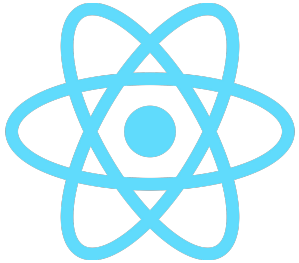


React.js

Lecture 3

Agenda

- Recap last lecture.
- Routing
- HTTP Requests using Axios
- Questions!

Routing

Routing is the capacity to show different pages to the user. That means the user can move between different parts of an application by entering a URL or clicking on an element.

React at its core is a very simple library, and it does not dictate anything about routing so you need to install it first :

`npm install react-router-dom`

https://reactrouter.com/en/main

Routing

<BrowserRouter>

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.

<Routes>

Rendered anywhere in the app, <Routes> will match a set of child routes from the current location, Whenever the location changes, <Routes> looks through all its child routes to find the best match and renders that branch of the UI.

<Route>

Routes are perhaps the most important part of a React Router app. They couple URL segments to components, each route should specify 2 mandatory props the path and element which takes JSX component.

Routing

```
<BrowserRouter>
    <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about-us" element={<AboutUs />} />
        <Route path="/list" case element={<List />} />
    </Routes>
</BrowserRouter>
```

Routing

- To Define **not found** page use the * for the path

  <Route path="*" element={<h1>Not found page</h1>} />

- To Define **dynamic segments** page use the **:property** for the path

  <Route path="/details/**:id**" case element={<Details />} />

Routing-Link

A <Link> is an element that lets the user navigate to another page by clicking or tapping on it.a <Link> renders an accessible <a> element with a real href that points to the resource it's linking to.

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

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

Routing - Active links [self study]

Most web apps have persistent navigation sections at the top of the UI, the sidebar, and often multiple levels. Styling the active navigation items so the user knows where they are

Using NavLink: is a special kind of <Link> that knows whether or not it is "active" or "pending"

```
<NavLink
        to="/"
        className={({ isActive, isPending }) => {
            return isActive ? "active" : isPending ? "pending" : "";
        }}>
        Home
</NavLink>
```

Routing-Routing hooks

- useNavigate:
  - This hook tells you everything you need to know about a page navigation, can use the navigate function to go from route to other like **navigate("/products")**
- useParams :
  - The useParams hook returns an object of key/value pairs of the dynamic params from the current URL that were matched by the route.
- useLocation:
  - This hook returns the current location object.
- useSearchParams:
  - The useSearchParams hook is used to read and modify the query string in the URL for the current location.
  - const [searchParams, setSearchParams] = **useSearchParams()**;
  - **searchParams.get('paramName')**

Routing- Having Layout [Extra]

We will need to wrap the routes with a parent wrapper that holds the layout element with the navbar and footer for example.

```
<BrowserRouter>
    <Routes>
      <Route element={<Layout />}>
          <Route path="/" element={<Home />} />
      </Route>
      <Route path="/about-us" element={<AboutUs />} />
    </Routes>
</BrowserRouter>
```

Routing- Having Layout [Extra]

And inside the layout component we can use the navbar and the <Outlet /> element from react-router-dom to be as a placeholder for the components that match the route.

An <Outlet> should be used in parent route elements to render their child route elements. This allows nested UI to show up when child routes are rendered.

```
function Layout() {
    return (
        <>
            <Header />
            <Outlet />
        </>
    )
}
```

Axios

Axios is basically an external library, which is used to make promise based HTTP calls

npm install axios

Creating instance for shared config :

```
import axios from 'axios';

export const instance =
axios.create({
  baseURL: 'API BASE URL',
});
```

Axios

After initialize axios instance you can use it for API calling to integrate with backend server :

```
instance.get('/endpoint_path')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

Let's check Axios Docs

Interceptors

Interceptors are functions that Axios calls for every request to transform the request before Axios sends it, or transform the response before Axios returns the response to your code.

https://github.com/axios/axios

There are two types of interceptors:

- **request interceptor**: this is called before the actual call to the endpoint is made.
- **response interceptor**: this is called before the promise is completed and the data is received by the then callback.

Interceptors

To use interceptors you need his two methods for request and another for response :

```
axios.interceptors.request.use(function (config) {
    // Do something before request is sent
    return config;
  }, function (error) {
    // Do something with request error
    return Promise.reject(error);
  });
```

Interceptors

To use interceptors you need his two methods for request and another for response :

```
// Add a response interceptor
axios.interceptors.response.use(function (response) {
    // Do something with response data
    return response;
  }, function (error) {
    // Do something with response error
    return Promise.reject(error);
   });
```

Interceptors

Interceptors can be used for example :

- Show loader in request method and hide it in response one
- Set authorization header
- Send repeated params with the apis
- Set Accept-language key based on user selected language
- Handling general errors returned from API

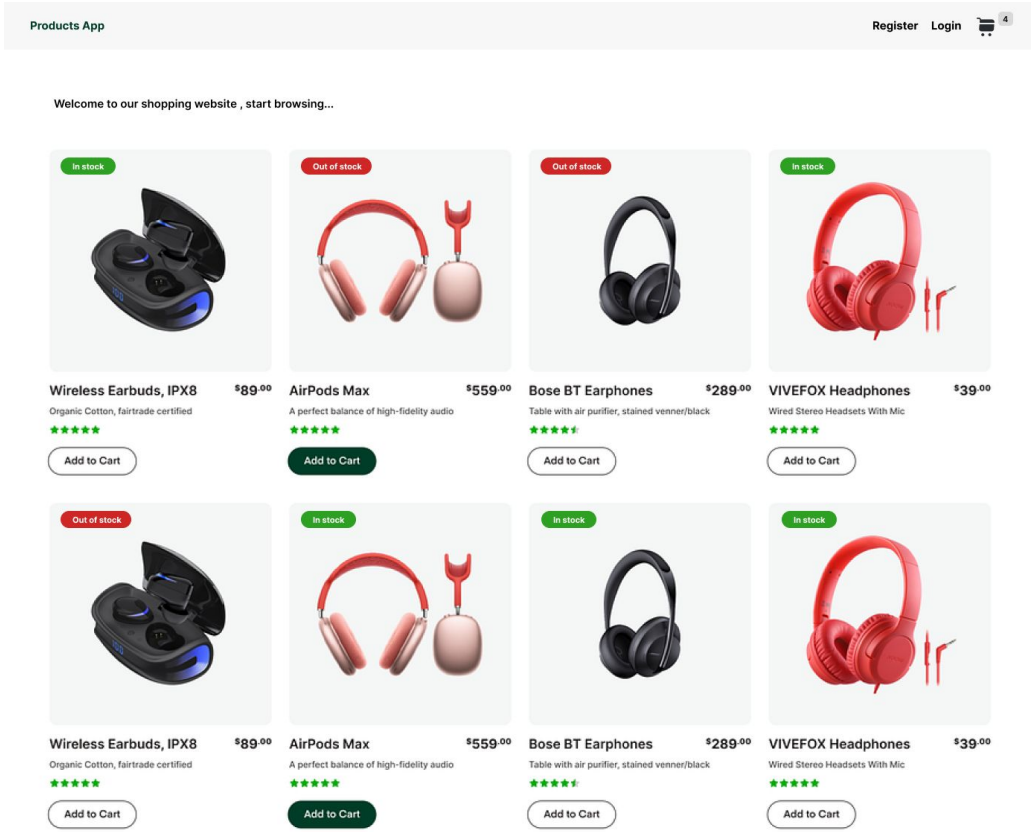
Thank you

Lap

Products App

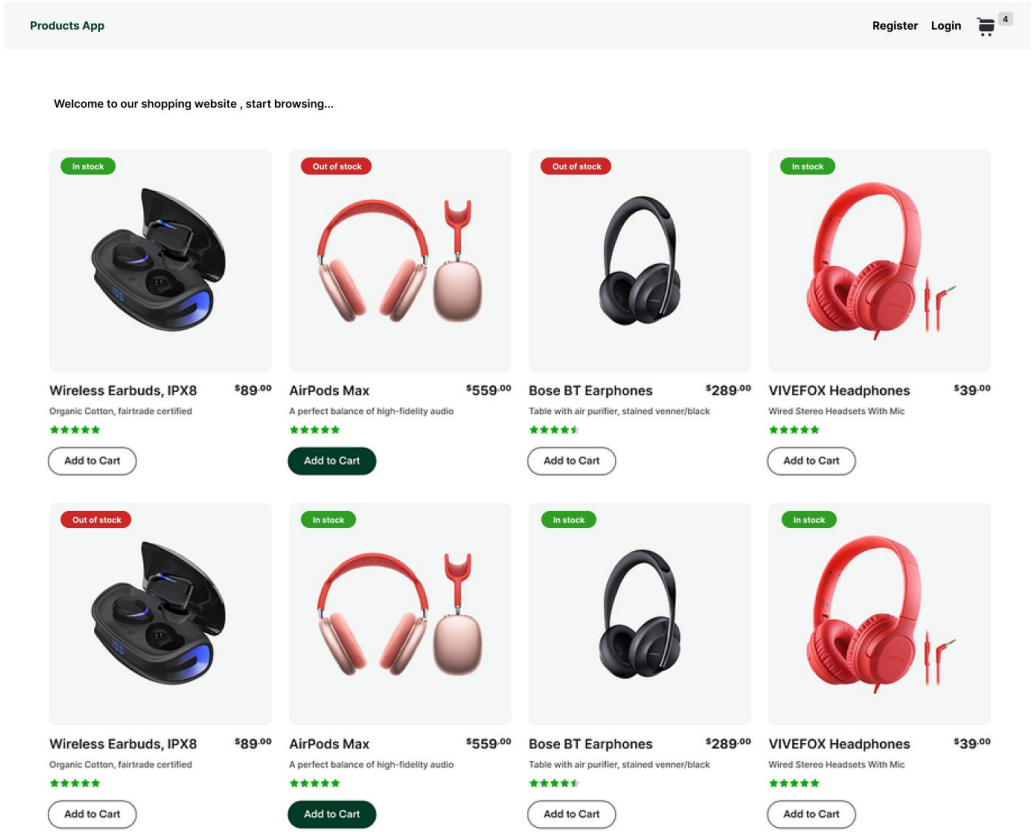
Create a new project with Routing Module

- Create a separate components for navbar and product list and product card
- Navbar will contain Products as a logo or text , Register and Login Links [not working for now]
- Render Products cards from the provided products array [attached]
- Each product card is separate component



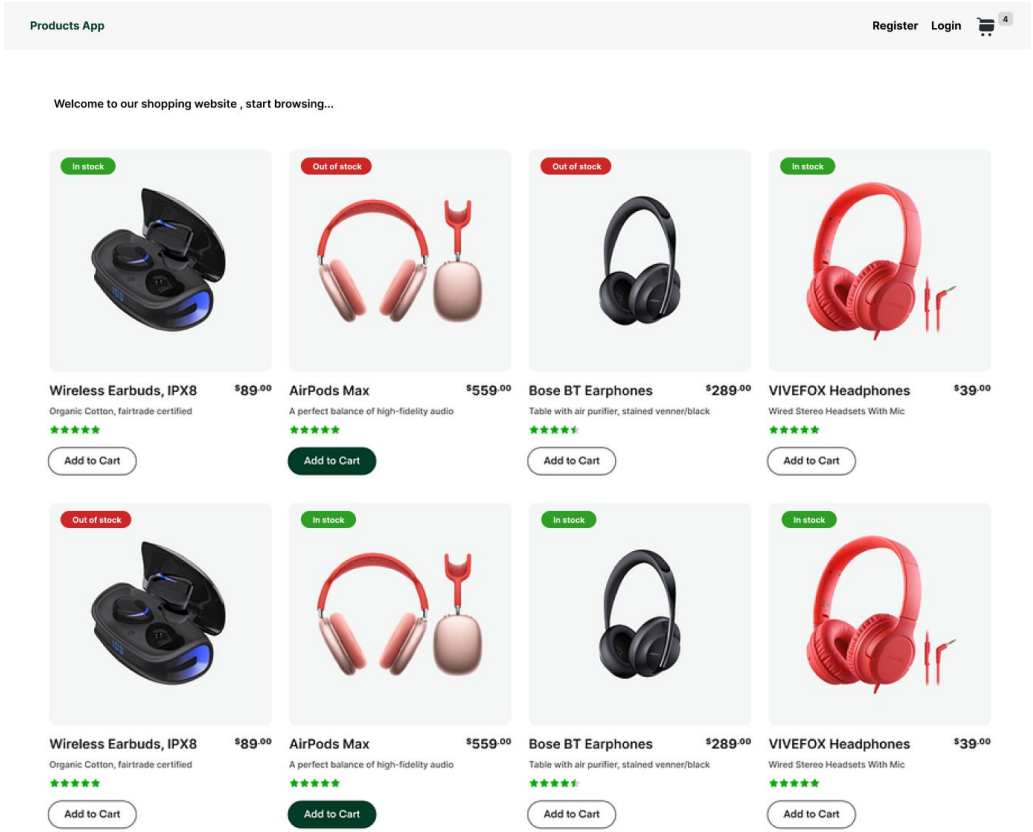
Products App

- Navbar will contain
  - Products page route and will be the default when open the project
  - Route for login
  - Route for register
- The active link should be colored with different color [like orange color for products as it’s the active route]
- Create a Not Found page
- Create a new route called **cart**



Products App

- if stock= 0 will return out of stock else will return in stock
- If stock= 0 => out of stock text will be red
- If stock> 0 => In stock text will be green



Products App

Each product card item should have:

- Product images
- Product name
- Product price 20 $
- Rate (Bonus if used stars to show rate)
- “Add to cart” button

Products App

- Create a product details component and route.
- When user click on any product card from the list he should be redirected to details page with the route
    - /product/:id [using dynamic routes]
- Filter the array of products with the product id you have from the route to get the single product data
- Pass the product details to the html to show data to user [Product images , title, category, brand, rating, description].
- If *discountPercentage* exists, show the product price before and after discount and the discount percentage