

Storefront Testing App

**Coding Assignment**

CLOUD 9

Website Bug Implementation

University Of Kentucky Spring 2020 CS499

Customer: Ben Fox, Trissential



Team members

Alex Tanner

Alex Reel

Benjamin Ellis

Patrick Yoder

Camron Savage

Yankai Zhao

# Introduction

For this project we were to edit an existing E-commerce website for Trissential. The initial creation of the website was to provide Trissential clients and employees with a training/demonstration platform for quality assurance and test automation by giving admins the ability to select/add bugs to users. This allows their employees to find the added bugs and take additional action while showing clients the quality of service. Our objective as Cloud 9 is to enhance the pre-existing website with a more professional appearance for employees and clients while also providing new unbefore seen bug options.

1. Implementation
   1. Source Code

For the overall development of this project we used a web-based platform called GitHub. GitHub is used for version control and allows us to store our code for access to all team members of CLOUD 9 allowing for simultaneous project development.

Source code can be accessed through this link:

<https://github.com/GreenAlex96/StorefrontTestingApp>

* 1. Quality Review
     1. Methods

To ensure the quality of our additions to the project, we took a few steps before merging branches to the master version. This included having other team members look through the submitted changes in code to find any major errors and to understand the basics of what the changes were. Upon verification here, we merged versions and verified that we did not break existing features or new features through manual testing of all functionality related to the update.

* + 1. Issues Found

In checking the quality of our updates, we were able to avoid any major issues. The largest issues we discovered involved database changes which occured when new features added columns to the database which did not exist on other users' systems. We resolved this issue by updating existing databases to match the new one. Other than this, we did not discover any major problems with functionality. Some issues did exist with style choices which were easily resolved without problem.

# User’s Manual

**Note:**

All website addresses in this manual are based on the assumption that the website is being run off of a local server on port 8080. When the website is eventually hosted on a proper web server, the “<http://localhost:8080>” portion of the website addresses should be altered to reflect the proper address.

Basic User

**Setting Up**

To access the website, you must have an account registered in the database. In order to do this, navigate to <http://localhost:8080/register.php>. Here you will enter the information you would like you use for your new account. This includes a valid email address and a password. Once the information has been entered and you press the “Sign me up!” button, you will be sent an email at the email address you just registered with. The email contains a validation link which you will need to follow in order to complete your registration. After following this link, you will be prompted to enter some personal information such as your name and address. Provided that the registration was successful, you will then be redirected to the homepage logged into your new account.

**Login**

Provided you have an account, navigate to <http://localhost:8080/login.php>. If you do not have an account yet, refer to the “Setting Up” section above. On the login page, enter the email and password associated with your account and click the “Login” button.

**Forgot Password?**

Below the login credential fields on the login page, there is a link that says “Forgot your password? Click here!”. Clicking on that link will allow you to enter the email address associated with your account and have an email with a reset password link sent to that email address. After clicking that link and entering your new password, click the link that appears below the form to go back to the login page.

**Navbar**

Once you have logged in, you will have access to the rest of the website. At the top of each page, there is a navbar which contains buttons that lead to the different pages of the website. From left to right:

* **Logo**Leads to the homepage.
* **Product List**  
  Leads to the product list, which allows you to filter through products and view the product pages.
* **Search Bar**Type a phrase into the search bar and click the magnifying glass button. You will then be redirected to a product list that only shows products that contain the phrase you inputted in its name.
* **Hello, “Name”**  
  Hovering over this will reveal two more buttons:
  + **View Account**Leads to the account page, which allows you to view and update your account information.
  + **Logout**Logs you out and redirects you to the login page.
* **Orders**Leads to the order history page, which allows you to view your past orders and get in-depth information on each one.
* **Wishlist**Leads to the wishlist page, which allows you to remove products from your wishlist and add products on your wishlist to your cart.
* **Cart**Leads to the cart page, which allows you to checkout products in your cart.

**Homepage**

To access the homepage, you can either click the logo button at the far left of the navbar located at the top of the page or navigate to <http://localhost:8080/home.php>. This page contains a slideshow with captions that can be navigated by either clicking on the forward and back arrows or clicking on one of the grey dots below the slideshow.

**Product List**

To access the product list, you can either click the “Product List” button on the navbar located at the top of the page or navigate to <http://localhost:8080/productList.php>. This page contains a complete list of all the products you can order from the website. The sidebar allows you to filter through the products in various ways. Firstly, you can sort by both name and price. After that, you can set both a minimum and maximum price for your product search. Lastly, you can choose to look at a specific category of product. Each product is listed separately, displaying its name, product description and price. Clicking the “View” button below any item will take you to its corresponding product page.

**Product Page**

After clicking the “View” button under a product on the product list, you will be taken to this page. Here you will see an image of the product accompanied by its name, description, price and how many units of it are in stock. There are also buttons to select the desired color, size and SKU of the product, provided there are multiple variants of the product. After selecting the variant of the product you would like to order, enter the amount you would like into the field near the bottom and click the “Add to Cart” button. Alternatively, click the “Add to Wishlist” to save this product to your wishlist for ordering later.

**Account Page**

To access the account page, you can either hover over the “Hello, Name” button on the navbar located at the top of the page and click the “View Account” button that appears or navigate to <http://localhost:8080/accountPage.php>. This page contains a form that displays your personal information, such as your name and address. To change this information, simply fill out the form so that it contains the updated information and click the “Update” button at the bottom.

**Order History**

To access the order history, you can either click the “Orders” button on the navbar located at the top of the page or navigate to <http://localhost:8080/orderHistory.php>. This page contains a list of your past orders and displays their order ID and status. Next to each order is an “Inspect” button. Clicking on this button will lead you to another page that contains more specific information about that order, such as product names, prices, quantity and tax.

**Wishlist**

To access the wishlist, you can either click the “Wishlist” button on the navbar located at the top of the page or navigate to <http://localhost:8080/wishlist.php>. This page contains a list of all the products you have added to your wishlist. Next to each product there is a checkbox to select it. There are two buttons on this page, “Remove” and “Add to Cart”. The “Remove” button will remove all selected products from your wishlist while the “Add to Cart” button will add one unit of all selected products to your cart.

**Cart**

To access the cart, you can either click the “Cart” button on the navbar located at the top of the page or navigate to <http://localhost:8080/cart.php>. This page contains a list of all the products you have added to your cart. Each product has its name, price and a quality field that you can change to set how many units of the product you want to order. Also shown is tax and the total price for the order. If you change the quantity of any product in the cart, you will need to click the “Save” button in order for the price totals to reflect this change. The “Checkout” button will also update the cart, so it is not necessary to click the “Save” button before checking out your cart.

Pressing the “Checkout” button will take you to the checkout page which asks you to fill in your billing information, payment type and shipping information. You also have the ability to choose between different types of shipping before finalizing your order.

Storefront Administrator

**Note:**In order to be assigned this role, your account type must be manually changed in the database. If you are supposed to have this role but do not have any of the privileges described below, please contact the system administrator.

**Changes to Product Page**

As a Storefront Administrator, you have the ability to update products from their respective product page. This includes changing the product’s name, description, SKU, price, category, units in stock, color and size.

**Store Management**

To access the store management page, you can either click the “Store Management” button on the navbar located at the top of the page or navigate to <http://localhost:8080/storeManagement.php>. This page contains a form that allows you to add a new product to the store. Similar to updating a product in the product page, you will need to input the product’s name, description, SKU, price, category, units in stock, color and size. There is also another form for setting a store-wide discount.

Bug Administrator

**Note:**In order to be assigned this role, your account type must be manually changed in the database. If you are supposed to have this role but do not have any of the privileges described below, please contact the system administrator.

**General Changes**

If you are a bug administrator, you will be redirected to the admin page upon login instead of to the home page. However, you can still access the storefront by entering the URL of any page a basic user can visit.

**Admin Page**

To access the admin page, you can login as a bug administrator, hover over the “Hello, Name” button on the navbar located at the top of the page and click the “Admin Page” button that appears or navigate to <http://localhost:8080/admin.php>.

The admin page has three tabs for bug management purposes.

**Bug Assignment**

This tab contains a “User List” field with a drop down menu that contains the name of all users in the system. You can also type into the field to filter down the results. After selecting a user you can select a bug (or multiple by utilizing the control key) from the bug list to apply to the user. Also, select the start and end date for the bug (the end date will default to the start date if no end date is inputted).

Click the “Review Bug Assignment” button and the user with the bugs you selected will be displayed. After confirming that these are the correct bugs to assign this user, click the “Save Bug Assignment” button.

**Update Bug Assignment**

This tab contains a “User List” field similar to the previous tab where you can select a user. Upon selecting a user, the page will display a table of bug assignments. This table allows you to either change the start and end dates for a bug assignment or delete the assignment entirely.

**Bug Creation and Editing**

This tab contains a “Bug List” field that either lets you select an existing bug. Selecting an existing bug will fill in the bug information fields with the bug name, functional area, description and code block. If you want to change any of this information, edit the fields and then click the “Update Bug” button.

If you want to create a new bug, do not select a bug from the “Bug List” field. Instead, fill in the bug information fields as described above and then click the “Save New Bug” button. However, this will not implement the bug without some modifications to the php file of the page the bug is intended for.

If you are not a system administrator or are not working with one, do not use the “Save New Bug” feature.

# System Administrator’s Manual

**Getting Started**

MySQL and PHP are the two required assets to run this website on a server. Recommended is using the XAMPP package (<https://www.apachefriends.org/index.html>) which includes both PHP and MySQL and enabling both the Apache and MySQL services in it.

If you are on Windows, you will have to add PHP to your PATH. If you are not familiar with this process, you can refer to this tutorial <https://docs.alfresco.com/4.2/tasks/fot-addpath.html>.

After setting up MySQL and PHP, clone the project Github (<https://github.com/GreenAlex96/StorefrontTestingApp>) using your prefered method (Git, Github Desktop, Sourcetree, etc.).

**Email Setup (for XAMPP users)**

In order for the registration page and other pages that send emails to work you will need to set up your PHP and XAMPP sendmail configuration files to use the gmail mail server and send through the Trissential bug site gmail.

Find xampp/php/php.ini on your installation

In php.ini find [mail function] and change:

SMTP=smtp.gmail.com

smtp\_port=587

sendmail\_from = trissentialbugsite@gmail.com

sendmail\_path = "\"C:\xampp\sendmail\sendmail.exe\" -t" *// Replace with your XAMPP installation if not installed in root of C:\*

Now go to xampp\sendmail\sendmail.ini and change:

[sendmail]

smtp\_server=smtp.gmail.com

smtp\_port=587

error\_logfile=error.log *//uncomment*

debug\_logfile=debug.log *//uncomment*

auth\_username=trissentialbugsite@gmail.com

auth\_password=BugsAreFeaturesToo

force\_sender=trissentialbugsite@gmail.com

If you need to access the gmail account directly, here are the account details:

Account: [trissentialbugsite@gmail.com](mailto:trissentialbugsite@gmail.com)

Password: BugsAreFeaturesToo

**Note:**The webpages register.php, forgotpassword.php and reverify.php all have references to the gmail account [trissentialbugsite@gmail.com](mailto:trissentialbugsite@gmail.com). Also, the emails those pages compose send links that are based on a localhost:8080 server set up. These will have to be changed to reflect changes in the email used and server address.

Also, if the emails fail to send after doing these set up steps, you may need to log in to the [trissentialbugsite@gmail.com](mailto:trissentialbugsite@gmail.com) gmail account and go to <https://myaccount.google.com/lesssecureapps>. If “Allow less secure apps” is set to OFF, it may be blocking the webpage’s attempts at sending email. Toggle this setting to ON and try sending the email again.

**Database File Setup**

The website needs to have access to the MySQL database, so you will need to enter the MySQL root account’s login credentials into the files “dbConnect.php” and “database.php”. You will need to change the variable definitions for servername, username, password and dbname (database name) to match your implementation.

**Server Initialization (for localhost:8080 setup)**

Using the terminal of your choice (Windows Powershell from within Visual Studio Code is what we used for this project), navigate to the project folder and enter the command “php -S localhost:8080”. The PHP Development Server should start with a new console window.

**Database Initialization**

Once both the PHP server and database are running, you can now initialize the database. Simply navigate to <http://localhost:8080/index.php> and wait until you are redirected to the login page. Provided everything was set up properly, you should be able to start registering users and log in to access the website.

**Changing User Verification Status and Account Type**

Go to the database that you named in “dbConnect.php” and “database.php” and open up the authentication table. Each account in the database is listed here. Most of the fields are self-explanatory but the type and status fields have certain codes.

**Type field:**

0 = Basic User

1 = Store Admin

2 = Bug Admin

**Status field:**

0 = Unverified

2 = Verified

**Adding New Bugs**

After creating a new bug as described in the “Bug Creation and Editing” section of the User’s Guide or manually adding it into the database through the “bug” table, you will need to encapsulate the part of the code you would like to replace with the bug code.

**Note:**

The PHP file you want to implement bugs in must include “bugCheck.php” to prevent an error from occurring. If the file includes “header.php”, then including “bugCheck.php” is not necessary since “header.php” includes it by default.

If you don’t know the bug ID, you can either look for it in the “bug” table in the database or the “Bug Creation and Editing” tab of the admin page.

Example for implementing bugs in PHP code:

//\*\*\*\*\* Bug Start \*\*\*\*\*//

$bugCode = bug\_check(<bug id>);

if(is\_null($bugCode))

<Original Line/Lines of code>

else

eval($bugCode);

//\*\*\*\*\* Bug End \*\*\*\*\*//

Example for implementing bugs in JavaScript code:

//\*\*\*\*\* Bug Start \*\*\*\*//

var bugCode = "<?php echo bug\_check(<bug id>);?>";

if(bugCode == "")

<Original Line/Lines of code>

Else

eval(bugCode);

//\*\*\*\*\* Bug End \*\*\*\*//

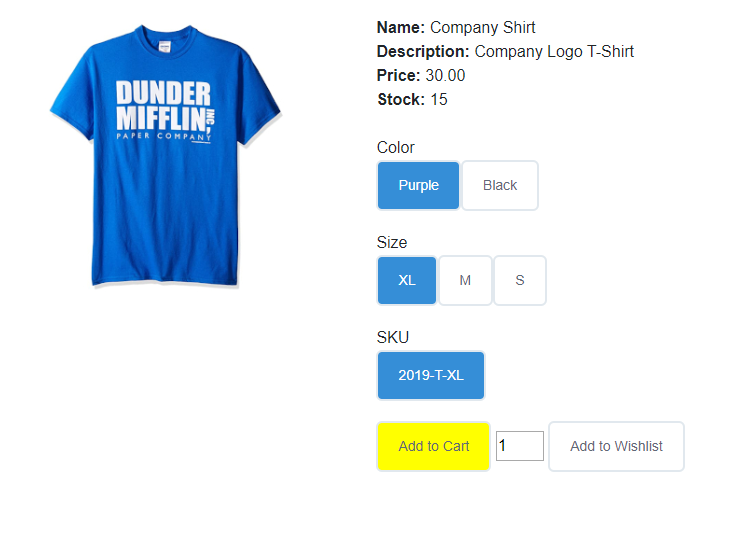
# Testing

Our initial testing plans involved utilizing test automation. However, after discussing this with our customer, we decided to focus our efforts on feature implementation and the sole use of manual testing. Testing involved verifying the functionality of individual feature adds, which can be seen below, as well as their behavior relative to each other and the existing application.

Review of the test plans primarily lead to clean up of our test cases and filling in a few gaps. We had to scale down our planned features, which naturally removed some test cases. Our initial list was lacking coverage on front page implementation, adding to cart from wishlists, removing items from wishlists, and store discounts.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Area** | **Test Description** | **Expected Result** | **Test Result** |
| Add to wishlist | Search for a product, click on the add to wishlist button. | The correct product has been added to the users wishlist. | Passed |
| Wishlist view | Click on the user’s wishlist | The page loads with the correct products. Wishlist products also appear in the database. | Passed |
| Remove from wishlist | Click ‘remove’ next to an item on a wishlist | The item is removed from the screen and database record | Passed |
| Add to cart from wishlist | Click ‘add to cart’ next to an item on a wishlist | The item is added into the user’s current cart | Passed |
| Store admin user | Create a user as a store admin, login as the new user. | The user is able to view and enter new pages to add and update products and set product discounts. | Passed |
| Add product | As a store admin, enter the add product page. Enter information into the form and click save. | A new product is added to the database with the appropriate information. The product is also visible by searching for it in the application. | Passed |
| Update product | As a store admin, select a product and click the update button. Change its information in the form field and click save. | The product’s information is changed in the database as appropriate. These changes are also reflected within the application. | Initially failed, resolved after discovery |
| Front page | As a user, log into the application. | The user is redirected to the new home page on successful login | Passed |
| Front page images | As a user, go to the application’s home page. | A slideshow of images are displayed that can be cycled through | Passed |
| Store discount | As a store admin, go to the store management page and set a store-wide discount. | The discount is set in the database and shown to the user in the product list | Passed |
| Store discount application | As a store user, complete an order. | The discount is reflected in the cart, checkout, and when viewing order history | Passed |

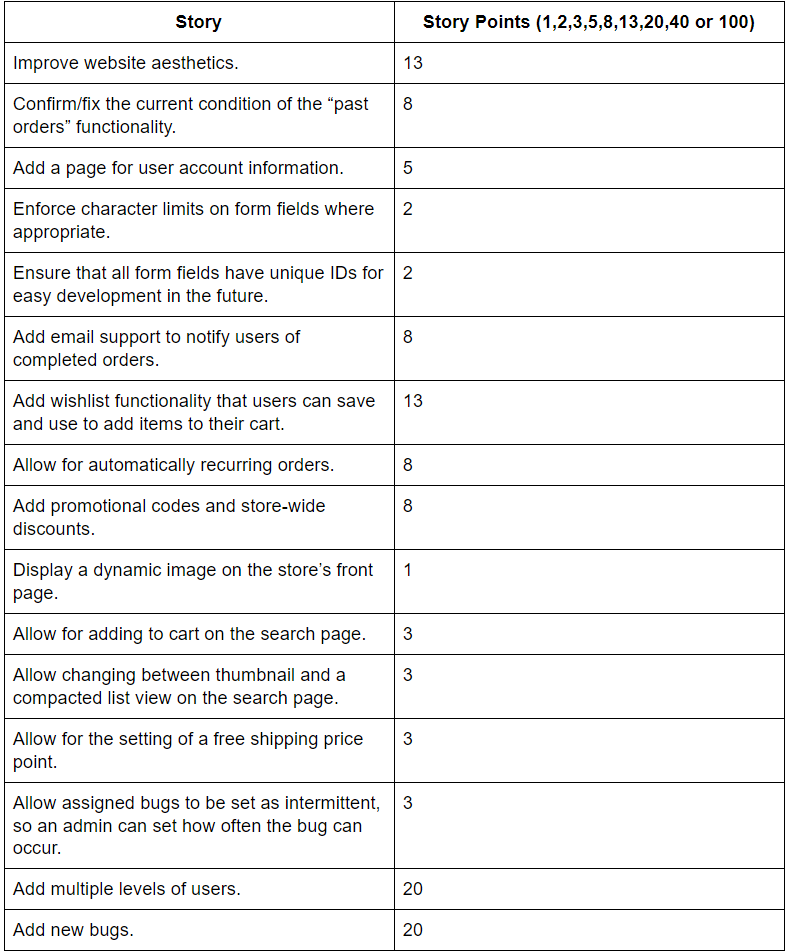
As you can see, the overall test results were positive. There was an edge case discovered that we did not have time to fix prior to this report. The application allows for products with the same name but different color, size, and sku combinations to link them to the same page. An example of this is below. Note that Every combination of color, size, and sku is a unique row in the database:



The problem arises when a product is added with the same name as an existing product but without a shared color, size, or sku with that product (in this case, a Company Shirt added with a Red color, XS size, and an sku of 123 would trigger this bug). When that happens, the new product is not accessible because there are no common attributes to link the two. This might require a redesign to how products are handled to solve cleanly. Another minor issue discovered was on the checkout page. The page’s form became misconfigured when adjusting its styling. This issue was quickly resolved.

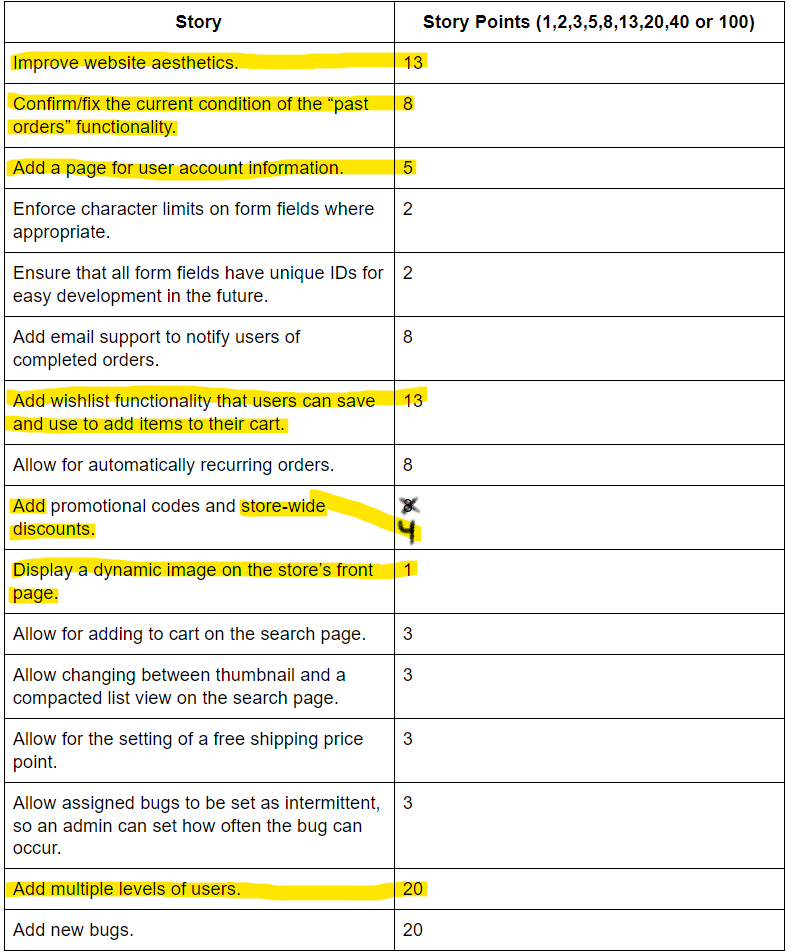
1. Metrics

These were our initial story point estimations.



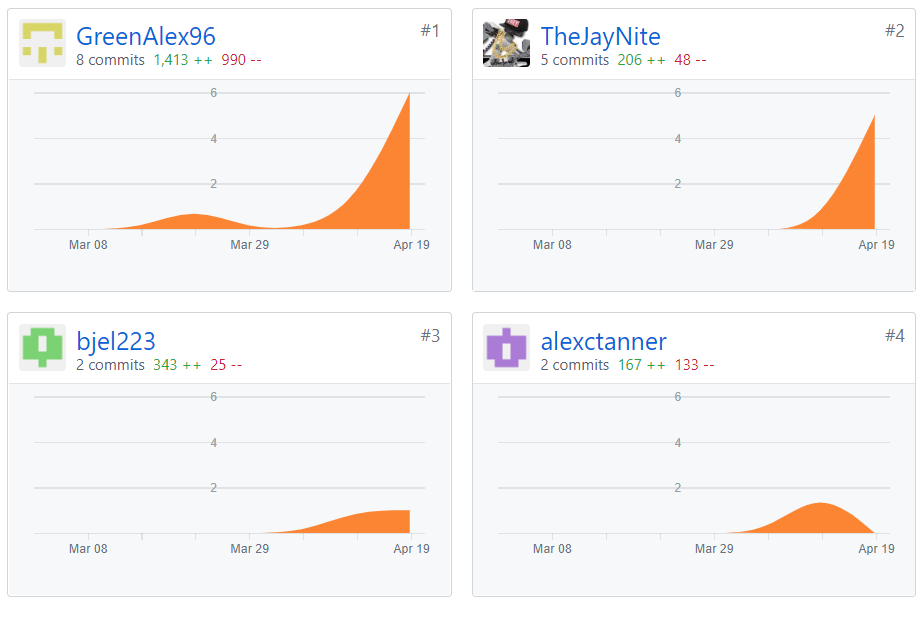
After our meeting with the customer, we had a lot of potential user stories to address.

Given the global pandemic, we had to scale back the scope of our project and focus on what we could get done based on our changed schedules and work loads.



Highlighted are the user stories we ended up completing.

Looking at our Github, we have changed 2,129 lines of code with 933 of those lines being brand new code.



(From March 1st to April 21st)

1. GitHub Developer Notebook

Our Github developer notebooks can be accessed from this page: <https://github.com/GreenAlex96/StorefrontTestingApp/wiki>

1. Demo Video
2. Word Count (This document)

Alex Reel: 625

Patrick Yoder: 2750

Benjamin Ellis: 160

Camron Savage 357