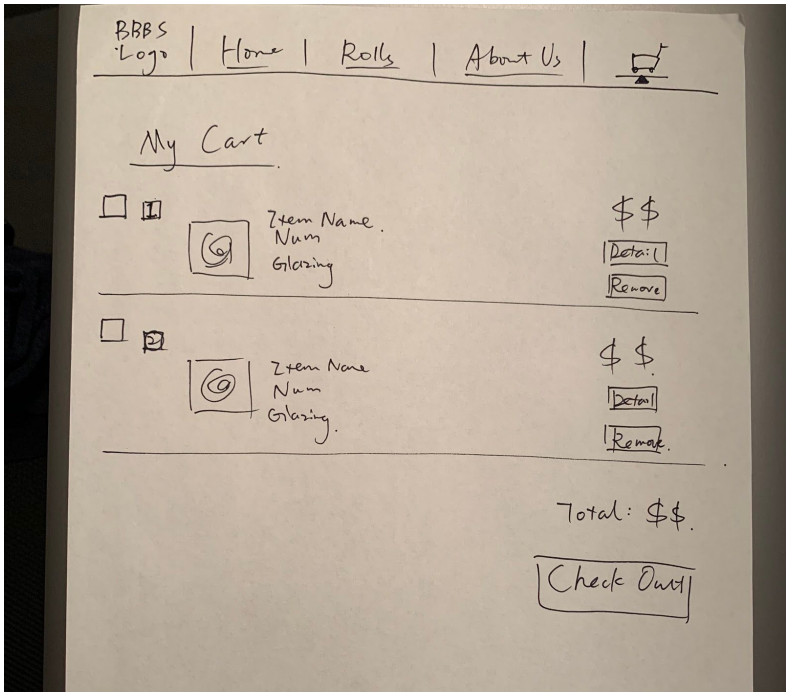
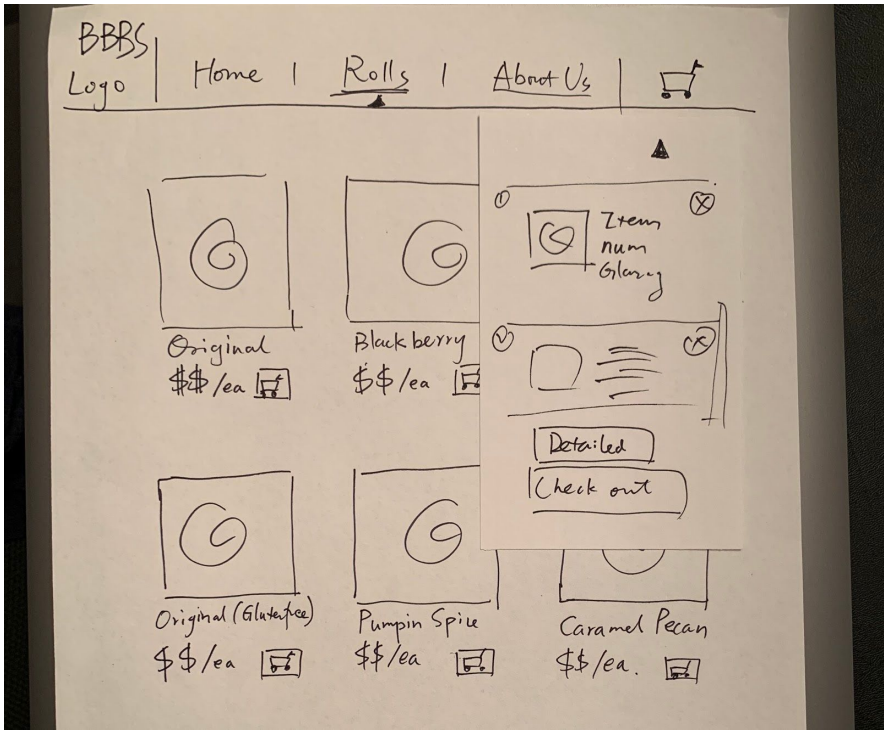
**HW 6**

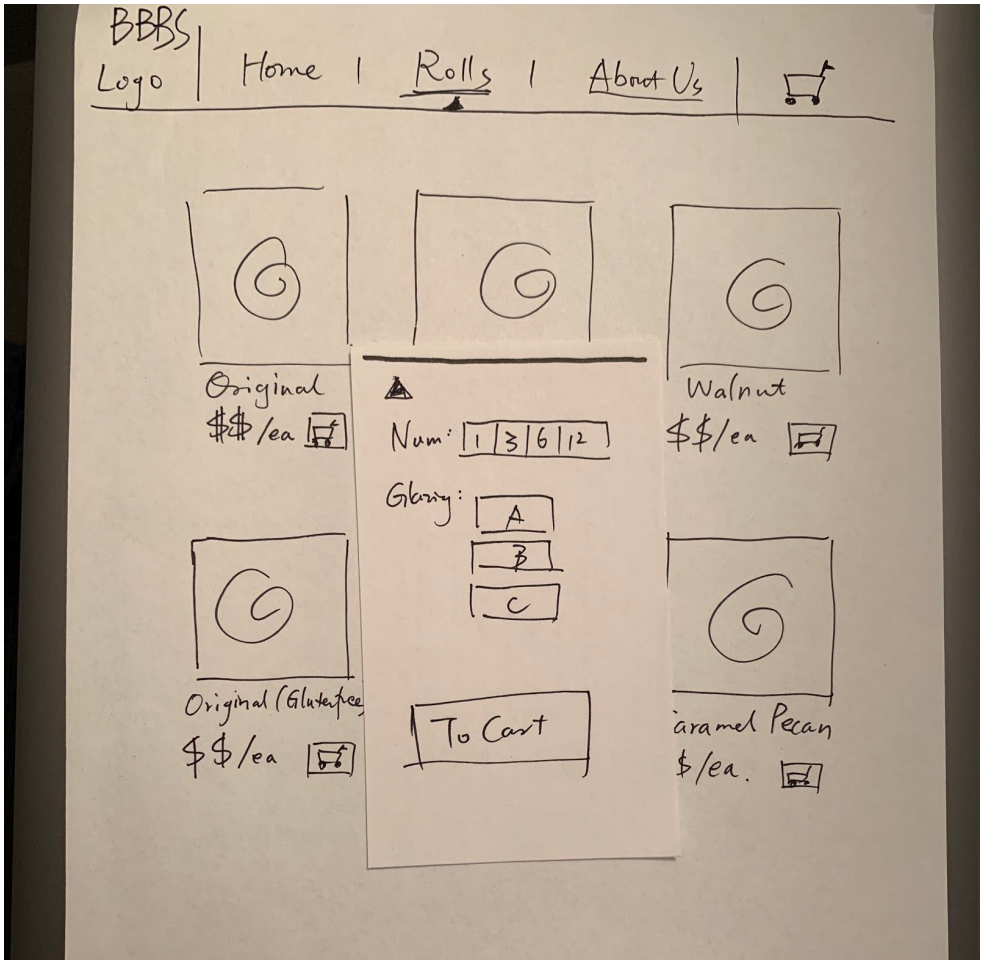
1. Low-fidelity screen for Shopping cart




a. On the shopping cart page, users are able to see the thumbnail picture, the name of the product, the number and glazing they have chosen, and the total price of the product as well as the total price for all the items in the cart. Users will have the option to delete the items from the cart.

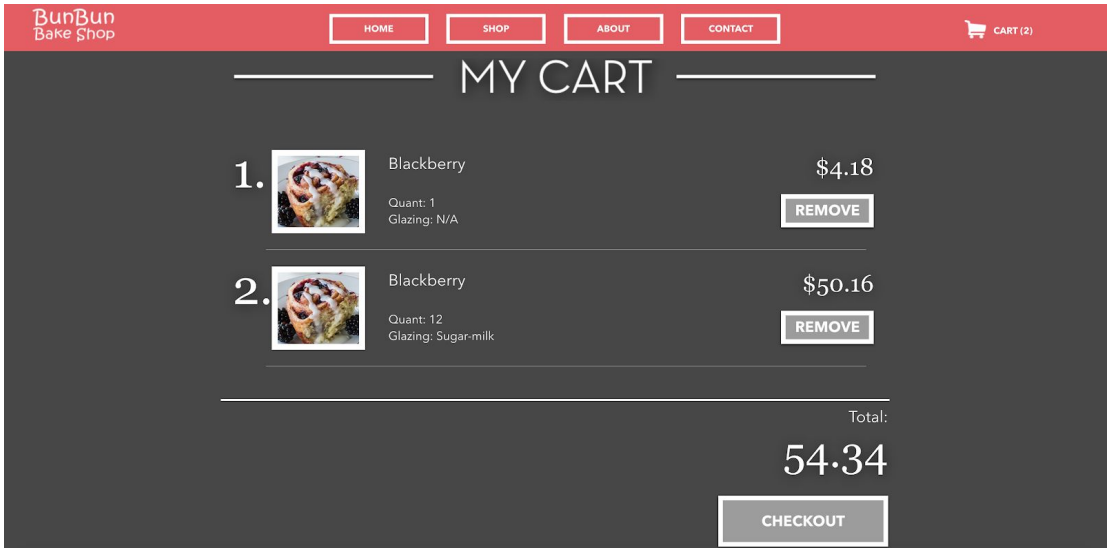


b. When users use their mouse to hover on the cart button on the top right, they are able to see a less detailed version of the cart. This will only still display important information such as item name, number, and glazing.

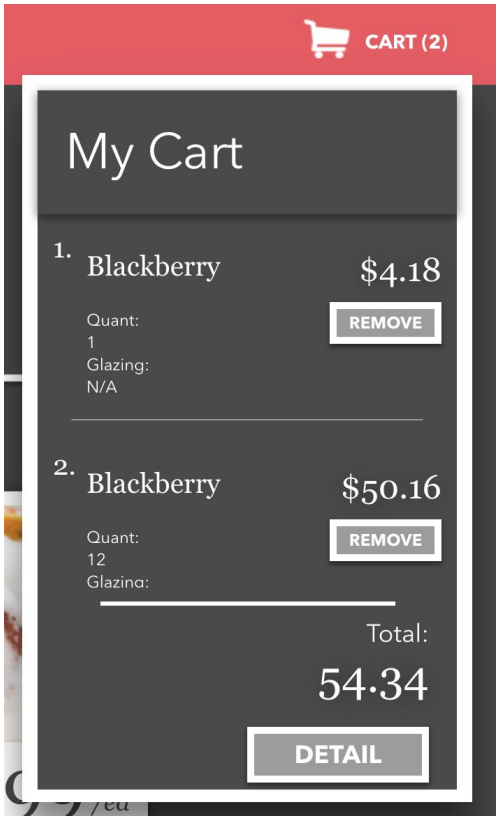



c. On the list of all item page, experienced users should be able to click the “add to cart” directly and choose number and glazing of the item without going into a detail page and read the introduction of the product.

2. High-fidelity screen for Shopping cart

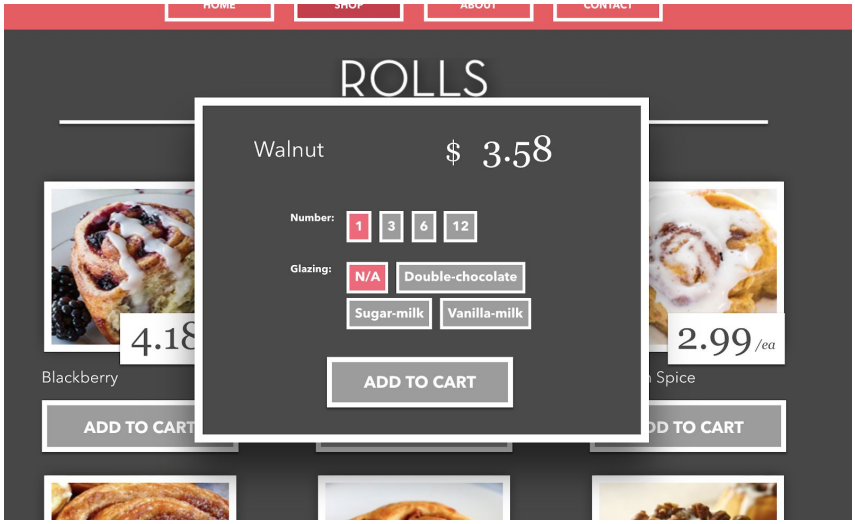



a. Not many changes have been applied to the cart page since the design idea presented in the low fidelity prototype is able to let users complete their tasks easily. The design also follows the convention of shopping carts that exist in other online shopping websites.




b. For the “mini” cart page that displays when users hover on the cart button, I deleted the thumbnail to give it more space, since, on the real screen, the original

design would take up too much space and would become too distracting for the users.




c. For the “add to cart” pop-up on the shop page, I made it wider so the content inside could have room to breathe. I also positioned it to the center of the screen so users would not have to scroll up or down to finish the tasks.

3. Reflection

When I was implementing the shopping cart function for the website, the biggest challenge I encountered was to allow users to delete a specific item in the cart. This seemed like a small problem, however, turned out to be linked directly with the structure of the cart itself.

Before the delete function was implemented, when tring to make the add function for the cart work properly, initially, I tried to construct the cart using the methods that was taught during lab. As the to-do list example, my idea was to insert the code directly into the HTML container that was supposed to hold all the entries of items. This method was no longer usable here since directly adding or deleting items using this method would only work when all the activities are happening on the same page. Shopping cart, however, requires transitions between multiple pages, and the changes that have been applied to one page would not be stored once the page is refreshed or revisited. In order to confront this problem, I saved the HTML code that contained all content of the cart to a string and saved the string to local storage. Each time a page is refreshed, the string will be retrieved from the storage and inserted into the empty container that is supposed to hold all the item data.

This was when deleting Deleting an item became a problem. Not only would I have to locate the code for this item in the string that held all HTML code for the items in the cart, I would also have to identify which specific delete button had been called. I ended up

utilizing the substring method of javascript to cut the code belonging the item user wishes to delete out of the string.

What happened after this, however, was not an easy task for me as well. Since in my design, I numbered each item in the shopping cart by their sequence. In this case, when an item is deleted, sequence numbers of the items following that deleted item would have to be updated. I ended up using a for loop to traverse deep into the child nodes that hold the data of the sequence numbers of the items and update them one by one.