

Website Link: https://victor-grajski.github.io/assignment_6/
GitHub Repo: https://github.com/victor-grajski/assignment_6

Reflection

The main challenge I encountered when adding the remove from cart functionality was in deleting the correct item. The first approach I tried was to pass in the item to my remove function and then remove that whole item from my cart using `.pop()`. However, that didn't seem to work because a different item than the one I clicked on got removed. I then decided to pass in the `cartID` and filter my `cartItems` list to get the index of the corresponding item. I would then remove that item using `.pop()`. However, the wrong item was still getting removed so I dug further into the removal mechanic. Upon reading more documentation, I was reminded that `.pop()` always removes the last item in a list, and I needed to remove a specific item. Thus, I used `.splice()` to remove the item at the index I had determined was correct, and after that, the correct item got removed! Moving forward, I will remember the difference between `.pop()` and `.splice()`, and I will know when each is appropriate to use.

Programming Concepts

1. **Conditional Rendering:** In my Cart component, I used conditional rendering that checked if my `cartItems` list had any items. If `cartItems` did have items, then it rendered them as one would expect, but if it did not have any items, then my component rendered a div that told the user their cart was empty.
2. **Object Destructuring:** In my Product component, I used object destructuring to render the values of each product by its property name alone rather than having to add "product." before each property. This allowed my code to be cleaner, less redundant, and clearer to a first-time reader since I declared all the properties I was going to use at the top of my file.
3. **List Filtering:** In my App component, I created a `removeFromCart` function which used list filtering to find the object in my `cartItems` list that corresponded to the `cartID` I passed into the function. This allowed my code to be much less verbose, easier to write, and more efficient since I didn't have to create a for loop that iterated over the entire list until it found the item.
4. **React Hooks:** In my Product component, I used Hooks to track the color, size, quantity, and image states. This first allowed me to continue to use functional components instead of class-based components. Second, it allowed my state code to be modular which made it easier to manage since I didn't have to worry about tracking all the other state variables when updating any given state variable.
5. **React Context:** Across my app, I used Context to track global state variables like cart items, the main list of products, and it also allowed me to house and pass down my helper functions like `removeFromCart` from a central place. This mental model was much simpler for me to employ than passing props down the entire component tree since I could drop in props only when needed, and I knew I'd always have access to global state when needed.

New Page: Order Status

The user can reach the Order Status page by “completing” the checkout process after adding an item to the cart. After clicking “Place Your Order” on the Checkout page, the user can click “Check Order Status” on the Confirmation page.

Both the lo-fi sketch and the hi-fi mockup are documented below, and they are also stored in the “mockups” folder at the top level of my GitHub repo.

Order Status Page Lo-Fi Sketch

I kept the header the same to maintain consistency with the rest of the site. Next, I decided to display the three main phases of the order process with checkmarks to serve as visual cues of the order status. Next, I included the shipping address and the order total to remind the user of the most salient information from the order so they can spot any potential errors. Finally, I include a list of all the items that were purchased along with their size, color, and quantity to give users visibility into what they bought and spot any errors.

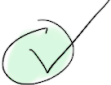
Muddy Paws

Order Status

Products


Cart

Order Placed




Oct 25

Shipped




Est: Oct 27

Delivered



Est: Oct 29

Shipping Address



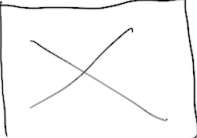
Order Total: \$xx

Subtotal: \$xx

Shipping: \$xx

Tax: \$xx

Items



Dog Harness

Size: Large

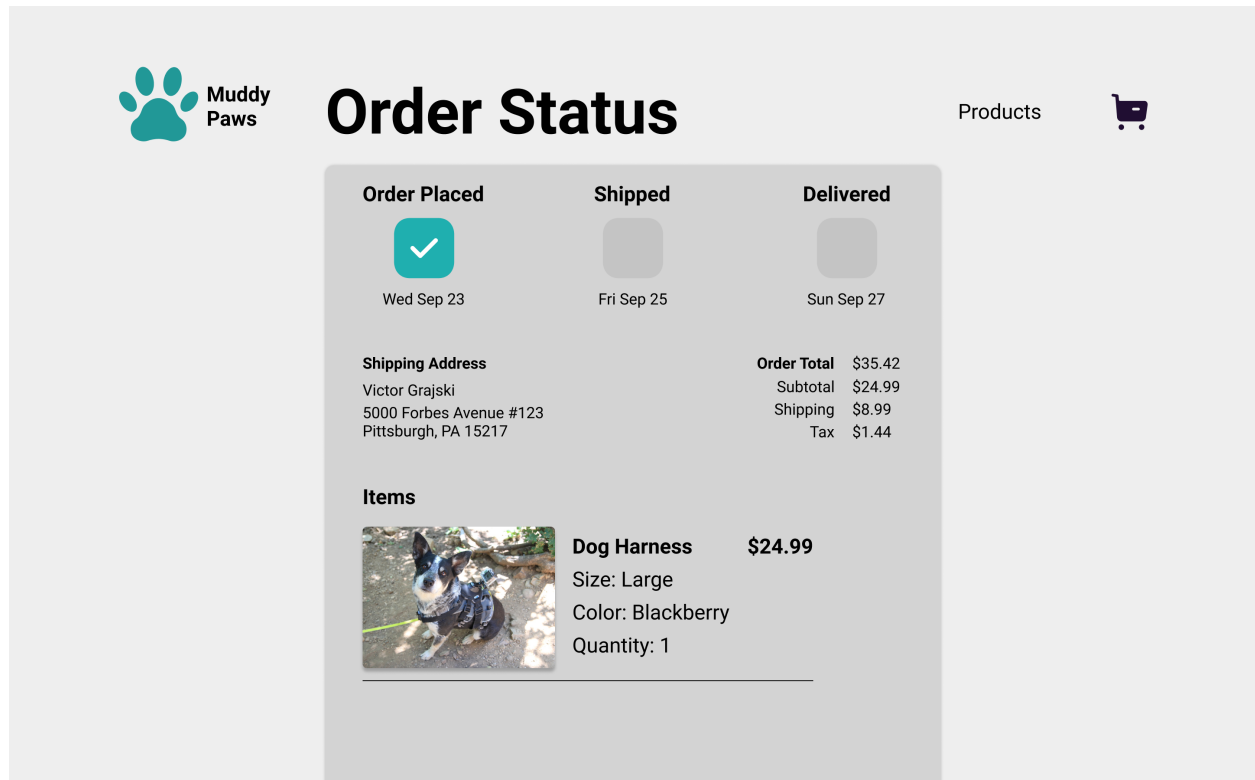
Color: Blackberry

\$24.90

Qty: 1

Order Status Page Hi-Fi Mockup

For my hi-fi mockup, I retained the overall visual look and feel of my other pages to maintain a consistent experience for the user. My header layout, fonts, colors and icons are the same. Next, I used the same grid as I did in other pages to maintain a consistent design. When representing the order status visually, I chose the same accent color as the rest of my site to maintain a consistent look. Finally, I used the same typographic hierarchy and the same general layout for my cart item, again, to maintain consistency.



[Figma Prototype Link](#)