src/components/Profile.js
import React from 'react';
import { Route, useRouteMatch, Link } from 'react-router-dom';
import ProfileDetail from './ProfileDetail';

const profiles = [
 { id: 1, name: 'John Doe', occupation: 'Software Engineer' },
 { id: 2, name: 'Jane Smith', occupation: 'Graphic Designer' },
 { id: 3, name: 'Sam Johnson', occupation: 'Product Manager' }
];

const Profile = () => {
 const { url, path } = useRouteMatch();

 return (
 <div>
 <h2>Profiles</h2>

 {profiles.map(profile => (
 <li key={profile.id}>
 <Link to={`${url}/${profile.id}`}>{profile.name}
 </Link>

))}

 <Route path={`${path}/:profileId`} component={ProfileDetail} />
 </div>
);
};

export default Profile;

// src/components/ProfileDetail.js
import React from 'react';
import { useParams } from 'react-router-dom';

const profiles = [
```

```

 { id: 1, name: 'John Doe', occupation: 'Software Engineer' },
 { id: 2, name: 'Jane Smith', occupation: 'Graphic Designer' },
 { id: 3, name: 'Sam Johnson', occupation: 'Product Manager' }
];

const ProfileDetail = () => {
 const { profileId } = useParams();
 const profile = profiles.find(p => p.id === parseInt(profileId));

 if (!profile) {
 return <h3>Profile not found</h3>;
 }

 return (
 <div>
 <h2>{profile.name}</h2>
 <p>Occupation: {profile.occupation}</p>
 </div>
);
};

export default ProfileDetail;

```

- Goal: Learn to handle nested routes and dynamic segments in URLs to render specific content.

## 2. Adding Navigation Links and Programmatically Navigating (15 minutes):

- Use the `Link` component to add navigation links in your application, and implement programmatic navigation upon certain actions, like after updating a profile.
- Example:

```

// src/components/Navbar.js
import React from 'react';
import { Link } from 'react-router-dom';

const Navbar = () => {
 return (
 <nav>

 <Link to="/">Home</Link>
 <Link to="/profile">Profiles</Link>
 <Link to="/about">About</Link>
 <Link to="/contact">Contact</Link>

 </nav>
);
};

export default Navbar;

```



```

// src/App.js
import React from 'react';
import { Route, Switch } from 'react-router-dom';
import Navbar from './components/Navbar';
import Home from './components/Home';
import Profile from './components/Profile';
import About from './components/About';
import Contact from './components/Contact';

const App = () => {
 return (
 <div>
 <Navbar />
 <Switch>
 <Route exact path="/" component={Home} />
 <Route path="/profile" component={Profile} />
 <Route path="/about" component={About} />
 <Route path="/contact" component={Contact} />
 </Switch>
 </div>
);
};

export default App;

// src/components/ProfileDetail.js
import React from 'react';
import { useParams, useHistory } from 'react-router-dom';

const profiles = [
 { id: 1, name: 'John Doe', occupation: 'Software Engineer' },
 { id: 2, name: 'Jane Smith', occupation: 'Graphic Designer' },
 { id: 3, name: 'Sam Johnson', occupation: 'Product Manager' }
];

const ProfileDetail = () => {
 const { profileId } = useParams();
 const profile = profiles.find(p => p.id === parseInt(profileId));
 const history = useHistory();

 const handleBack = () => {
 history.push('/profile');
 };

 if (!profile) {
 return <h3>Profile not found</h3>;
 }

 return (
 <div>
 <h2>{profile.name}</h2>

```

```

 <p>Occupation: {profile.occupation}</p>
 <button onClick={handleBack}>Back to Profiles</button>
 </div>
);
 };

 export default ProfileDetail;

```

- Goal: Understand how to navigate users around the site using links and programmatically change routes based on user interactions.




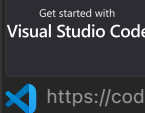
### Instructions:

1. Write the required code in the appropriate files in the `src/components` directory.
2. Open the terminal and navigate to the project directory.
3. Run the React project using:

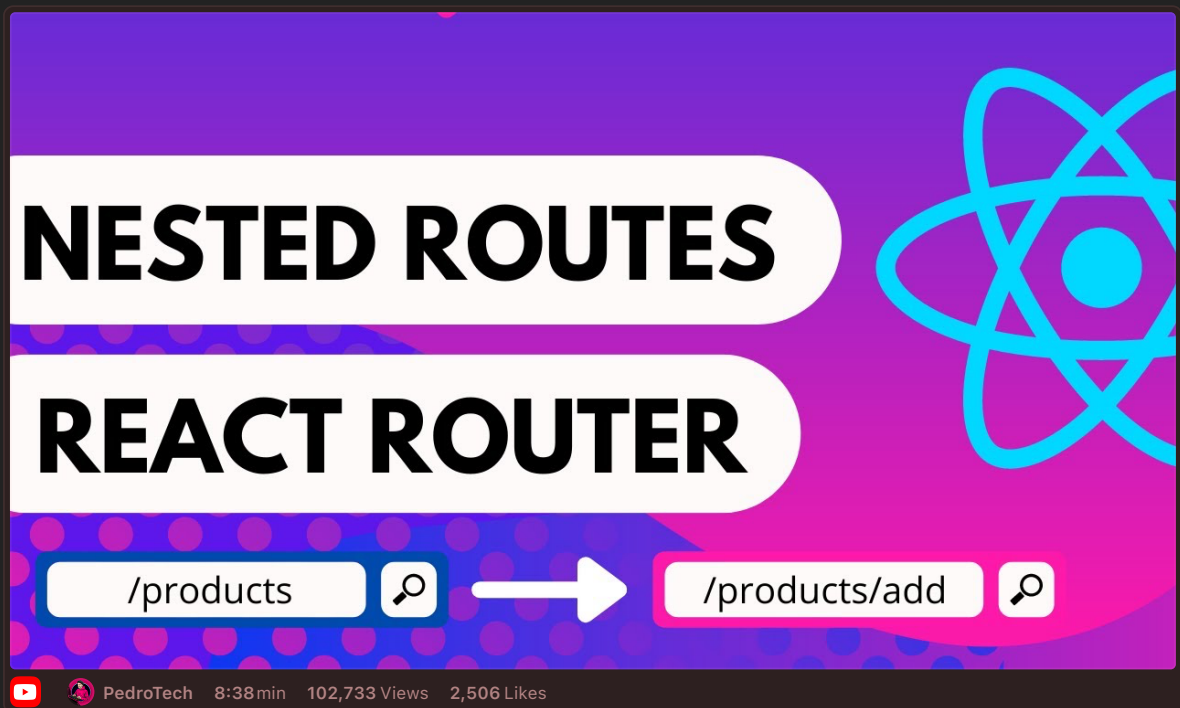
```
npm start
```

4. Open the project in a web browser to ensure the code displays correctly.
5. Use the browser's developer tools and React Developer Tools to debug and inspect the elements.

### Resources:

- 
**Not Found**  
<https://reactrouter.com/web/guides/quick-start>
- 
**Getting Started – React**  
 A JavaScript library for building user interfaces  
<https://reactjs.org/docs/getting-started.html>
- 
**Create React App**  
 Set up a modern web app by running one command.  
<https://create-react-app.dev/>
- 
**Documentation for Visual Studio Code**  
 Find out how to set-up and get the most from Visual Studio Code. Optimized for building and d...  
<https://code.visualstudio.com/docs>

### Videos:





#### GitHub Instructions:

1. **Open in Visual Studio Code:**

After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, VSCode will open the repository directly. If prompted, select "Open" or "Allow" to open the repository in VSCode.

2. **Open the Terminal in VSCode:**

In VSCode, open a terminal by selecting Terminal > New Terminal from the top menu.

3. **Complete the Task:**

In VSCode, write your solution in the appropriate files in the `src/components` directory.

4. **Run and Test Your Code:**

Open your terminal, navigate to your project directory, and run:

```
npm start
```

5. **Commit Your Changes:**

In the VSCode terminal, add your changes to git:

```
git add src/components/*
```

Commit your changes with a meaningful message:

```
git commit -m "Completed task 25"
```

## 6. Push Your Changes to Your Repository:

Push your changes to your forked repository:

```
git push origin main
```

## 7. Create a Pull Request:

Go to your repository on GitHub.

Click on the "Pull Requests" tab.

Click the "New Pull Request" button.

Ensure the base repository is the original template repository and the base branch is `main`.

Ensure the head repository is your forked repository and the compare branch is `main`.

Click "Create Pull Request".

Add a title and description for your pull request and submit it.

## Summary of Commands:

```
Open in Visual Studio Code

Open terminal in VSCode

Complete the task by editing src/components/*

Navigate to the project directory
cd profile-website

Run your code
npm start

Add, commit, and push your changes
git add src/components/*
git commit -m "Completed task 5"
git push origin main

Create a pull request on GitHub
```