Shoply

Download starter JSON File

This is an exercise to practice React, Redux, and the React Router! We'll be building an e-commerce site where users can browser your inventory and add items to their shopping cart.

Part One: Displaying the Products

Don't worry about the router for this part, just focus on working with React and Redux.

We've provided you with a ***data.json*** file that contains names, prices, descriptions, and images for the products you sell. Create a new React app and move this json file into the ***src*** directory.

Make a page that displays these items in a single list, each with an "add to cart" and "remove from cart" button.

Allow users to add and remove products to the cart.

You should display the total of all items in the cart as users add and remove items from the cart. Note that users should be able to add more than one of the same product to their cart!

You should be storing the cart state in Redux as well as the complete product inventory (to do this, you should import the JSON file into your ***rootReducer*** file). At a minimum, make sure your state object has these two keys in it!

Make sure your action creators and reducers have tests before moving on!

Make sure you have styled this page nicely before moving on!

Part Two: Displaying Product Details

Your product listing page should just provide high level information on each product (like its name). For more detailed information, let's build a product details page.

Include React Router and make a ***Routes*** component that's rendered by your App. The "/" route should render your existing product listing page, but the "/products/:id" route should render a product details page. This page should include the image and description of the product.

Visitors should be able to add or remove copies of the product from their cart on the product details page.

Part Three: Displaying the Cart

It's time to start displaying the cart! Add a route to "/cart" which renders a component that displays all of the items in your cart. You should be able to also add and remove items at this point — including changing the quantity of items (for example, to change "1 tv" to "2 tvs".

Make sure you have styled the cart nicely before moving on!

Further Study: Discounts and Taxes

Now that you have routes to show the cart, include a form on this page that when submitted will apply a discount to the order total. Include the following as valid discounts:

- *REMOVE10*: applies a 10% discount
- *REMOVE20*: applies a 20% discount
- *REMOVE30*: applies a 30% discount

Make sure that you can not add more than one discount at a time and display an error message to the user if they try to add multiple discounts.

More Further Study

- Add a NavBar to your app to help improve the UI. In the Nav should be links to "/" and "/cart", as well as the number of items in your cart and the total cost.
- Add taxes. When you show the cart total on the "/cart" route, make sure your total includes the 7.25% California sales tax. Find a good place for this logic — don't bury it inside of a component!
- Store your cart in localStorage so when the page refreshes you do not lose the cart information.
- Deploy this application on Heroku. Deploying an app made with Create React App is very straightforward!
- When you add an item to the store, attach a property numRemaining which will be decremented by one every time that item is added to a cart. If the numRemaining reaches zero, do not allow users to add that item to their cart.
- Make sure your application looks good on devices of different sizes.
- Write some component tests using React Testing Library!
- Read about the combineReducers function in redux, and refactor your redux logic to combine at least two reducers together.
- Add a backend! The ***jsonserver*** module from last weekend's assessment would be nice to include here, if you don't want to go through the trouble of building out an entire backend application.
- Add a ***/secret_discounts_page*** route that allows you to add or remove coupon codes. Any coupon code you add should then work as expected on the ***/cart*** page.
- Add a ***/products/new*** route that allows you to add a product to the inventory.
- Add "save for later" functionality to your cart. Products that are saved for later don't count towards the total price you'll pay (similar to the behavior on Amazon.com).

Solution

View our solution here