CS 628: Full-Stack Development – Web App

City University of Seattle
School of Technology & Computing
Professor Sam Chung

HOS06: React Router

Samantha Hipple
August 7, 2023

PLEASE NOTE

Screenshots in this guide may differ from your environment (e.g., directory paths, version numbers, etc.). When choosing between a stable or most recent release, we advise you install the stable release rather than the best-testing version. Additionally, there may be subtle discrepancies along the steps, please use your best judgment to complete the tutorial. If you are unfamiliar with terminal, command line, and bash scripts, we recommend watching this video prior to moving forward with this guide. Not all steps are fully explained. Lastly, we advise that you avoid copy-pasting code directly from the guide or GitHub repositories. Instead, type out the code yourself to improve familiarity.

More information on this guide can be found under the related module in this repository. Please save a screenshot of the app at the end of each section and save it in the current module folder with the relevant section number.

SECTION CONTENTS

SECTION 1. ACCESSING GITHUB CODESPACES

GitHub Codespaces is an online cloud-based development environment that allows users to easily write, run and debug code. Codespaces is fully integrated with your GitHub repository and provides a seamless experience for developers. In order to access Codespaces, users only need a GitHub account and an active internet connection.

After downloading the current HOS assignment, in the top-right corner of the repo, click on the **<> Code** drop-down menu and select **Create codespace on main** as shown in the following image. The free and pro GitHub subscriptions include free use of GitHub Codespaces *up to a fixed*

amount of usage each month. In order to avoid unexpected charges, please review the billing information.

SECTION 2. BROWSERROUTER

React Router is a widely used library in the React ecosystem that facilitates efficient client-side routing for single-page applications (SPAs). React Router enables developers to create navigable user interfaces within a single HTML page, eliminating the need for full-page reloads. By defining routes and matching them to specific components, React Router dynamically renders the appropriate content based on Uniform Resource Locator (URL) changes, resulting in a seamless and fluid user experience. This library supports advanced features like nested routing, query parameter handling, and programmatic navigation, making it a powerful tool for managing client-side routing in React applications.

A **<BrowserRouter>** stores the current location in the browser's address bar using clean URLs and navigates using the browser's built-in history stack. The browser router is typically used as the top-level router component in the application and serves as the entry point for defining routes and mapping them to specific components.

1. Use the following terminal commands to start the project:

```
>>>npx create-react-app ./myapp
>>>cd myapp
>>>npm install react-router-dom
>>>npm start
```

2. Replace the code in **App.css** file with the content found here.
3. Create a file named **home.js** and add the following code:

```
const Home = () => ( <h1>HOS06 - React Router Fundamentals</h1> );

export default Home;
```

4. Create a file named **about.js** and add the following code:

```
const About = () => (
  <div>
    <h2>Fundamentals of ReactJS Router</h2>
    Learn more about React Router <a href="https://reactrouter.com/en/main"
                                     target="_blank"
                                     rel="noopener noreferrer">here</a>.
  </div>
);

export default About;
```

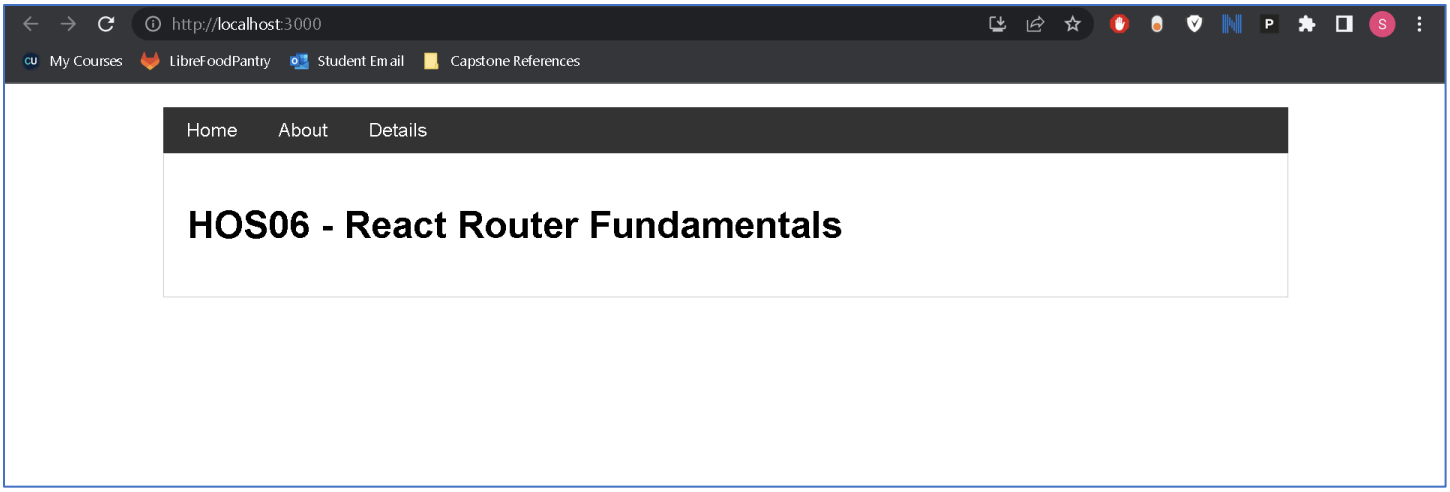
5. Create a file named **details.js** and add the following code:

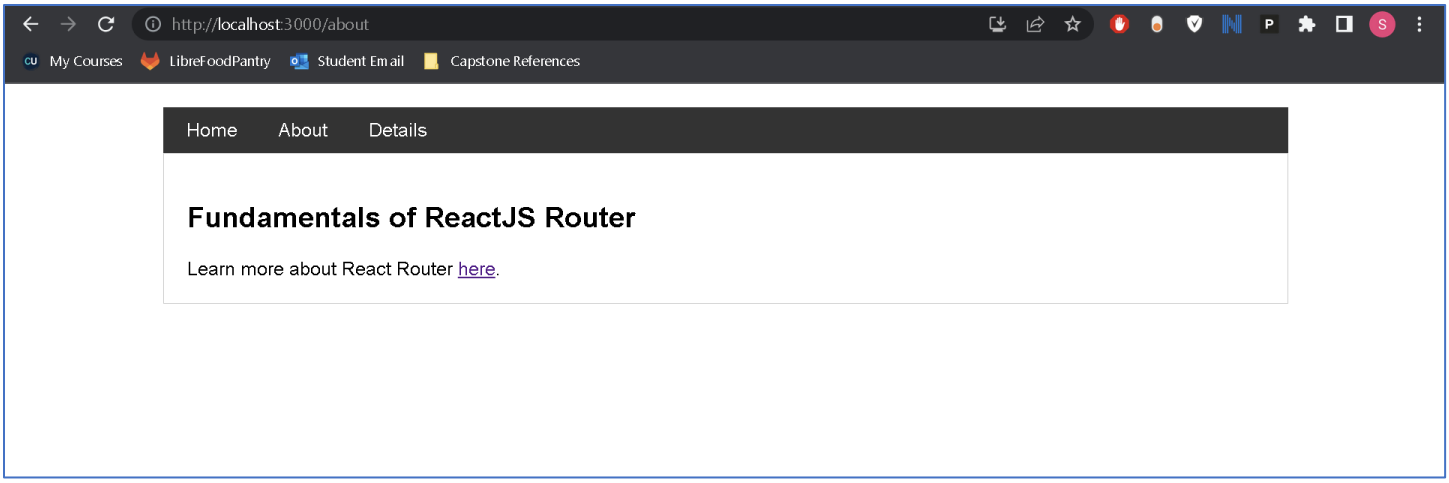
```
const Details = () => (
  <div>
    <h2>Understanding BrowserRouter</h2>
    <p>
      In this HOS, we are implementing routing using BrowserRouter! <br />
      You can learn more <a href="https://reactrouter.com/en/main/router-components/browser-router"
                            target="_blank" rel="noopener noreferrer">here</a>.
    </p>
  </div>
);

export default Details;
```

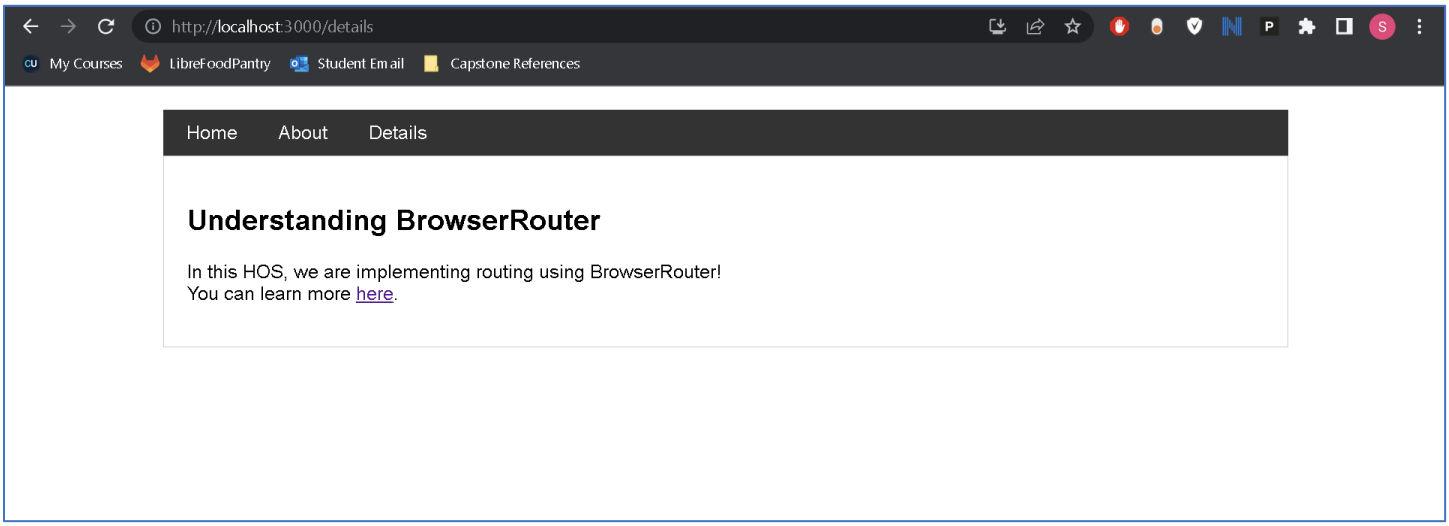
6. Updated **App.js** to match the following:

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

import Home from './components/home.js';
import About from './components/about.js';
import Details from './components/details.js';

import './App.css';

const App = () => (
  <BrowserRouter>
    <div className='container'>
      <nav className='navbar'>
        <ul>
          <li><Link to="/">Home</Link></li>
          <li><Link to="/about">About</Link></li>
          <li><Link to="/details">Details</Link></li>
        </ul>
      </nav>
      <div className='content'>
        <Routes>
          <Route exact path="/" element={<Home />} />
          <Route exact path="/about" element={<About />} />
          <Route exact path="/details" element={<Details />} />
        </Routes>
      </div>
    </div>
  </BrowserRouter>
);

export default App;
```

7. Refresh the browser and test the changes by viewing the different routes.







SECTION 3. MEMORYROUTER

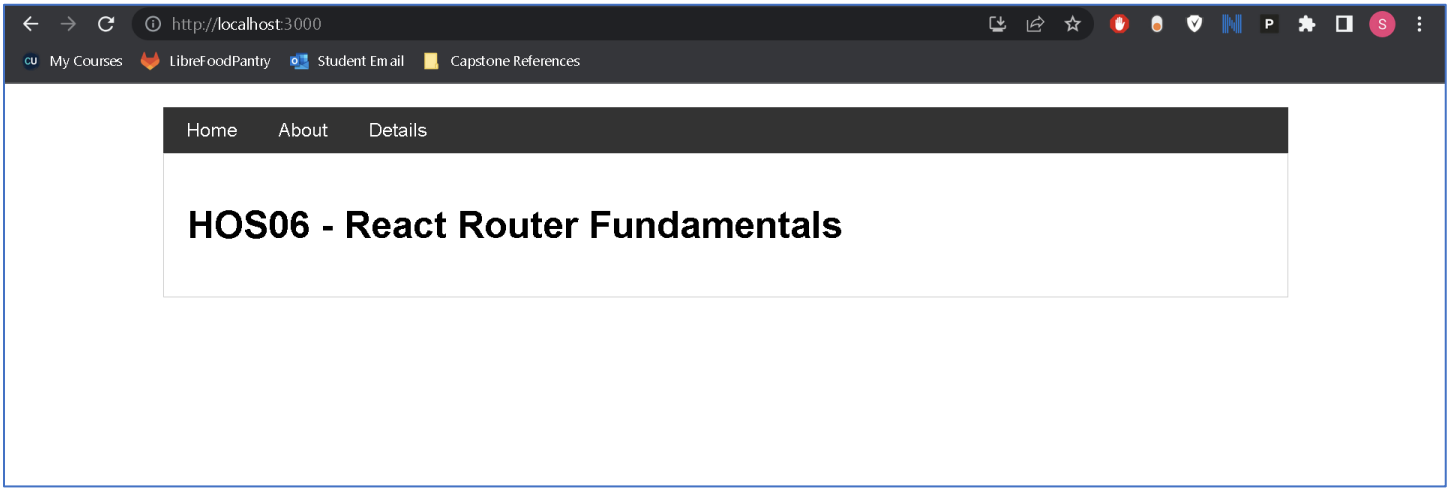
The **<MemoryRouter>** stores URL changes in memory instead of the user's browser. This keeps a history of URLs internally, bypassing the address bar and disabling the back & forward buttons on the browser. This is most beneficial for testing and non-browser environments, such as React Native, where browser-specific navigation features are not needed. Additionally, the memory

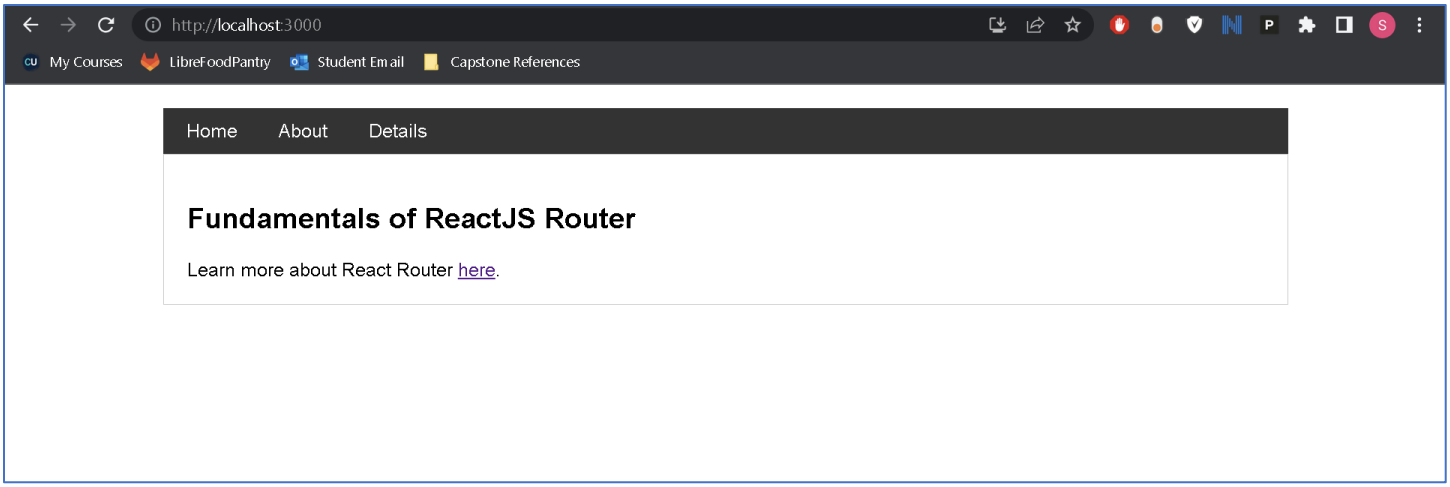
router allows developers to simulate URL changes and test routing behavior without affecting the actual browser history.

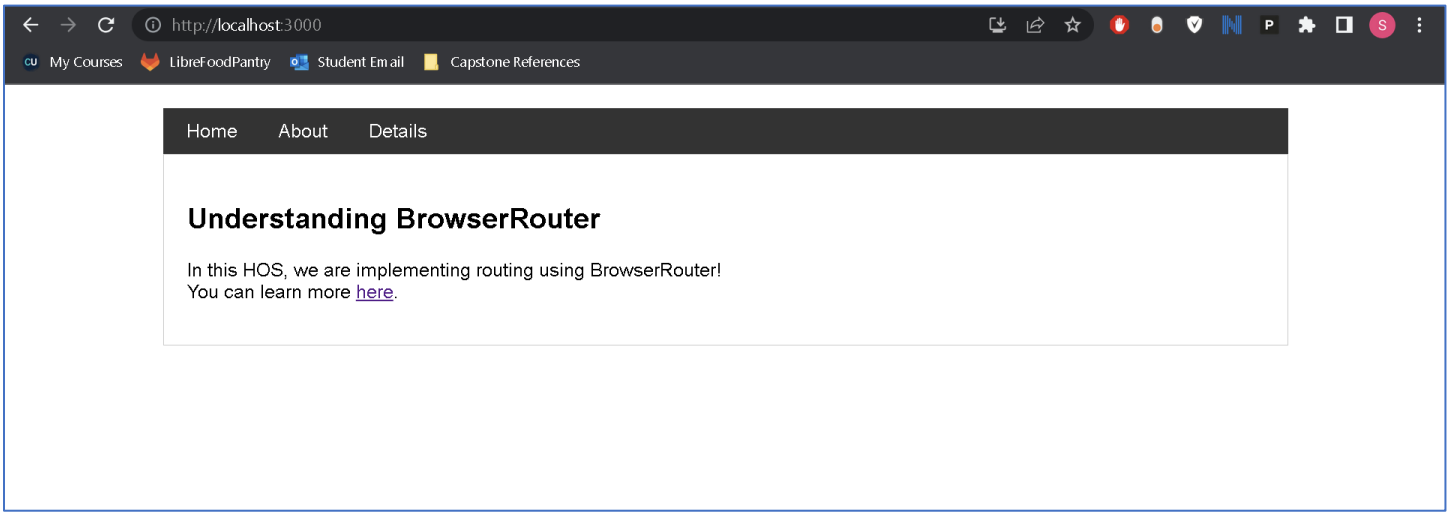
1. Update **App.js** by replacing the **<BrowserRouter>** with a **<MemoryRouter>**:

```
import { MemoryRouter, Routes, Route, Link } from 'react-router-dom';

import Home from './components/home.js';
import About from './components/about.js';
import Details from './components/details.js';

import './App.css';

const App = () => (
  <MemoryRouter>
    <div className='container'>
      <nav className='navbar'>
        <ul>
          <li><Link to="/">Home</Link></li>
          <li><Link to="/about">About</Link></li>
          <li><Link to="/details">Details</Link></li>
        </ul>
      </nav>
      <div className='content'>
        <Routes>
          <Route exact path="/" element={<Home />} />
          <Route exact path="/about" element={<About />} />
          <Route exact path="/details" element={<Details />} />
        </Routes>
      </div>
    </div>
  </MemoryRouter>
);

export default App;
```

2. Refresh the browser and test the changes by viewing the different routes.







Notice that the URL does not update with the routes when using the memory router.

SECTION 4. HASHROUTER

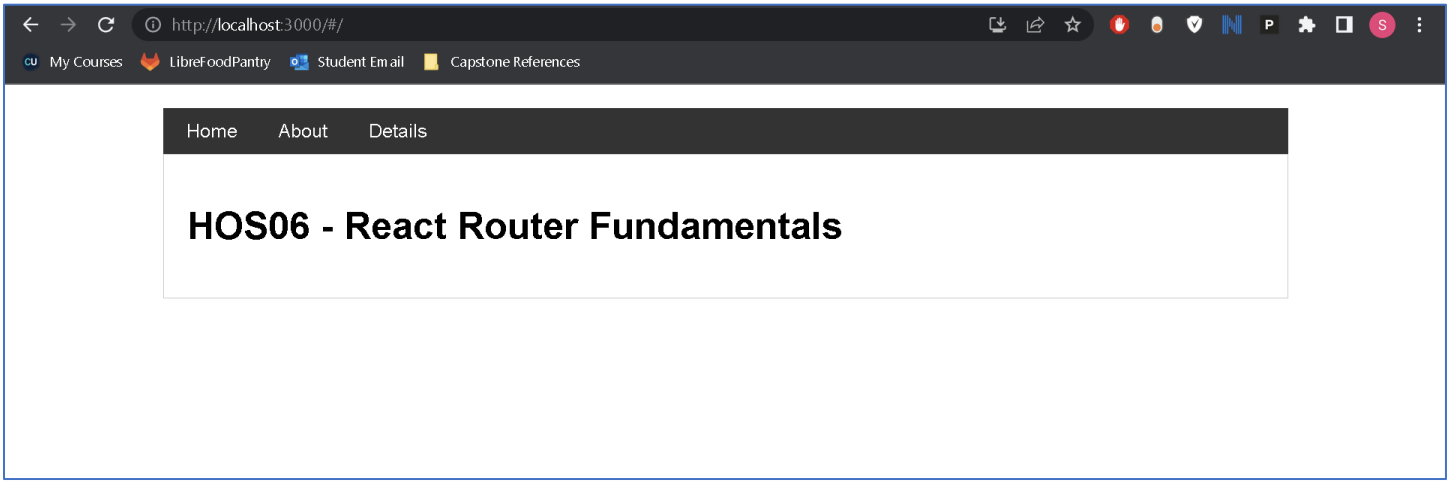
The **<HashRouter>** uses URL hashes to enable client-side navigation in SPAs. This is particularly useful in scenarios where server-side handling of URL changes is unavailable as it enables the back and forward buttons on the browser while the server disregards the hash portion of the URL and continues serving the **index.html** file for every request, leaving that handling of the hash values solely to the client-side application.

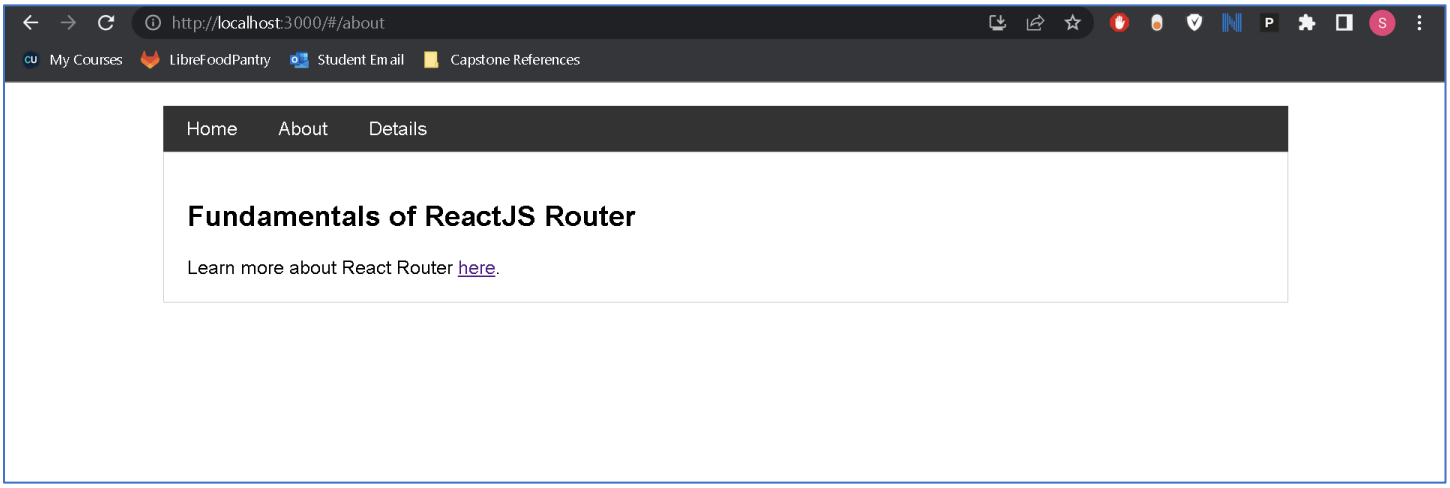
1. Update **App.js** by replacing the **<MemoryRouter>** with a **<HashRouter>**:

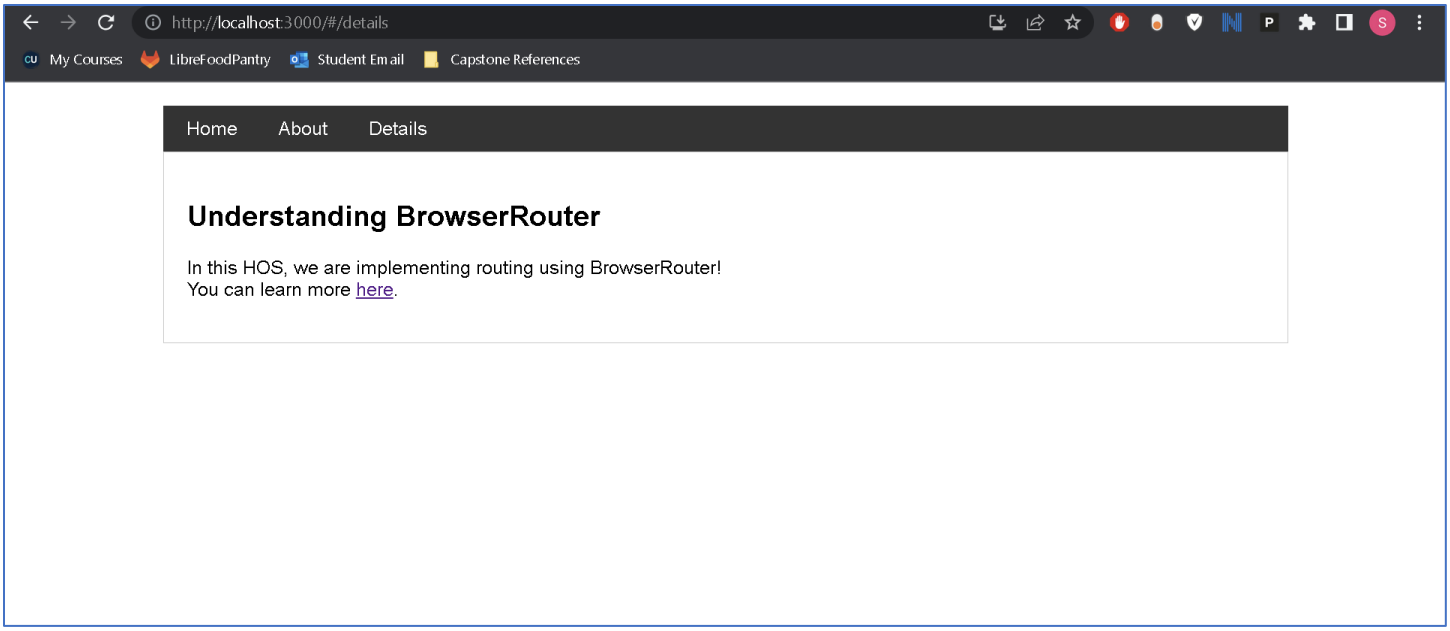
```
import { HashRouter, Routes, Route, Link } from 'react-router-dom';

import Home from './components/home.js';
import About from './components/about.js';
import Details from './components/details.js';

import './App.css';

const App = () => (
  <HashRouter>
```

```
    <div className='container'>
      <nav className='navbar'>
        <ul>
          <li><Link to="/">Home</Link></li>
          <li><Link to="/about">About</Link></li>
          <li><Link to="/details">Details</Link></li>
        </ul>
      </nav>
      <div className='content'>
        <Routes>
          <Route exact path="/" element={<Home />} />
          <Route exact path="/about" element={<About />} />
          <Route exact path="/details" element={<Details />} />
        </Routes>
      </div>
    </div>
  </HashRouter>
);

export default App;
```

2. Refresh the browser and test the changes by viewing the different routes.







SECTION 5. NESTED ROUTING

Nested routing in React Router defines routes within other routes, creating a hierarchical structure of components corresponding to specific URLs. With nested routing, you can render nested components based on the current URL path, allowing for more organized and modular code. Parent components can act as layouts, rendering common elements, while child components are swapped in and out based on the nested URL segments.

1. Create a new file called **team.js** and add the following code:

```
import { Link, Outlet } from 'react-router-dom';

const Team = () => (
  <div>
    <h2>This is Team Component</h2>
    <div>
      <ul><li><Link to="/team/member">Member</Link></li></ul>
    </div>
    <Outlet />
  </div>
);

export default Team;
```

The **<Outlet />** component is provided by the **react-router-dom** library and is used to render child routes within a parent route.

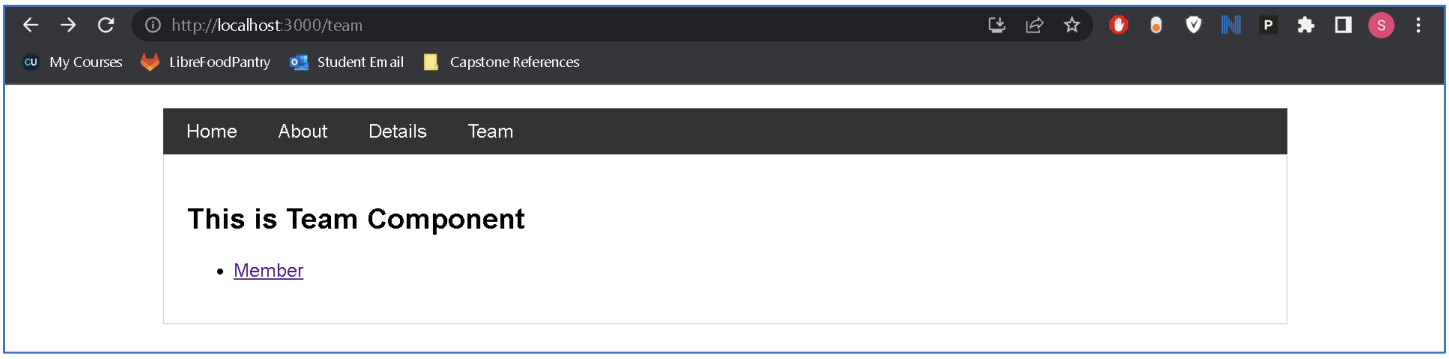
2. Create a new file called **member.js** and add the following code:

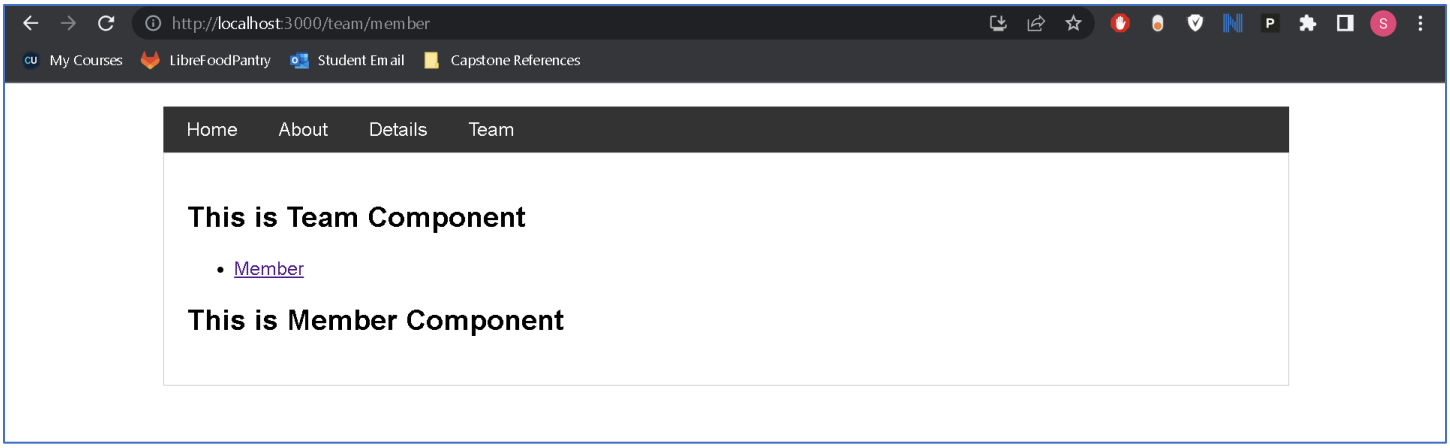
```
const Member = () => ( <div><h2>This is Member Component</h2></div> );

export default Member;
```

3. Update **App.js** to match the following:

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

import Home from './components/home.js';
import About from './components/about.js';
import Details from './components/details.js';
import Team from './components/team.js';
import Member from './components/member.js';

import './App.css';

const App = () => (
  <BrowserRouter>
    <div className='container'>
      <nav className='navbar'>
        <ul>
          <li><Link to="/">Home</Link></li>
          <li><Link to="/about">About</Link></li>
          <li><Link to="/details">Details</Link></li>
          <li><Link to="/team">Team</Link></li>
        </ul>
      </nav>
      <div className='content'>
        <Routes>
          <Route exact path="/" element={<Home />} />
          <Route exact path="/about" element={<About />} />
          <Route exact path="/details" element={<Details />} />
          <Route exact path="/team" element={<Team />}>
            <Route path="member" element={<Member />} />
          </Route>
        </Routes>
      </div>
    </div>
  </BrowserRouter>
);

export default App;
```

4. Refresh the browser and test the changes by clicking on the team and member links.





Here, **Member** is the child component rendered under the **Team** component using nested routing.

SECTION 6. QUERYING PARAMETERS

In React Router, **useParams** is a hook provided by the **react-router-dom** library that allows access to and extraction of dynamic parameters from the URL. The hook is specifically used to retrieve values from route parameters defined in the route path. When you define a route with a parameter placeholder in the path, such as **"/member/:id"**, the **useParams** hook allows you to extract the value of **:id** from the URL. Let us see an example.

1. Update **team.js** to match the following:

```
import { Link, Outlet } from 'react-router-dom';

const Team = () => (
  <div>
    <h2>This is Team Component</h2>
    <div>
      <ul>
        <li><Link to="/team/member/1">Member 1</Link></li>
        <li><Link to="/team/member/2">Member 2</Link></li>
        <li><Link to="/team/member/3">Member 3</Link></li>
      </ul>
    </div>
    <Outlet />
  </div>
);

export default Team;
```

2. Update **member.js** to match the following:

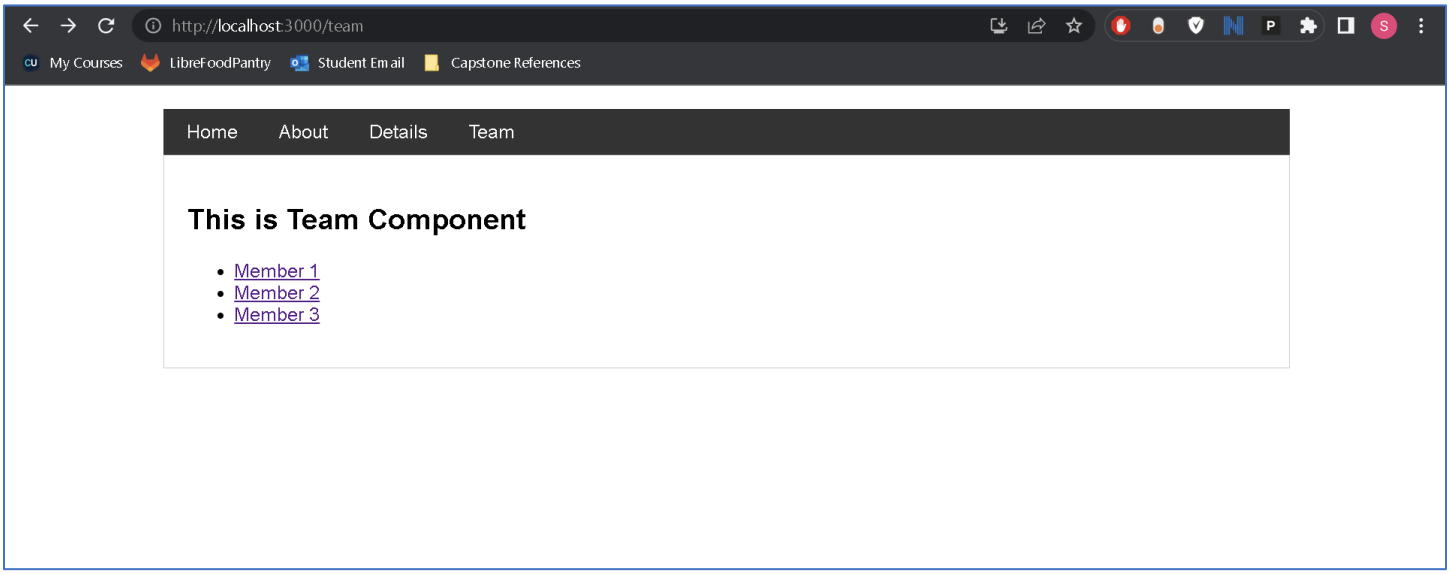
```
import { useParams } from 'react-router-dom';

const Member = () => {
  const { id } = useParams();
```

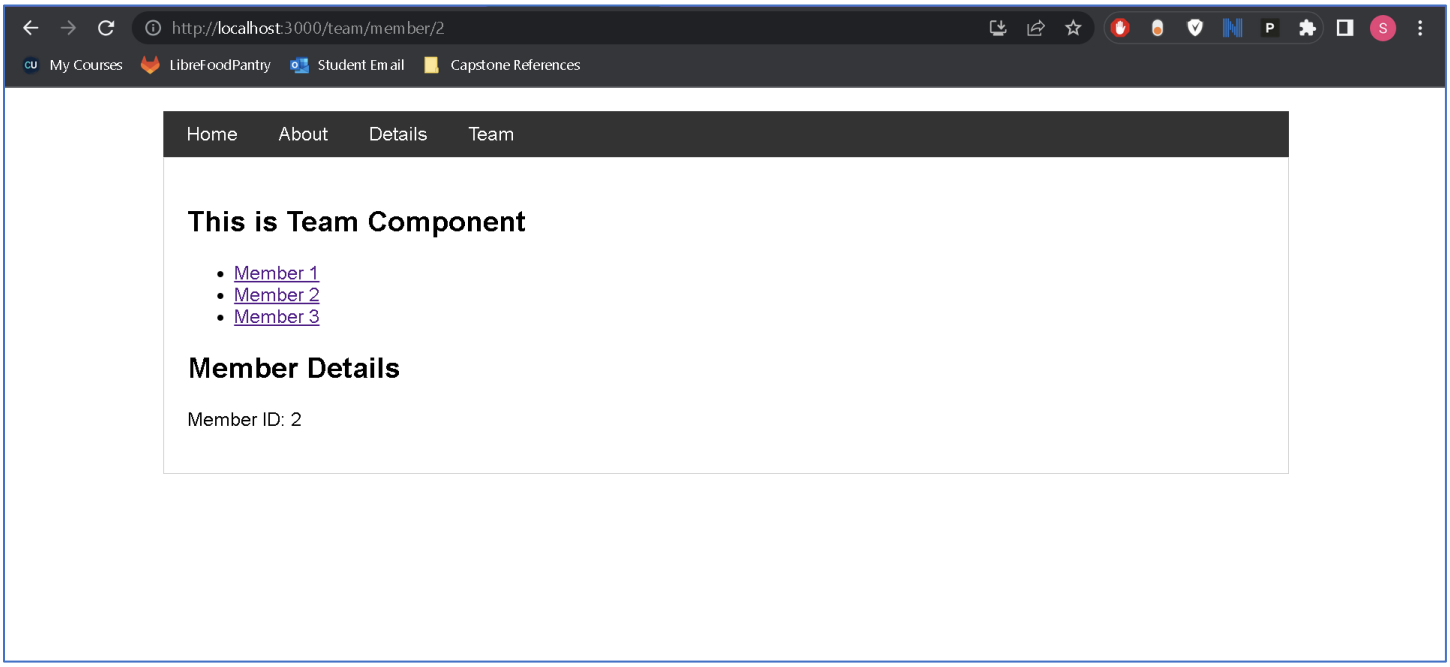
```
  return (
    <div>
      <h2>Member Details</h2>
      <p>Member ID: { id }</p>
    </div>
  );
}

export default Member;
```

3. Update the member **Route** in the **App** component to match the following:

```
          <Route path="member/:id" element={<Member />} />
```

4. Refresh the browser and test the changes by clicking on the team and member links.





SECTION 7. USING NAVLINK

In React Router, **NavLink** is a component provided by the **react-router-dom** library that is used to create navigation links within your application. It is like the regular **Link** component but comes with additional features specific to navigation. The **NavLink** component allows you to define navigation links that can have an **active** state based on the current URL. When the user clicks on a **NavLink**, React Router automatically applies an **active** class to the link's rendered HTML element if the current URL matches the link's **to** prop. Let us modify our **App** component to use **NavLink** components instead of **Link**.

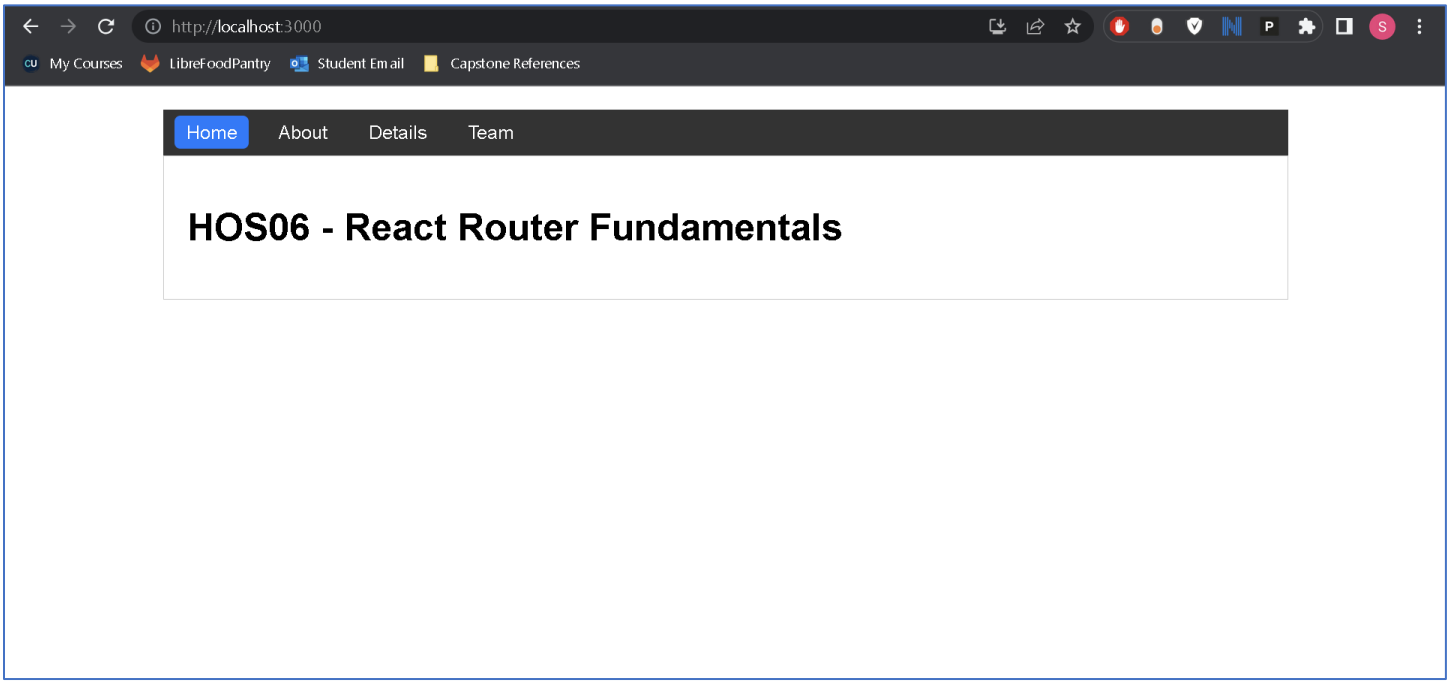
1. Replace the **Link** components with **NavLink** in **App.js** as follows:

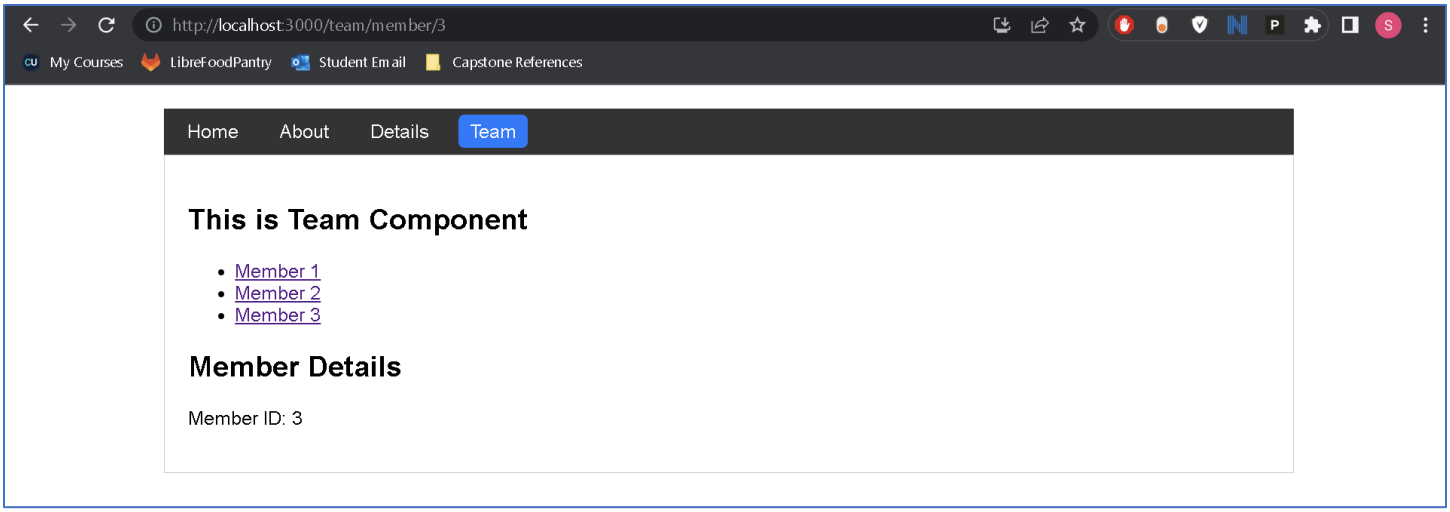
```
<ul>
  <li><NavLink to="/">Home</NavLink></li>
  <li><NavLink to="/about">About</NavLink></li>
  <li><NavLink to="/details">Details</NavLink></li>
  <li><NavLink to="/team">Team</NavLink></li>
</ul>
```

2. Update **App.css** to include the following style:

```
.active {
  color: #FFF;
  background-color: #007BFF;
  padding: 5px 10px;
  border-radius: 5px;
}
```

3. Refresh the browser and test the changes by viewing the different routes.





Notice that the selected tab is highlighted due to the styling we added in **App.css** without requiring you to provide a value to the **activeClassName** prop within the **NavLink** component.

SECTION 8. REDIRECT USING NAVIGATE

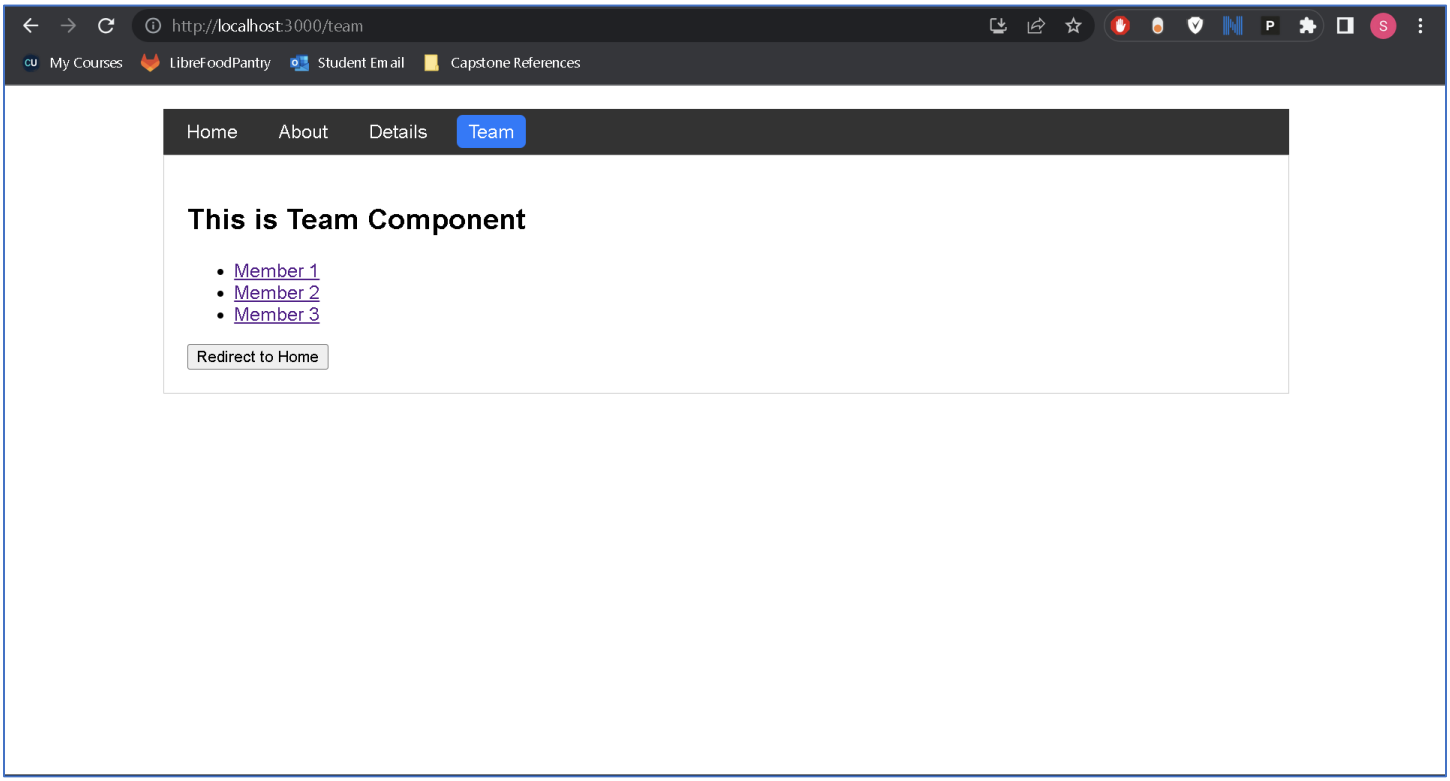
In React Router, the **<Navigate>** component is used to redirect users to a different route within the application programmatically. This enables the developer to navigate users to specific URLs without requiring any user interaction (i.e., click a link or button). Let us implement **Navigate** under the **Team** component.

1. Update **team.js** to match the following:

```
import { useState } from 'react';
import { Link, Outlet, Navigate } from 'react-router-dom';

const Team = () => {
  const [ redirectToHome, setRedirectToHome ] = useState(false);

  const handleRedirectToHome = () => setRedirectToHome(true);

  if (redirectToHome) { return <Navigate to="/" /> }

  return (
    <div>
      <h2>This is Team Component</h2>
      <div>
        <ul>
          <li><Link to="/team/member/1">Member 1</Link></li>
          <li><Link to="/team/member/2">Member 2</Link></li>
          <li><Link to="/team/member/3">Member 3</Link></li>
        </ul>
        <button onClick={handleRedirectToHome}>Redirect to Home</button>
      </div>
      <Outlet />
    </div>
  );
}
```

```
export default Team;
```

2. Refresh the browser and test the changes by pressing the new button under **Team**.



SECTION 9. HISTORY MANAGEMENT

The **useNavigate()** hook was introduced in React Router v6 as an alternative to the **useHistory()** hook found in previous versions. With **useHistory()**, developers would access the React Router history object and use **push** or **replace** methods for navigation. The hook enabled moving to specific URLs as well as navigating back and forth between previously explored routes. Let us update the **Team** and **Details** components with go back and redirect functions using the **useNavigate()** hook.

1. Update **team.js** to match the following:

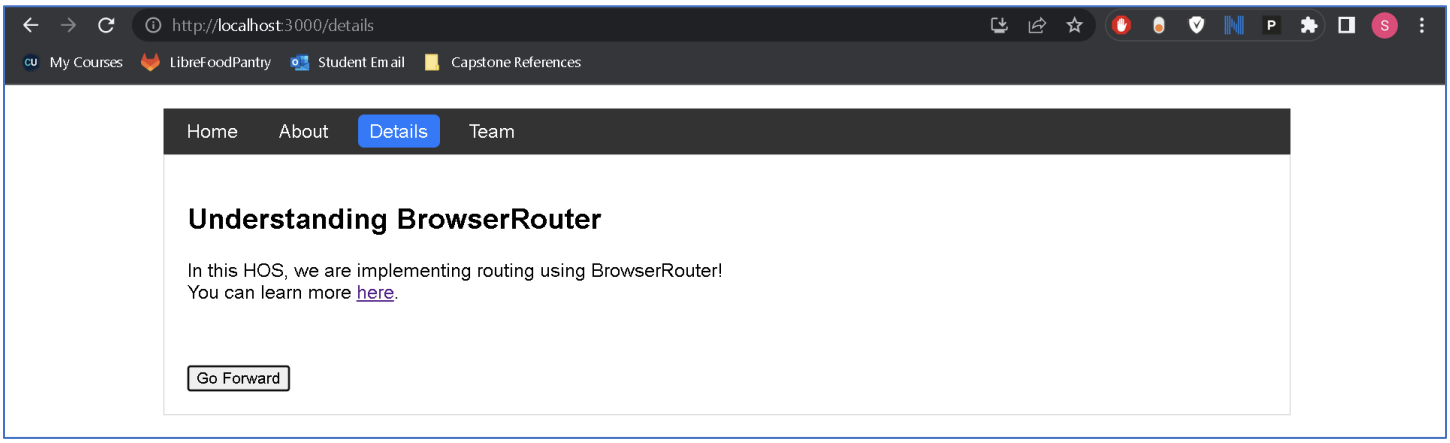
```
import { Link, Outlet, useNavigate } from 'react-router-dom';

const Team = () => {
  let navigate = useNavigate();

  const redirectToHome = () => navigate("/");
  const goBack = () => navigate(-1);

  return (
    <div>
      <h2>This is Team Component</h2>
      <div>
        <ul>
```

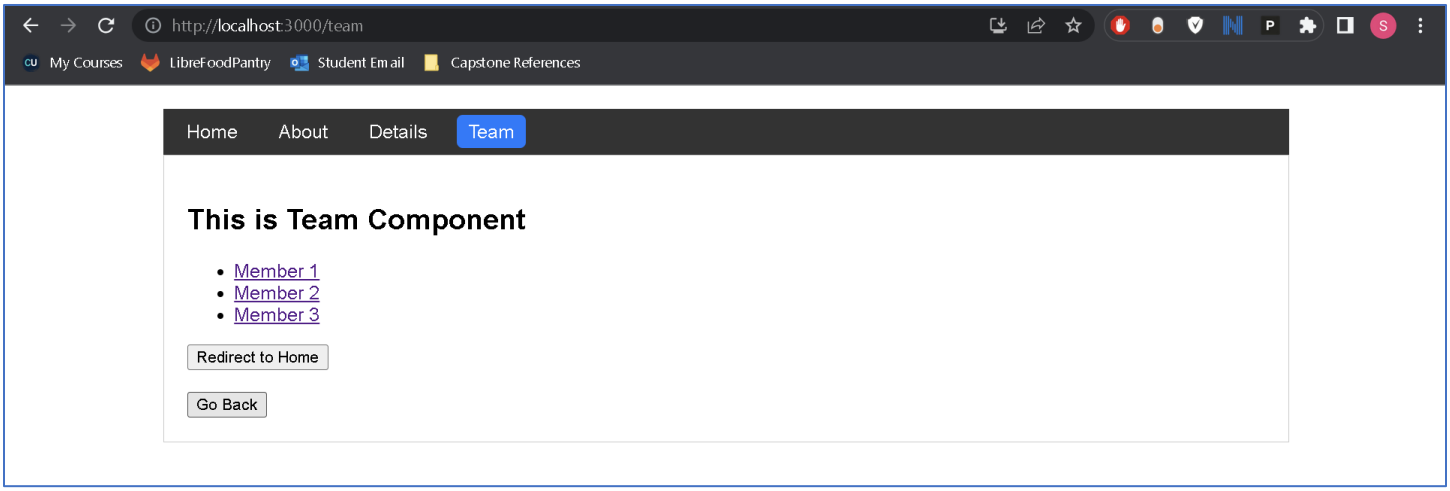
```
            <li><Link to="/team/member/1">Member 1</Link></li>
            <li><Link to="/team/member/2">Member 2</Link></li>
            <li><Link to="/team/member/3">Member 3</Link></li>
          </ul>
          <button onClick={redirectToHome}>Redirect to Home</button>
          <br /><br />
          <button onClick={goBack}>Go Back</button>
        </div>
        <Outlet />
      </div>
    );
}

export default Team;
```

2. Update **`details.js`** to match the following:

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

const Details = () => {
  let navigate = useNavigate();

  const goForward = () => navigate(1);

  return (
    <div>
      <h2>Understanding BrowserRouter</h2>
      <p>
        In this HOS, we are implementing routing using BrowserRouter! <br />
        You can learn more <a href="https://reactrouter.com/en/main/router-components/browser-router"
                              target="_blank" rel="noopener noreferrer">here</a>.
      </p>
      <br /><br />
      <button onClick={goForward}>Go Forward</button>
    </div>
  );
}

export default Details;
```

3. Refresh the browser and test the changes by playing with the navigation.





PUSH YOUR WORK TO GITHUB TO SUBMIT