Note: I was unable to properly link images in the shopping cart page on Github. In order to see those images, you can open my website at this link:
https://vv051.csb.app/

Reflection
Programming this website was definitely not smooth sailing, and there are still bugs I have to fix. One major bug I encountered was getting my local storage functions to work. I initially didn't understand where I should be implementing my local storage functions that would fill up my cart page. This was when I realized that 1) defining a function is not the same as calling it, and 2) Javascript runs sequentially. Onload() exists for the programmer to add functions that are called as the page loads. So for anything I want to add right as the user enters the page, it has to be done in OnLoad().

The location where you add the onLoad script in your HTML file also matters as well. Initially, I stupidly added this script tag in a parent class for one of my containers. This meant that any code that exists before the script tag would not be interactive. I also did something similar when I was linking my javascript file to my HTML document. Before I learned about the async function, adding the tag at the end of the document means you would have to wait longer for sites to load. In the future, I know to add any script tags in the head or body of my document.

Concept 1: Javascript Local Storage and the difference between onLoad() and Read() functions
Based on the bug that I encountered, I learned about the difference between the Ready() function and Onload() function. Because I followed a youtube tutorial that didn't utilize local storage, it used the document.ready function. The difference between this function and what we were taught in class was that the ready method checks if DOM elements are all loaded, while the onload function checks if everything including the DOM elements as well as images are loaded. This explained why some eventListeners that interacted with my images would sometimes not work properly. This was because the event listener was in the wrong place.

Concept 2: You can use for loops on DOM elements
Before I knew to use for loops, I would create separate eventListeners for every DOM element. This was especially inconvenient for me when I was trying to implement functionality for the remove-item-from-cart buttons. I realized that it was really important to label our elements as either classes or ids. Making all my remove buttons belong in the same class allowed me to loop through them to add event listeners. This was also really important to realize: getting an element by class name returns an array while getting an element by ID returns one object.

Concept 3: Future-proof your code by using Object Oriented Programming
Because I was learning a lot of things as I was coding this website, there were a lot of moments when, while trying to get something to simply work, I created some very janky solutions. One example of this was when I tried to get my images in my cart to show. I did this by getting the background-image url of the product page, and modifying the returned string to make it usable

for when that image url is added to the HTML element. Instead of using Objects in local storage and using HTML code instead, the process of debugging any issues was really complicated because I would have to go through the stringified HTML code to find mistakes.

Concept 4: Building Programmable User Interfaces.
As a design major, I always strive to make the most aesthetically pleasing websites. I have learned throughout this process that sometimes the prettiest websites are the hardest to code. Through the combination of using CSS grid, large images, and a complicated column structure, my HTML code was not only difficult to debug but also contained many different classes and ids that I would often have to go back and remember whenever I want to work with it in Javascript.

Concept 5: Sometimes it is more optimal to use inline Javascript
Although I might be wrong and there is definitely a better solution, there were some bugs that I came across that were easily solved using inline javascript. I have an element in my nav bar that tells the user the amount of items that are in their cart at any given time. I was having trouble getting this to work in my .js file but I realized that I could just add in a function that calculates the value based on the length of my cart array and place it in the HTML file so I wouldn't have to worry about when it was going to be executed.

RESOURCES USED

https://www.youtube.com/watch?v=YeFzkC2awTM&t=2191s
Javascript shopping cart tutorial for beginners. Does not include localStorage functionality.