Code Refactoring

“CycleScene”

Css

I. Going back over my previous projects I can see now I did use too much custom css. Partially because I really like writing css rules and I am intrigued by how it works. But ultimately someone would have to maintain and debug the code so I made a conscious effort in this project to change it. Although it might not seem that I wanted to use less css at first glance from my base.css! I tried to reuse rules as much as I could, an example here would be this one.

```
.fa-search,
.starcolor,
.results-number,
.toast-shopping-bag,
.product-rating,
.fa-cog,
.fa-tools,
.fa-ambulance,
.fa-check,
.order-number,
.register,
.sign-up,
.sign-in,
.fa-envelope,
.profile-name,
.fa-user-cog,
.bicycle-admin {
    color: orange;
    border-color: orange;
}
```

In previous projects there would probably be 10 or so rules all doing a similar task.
Similarly, to get away as much as I could from custom rules I relied heavily on the bootstrap 5 padding alignment and margin classes. This had the advantage of being a huge timesaver in the end without having to go back and find custom css rules to change and debug, as well as the classes being obvious to anyone familiar with bootstrap.

Python

I. I must say after the end of the Ms3 project I did struggle going from writing a huge amount of JavaScript in my MS2 to writing python with its strict indentation and formatting rules in MS3. So much so I came out of Ms3 not really liking python for these reasons.
   Only now spending so much time in MS4 testing, taking apart all the code from the BA project I can see why python is such a brilliant backend language and now I see the strict formatting as a plus since it discourages you from writing messy code.
II. As part of the testing and troubleshooting I used a lot of 'print' statements in the various functions to see what the code was doing at particular times, removing these cleaned up the code by another 50 characters.
III. Following on from the Ms3 and seeing how unruly long user messages could make the code. I went back over and refactored any longer message and split them up on different lines also so they read better.
IV. Towards the end of the project I ran a linter for the python file. I used 'pylint' based on my mentor's recommendations rather flake 8. This was installed into Gitpod at the terminal using the command 'pip3 install pylint'.

    The linter picked up a lot of indentation issues, whitespaces and some unnecessary code like unneeded 'else' statements. In addition, I wrapped any code in brackets or spilt the line at the appropriate point to fix linting errors. I also added proper docstrings to all functions as well as identifying a couple of unused variables. There is however some warning on some files that are just project specific and were not deliberately fix for this reason.

JavaScript

I. I originally wrote a time module which I very proud of which was to be display on certain pages just as a feature, but on running lighthouse it turned out to be a big performance hit. Sadly, I removed it from the final draft of the project, but on the plus side I gained performance back.
II. Similarly, from playing around the with the brandcarousel.js config file there were some settings that made each page it was loaded onto

sluggish. It took me a while trying different combinations to find settings that I was happy with visually and that didn't have a big performance hit so this was another good result, and performance is one reason why the carousel doesn't feature on mobile version since I don't think there is anything the user would gain from it on a small screen. It really only works on tablet and above screen resolutions.

III. From following the videos and looking at the overlay used to hide the main content, I used a similar approach but I used JavaScript. This worked great but again unfortunately I discovered all this content was being loaded behind the scenes every time and was a huge performance hit. The script was done away with and you can see how It was dealt with it in the html section below.

HTML

I. Following on from the last point on the overlay to hide the body of the content on the main page each time. This was simply fixed by separating out the main body content from the header and footer. So the main body content now resides in the home app index.html and just the header and footer content now live in the base.html, giving a huge performance boost and simplifying the base template.

II. Going back over my previous projects I was conscious that I did use some depreciated tags, namely the
, <u> and <b> tags. Again I made a conscious effort to either replace them with their modern equivalent tags or bootstrap classes.

III. Again in previous projects I was conscious of the 'get it working and fix it afterwards' trap I had become a Vitim of! On this project I made conscious effort to regularly validate the html and write it correctly from the start rather than finding a quick fix for it at the end of the project, this again saved a lot of headaches and time.

end