Testing

Pull Requests:

We were fortunate enough to have a dedicated QA Tester (Emma). Emma handled all of our testing. Individually we created a detailed request stating what feature that is being added and described how to test and what the expectation should be. She would receive the pull request and follow the testing instructions as well as using test of her own. If there were any issues she would report them. The person that created the pull request would then fix the bugs within their feature.

Testing During Development:

During my development of the feature I would do my testing as I'm writing the code. I dealt with both the front end and the back end. One of the features I created was the closet view on a user's profile. There was two parts to making this happen. I had to grab the items of that user which included the item details and item images, each are tables in the database, then display them on the website. To verify I was grabbing the data correctly I would print the current user that's logged in then print the correlated items and associated images via python. I cross reference the print statements with the database verifying the data matched. Once I verified I moved onto displaying the data. Testing if the items display correctly was very straight forward as I made just the printed items and images matched what was visibly shown. I then went to the extent of adding an item to the that user and verifying that it showed in the closet.

Example Module (Closet Item View):

- Equivalence Classes:
    - Item:
        - Category
        - Title
        - Trade type
        - Image

- Test Cases:
    - Case 1:

        - Print item data and verify it matches user's items in the database
    - Case 2:
        - Add item to user's closet and verify that data is printed
    - Case 3:
        - Matched displayed item details and image to printed out data
    - Case 4:
        - Add item to user's closet and verify it displayed in closet item view
    - Etc.