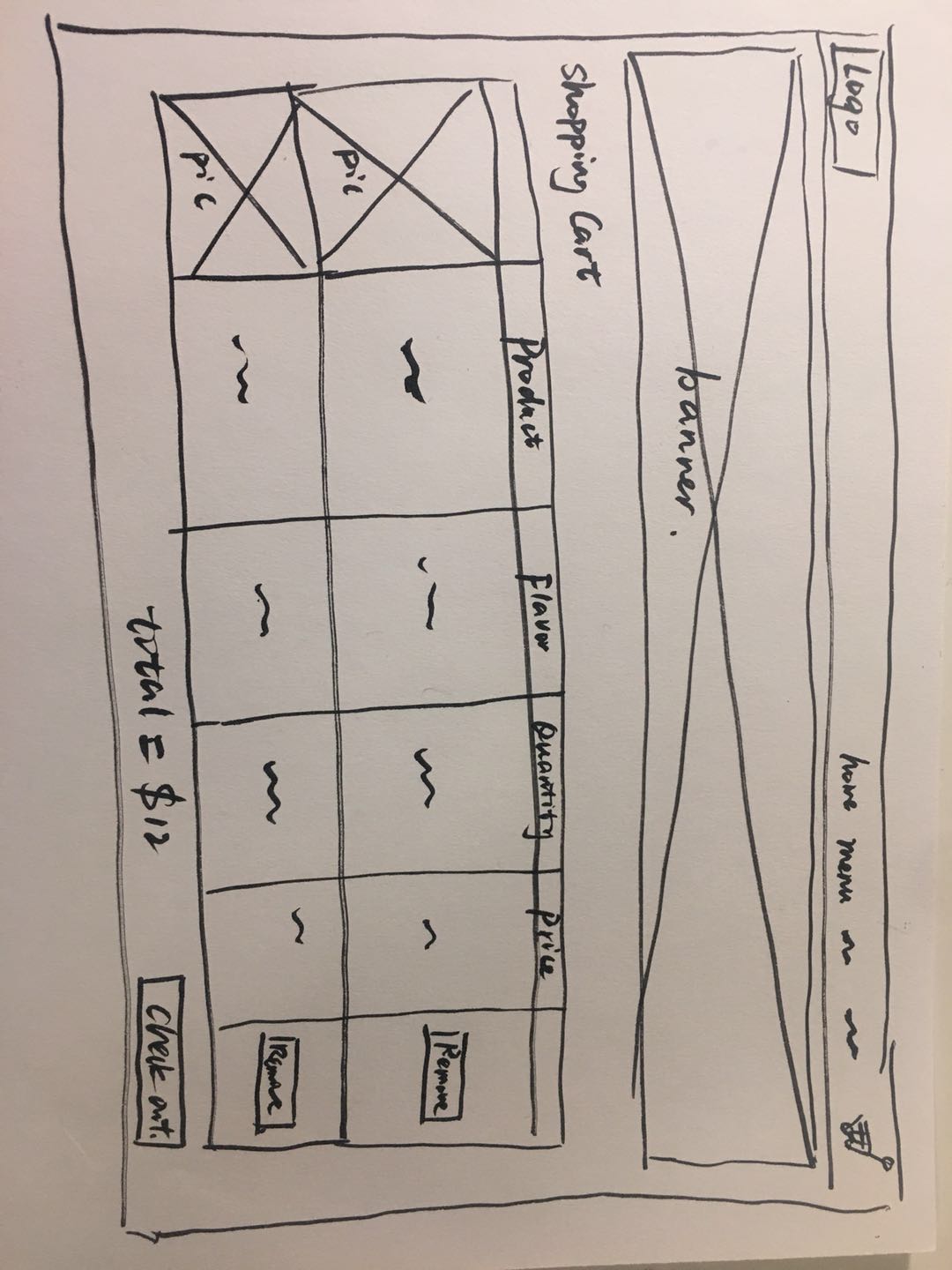
**Design decisions of new shopping cart page**

Low-fidelity Mock up

The new shopping cart page mainly consists of a table that lays out the detail of products in shopping cart. Both the price of a single item and a subtitle price is shown so that to provide a clear understanding of the breakdown of price. I chose this kind of layout because it’s most effective of showing information that has multiple attributes, and it will also be more flexible when adding on more items to this table.

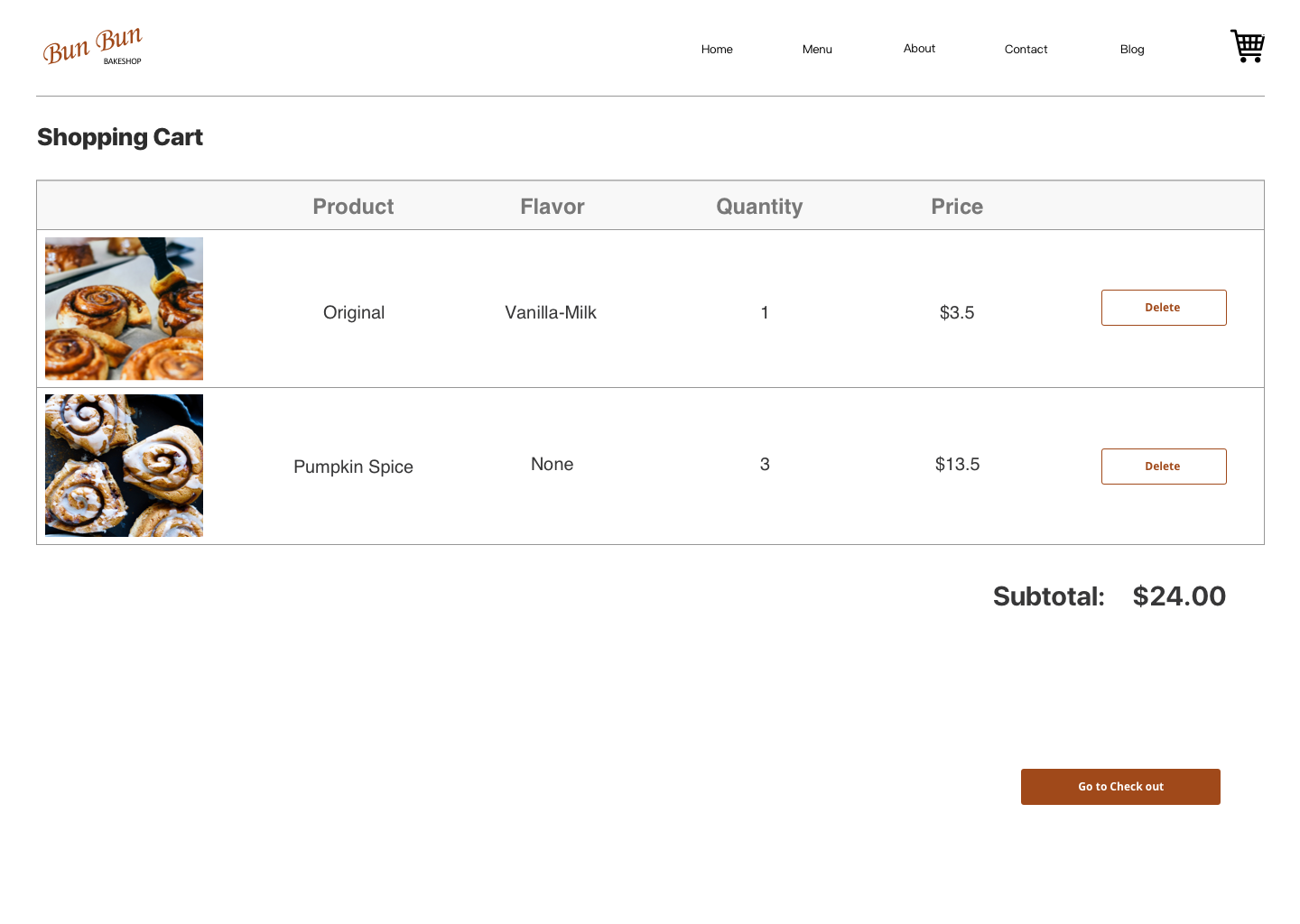


Hi-fidelity Mock up

I also designed the two types of button to have different styles:

1. The “go to check out” button is designed to be bold, in order to draw user’s attention and poses an inviting look to it, in order to subconscious persuade user to move on to check out.

2. The “delete item” button is much lighter, with white background and thin border. This function is necessary when user wish to remove items (user control and freedom), but also if they are designed to be bold, it would draw too much attention, thus could potentially invite user to delete items more often.



**Reflection:**

1. What challenges or bugs did you encounter?

During the programming of the shopping cart function, I identified that there were three important and also harder to implement functions: 1. Add items to a local storage so that computer remembers what users have decided added to cart. 2. Pull data out of the storage and still remain their integrity as properties of different items. 3. Summarize these data and display them in a cart, with function of deleting them from the storage, which will reflect on the interface as deleting item from the cart. These functions are very new to me, because before when only using html and CSS, there’s very simple logic, and with implementing JavaScript, these functions now have more backend operations. This reminds me of the MVC model we learned in lectures, and this project involves working more with the “model”.

While coding, I often encounter problems that has no direct error feedback, because when using html and CSS, the changes made to the interface is much more visible and mistakes are more easily discovered, but working with the data in storage, it’s harder to identify what went wrong.

1. How did you overcome these challenges?

To first define the logic behind the functions need to be identified, I familiarized myself with local storage, and how to store arrays to JSON. In this process I find the storing array in JSON most difficult, because JSON only store strings. To solve this problem, when storing array into JSON, I first stringified the array data, and when pulling the objects and their proprieties out of JSON, use the function JSON. parse and convert they back from strings.

To alleviate the invisibility of error problems of working with more backend data, I tried to implement more functions for feedback and constantly monitor them in the web inspector, for example looking at the changes happening in local storage, and using *console.log* function to constantly validate code.