Unit 8: React: Lab 9 – Menu (Routing Params)

Assigned: 02/09/23

Due: N/A

Completed: 02/09/23

**LMS:** <https://lms.grandcircus.co/mod/assign/view.php?id=22826>

**Google Doc:** <https://docs.google.com/document/d/1uMS1RPwzkFYjiLd1qlSrU1o99-rlsWkQNYgciiP6rzc/preview>

**GitHub:** <https://github.com/clintmsmith/GrandCircusLabs/tree/main/Menu_React_Routes/menuroutes>

**Task:** Create a React app [like this example](https://routing-exercise.surge.sh/) where a user can see a list of menu items and decide whether they would like to see additional information about a particular item.

**Build Specifications:**

1. Create components:
   1. Header
   2. MenuRoute
   3. BasicItem
   4. DetailsRoute
   5. SearchRoute (Extended Challenge)
2. The Header component will always be present, regardless of the route. The text “Menu” in the header is a link to the “/” path while the text “search” is a link to the “/search” path. (use <Link></Link>).
3. The MenuRoute component will be rendered when the path matches “/” exactly. The MenuRoute component will contain the BasicItem component. Use the map method to produce as many BasicItem components as there are items in the menu.
4. The DetailsRoute component will be rendered when the path matches “/details/:id” exactly. This component shows all the values of a particular menu item based on the id derived from the **path params**.
5. (**Extended Challenge**) The SearchRoute component will be rendered when the path matches “/search” exactly. This component will display a form where the user can enter particular criteria for menu items they wish to see. When the user submits the form, the user will be directed to the “/” path with relevant query string parameters. The MenuRoute should use the **query string parameter** values in order to filter the menu items displayed to the user. NOTE: You may pick which criteria you want to include in your form. Start with just one criteria and then add a second one if you have time.

1. Create an interface called Item which consists of:
   1. id: number
   2. name: string
   3. description: string
   4. calories: string
   5. price: string
   6. vegetarian: string
2. Install the React router

npm install react-router-dom @types/react-router-dom

1. Set up the Router with the various routes to components in App.tsx.  
   Make a file to hold the menu items that can be imported in the MenuRoute component and the DetailsRoute component.

const menu = [

 {

   id: "abc123",

   name: "Chicken and Waffles",

   description:

     "Fluffy waffles topped with lightly breaded chicken and maple syrup",

   price: 7.5,

   vegetarian: false,

   calories: 1100,

 },

 {

   id: "abc124",

   name: "Eggs benedict",

   description:

     "English muffin, each topped with Canadian bacon, a poached egg, and hollandaise sauce.",

   price: 6.75,

   vegetarian: false,

   calories: 550,

 },

 {

   id: "abc125",

   name: "Avocado Toast",

   description:

     "Smashed avocado, cojita cheese, shaved veg, cilantro, green chilli, watercress",

   price: 10.75,

   vegetarian: true,

   calories: 640,

 },

 {

   id: "abc126",

   name: "Beans on Toast",

   description:

     "Roasted mushrooms, poached egg, sweet peppers, pistachios, cashew butter",

   price: 9.25,

   vegetarian: false,

   calories: 800,

 },

 {

   id: "abc127",

   name: "Hash wrap",

   description:

     "Potato hash, smoked pork belly, egg, kale salad, bagna cauda sauce",

   price: 10.0,

   vegetarian: false,

   calories: 875,

 },

 {

   id: "abc128",

   name: "Cinnamon Roll",

   description:

     "Freshly baked rolls made from scratch topped with vanilla frosting",

   price: 4.5,

   vegetarian: true,

   calories: 970,

 },

 {

   id: "abc129",

   name: "Lox Bagel",

   description:

     "Smoked salmon, cream cheese and capers on top of a freshly baked sesame seed bagel",

   price: 8.5,

   vegetarian: false,

   calories: 1050,

 },

];

export default menu;

Because I didn’t have a React App already built for this, I just added the code to my existing Order Lab. We started by adding a Routes folder in our components folder and added five (5) files with basic function construction.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Starting in BasicItem we created the **interface**, which was then passed into the function, the return from which was the **props** for item.name and item.price that were put into some basic HTML tags.

Text

Description automatically generated

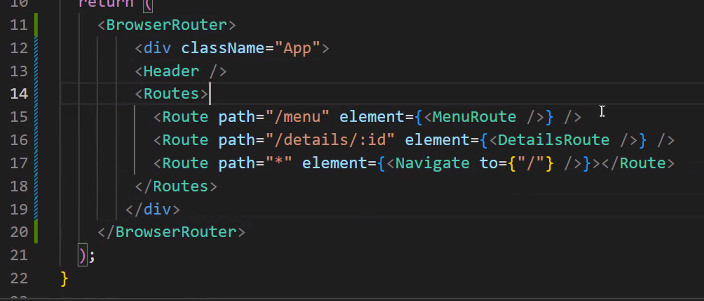
In App.tsx we added **<BrowserRouter>** opening and closing tags. This is imported from ‘react-router-dom’ which we installed with item 7 in the assignment (**npm install react-router-dom @types/react-router-dom**). This goes around the entirety of our App.

The Header **component** would stay where it is while the user is interacting with the App, but we want them to be able to move around the other **components**. This is done by wrapping those in **Routes** with an individual **Route** (both added as part of the react-router-dom) for each **component**.

The first one would take them to MenuRoute.tsx and in the URL of the app you see /menu appear.

If a user wants to visit a specific menu item’s details page, then you can get them there by adding the /details/:id code (think back to when we navigated the API in Postman).

The last one utilizes ‘\*’ as a wild card. This one Navigates to “/”, which would take the user back to the home page.



Text

Description automatically generated

In the BasicItem.tsx file we then add **Link** from react-router-dom. This helps populate the URL with the /details/ and then grabs the ‘id’ from the item associated. In the URL you would see the item’s id appear at the end.

Text

Description automatically generated

Then, in DetailsRoute.tsx we used the built in **hook** **useParams()** function that makes use of the ‘id’ and utilizes the **.find()** array method to locate the ‘item’ with that ‘id’ in ‘menuData’ (where the specific id, name, price, calories, etc. is stored).

Once this finds that into, then it returns a list with the name, description, or whatever else you want it to show from the JSON format properties.

Text

Description automatically generated

Text

Description automatically generated

In the HeaderRoute.tsx we added a simple Link tag that would take the user back to the main screen.

A screenshot of a computer

Description automatically generated with medium confidence

