Storefront Testing App

Coding Assignment

Team 4 - Bugs are Features Too

University of Kentucky - CS 499 - Fall 2019

Customer: Ben Fox, Trissential

December 4, 2019

Authors:
Seanna Lea LoBue - Unit Testing, Storefront Administrator
Michael Murray - Introduction, User Manual, Metrics
Eric Prewitt - User Manual, Admin Manual
Nicholas Wade - Defects, User Manual, Acceptance Testing
Kee West - Acceptance Testing, Unit Testing

0. Introduction

Our project was to make an E-commerce website for Trissential, a Software quality assurance company, with the ability to toggle the inclusion of intentional bugs on a per-user basis. The motivation was to provide our customer a platform to easily demonstrate the value of their services to prospective clients in an environment they control and can easily modify. The program consists of two major sections, the mock storefront and the administrator tools for assigning and creating new bugs. The storefront is designed to mimic the core functionality of a real E-commerce website, existing to provide an environment where a large number of bugs are possible to be inserted. The administrator tools allow a storefront admin to see all of the possible bugs that currently exist as well as a mechanism for inserting additional bugs. The admin can also see a list of all registered users and can assign an individual any combination of existing bugs. In the sections below, we will provide a link to all of the source code for our project, provide instructions relevant to basic users, storefront administrators and system administrators, outline our testing plan as well as review the results of our tests ending by highlighting the core metrics for our project.

1. Implementation

1.1 Source Code

To facilitate the development of this project we used GitHub to store our code as well as to maintain versioning across the primarily disjoint development by our team members. Our source code is available at the link below.

https://github.com/nawa236/StorefrontTestingApp

1.2 Quality Review

1.2.1 Methods

- Manual inspection of committed code
  - Readability/understandability
  - Comments
  - General complexity
  - MySQL query formation review
- Human review of the webpage output
  - Can the page be opened
  - Style
  - Output formatting
  - Manual functionality testing
- Automated testing
  - Are there appropriate element tags so tests can be made?
  - Do the created tests still pass based on newly committed changes?

- Finalized Page
  - Does the page meet the acceptance criteria?
  - Does the page link with all of the expected pages?
  - Is there a bug for all major points of interaction?

1.2.2 Issues Found

- Email verification links to [localhost/storefront/verify.php…] not the final address
  - Making this change for additional deployments will need to be acknowledged in the admin manual
- Firsttimelogin.php formatting is similar but noticeably different from login.php
- Number format (2 places past the decimal point) is not properly enforced in price display in checkout.php
- Receipt page contents generation and formatting
- bugCheck.php not included in login.php
- document.getElementById("review").addAttribute("hidden"); addAttribute not an accepted function
- User could login without completing email verification as if they were logging in after email verification but before completing account information
- Long bug names - functional area exceed the admin panel display size
- When adding bugs through the webpage, need to know the id number for configuring bugCheck call.

1.3 User's Manual

1.3.1 Basic User/Testing Employee

Getting Started
By default, our webpage begins at http://40.71.228.49/StorefrontTestingApp/login.php, however; depending on deployment by a local system administrator, the target address will be different and it will be the responsibility of your system administrator to provide you the correct address. In further explanations, this IP address prefix will be used, so in the event of using a different origin address, adapt accordingly. Functionality descriptions are based on the bug-free deployment of the webpage, as such, when bugs are assigned, it is possible that some descriptions are intentionally no longer valid. However, core functionality of both registration and login are exempt from bugs as this would prevent the ability to enforce that users must be logged in to use the system or that a bug could be assigned to a specific logged in user. Some additional functional explanation is provided in the following sections in order to make clear the intended behavior to help differentiate between what may be an intentional bug assigned to you and what may have been an oversight on our behalf.

Registration
In order to access the contents of the website you are required to be logged in, as such, if you do not currently possess an account of your own or are not provided with login

credentials you will need to navigate to http://40.71.228.49/StorefrontTestingApp/register.php to create an account. You will be required to enter a valid email address and submit your desired password. Once filled out, you will click “Sign me up!”. This will send a verification email to the email address you provided containing a link that will complete the registration process. After completing this step, you will be redirected to a page where you will be prompted to provide basic personal information such as name and address. Upon completion would will be taken to the storefront homepage.

Login
Assuming that you already have an account that has been completed in line with the previous registration steps, you can navigate to http://40.71.228.49/StorefrontTestingApp/login.php and type in your email and password into the appropriate fields. Upon successful login you will be taken to the storefront homepage.

Storefront Homepage
The main purpose of the homepage is to show a list of all of the products along with a number of filtering options to help you to find the target item. The page contains three major filters, first is the search box present at the top of the page in the header that is present on all pages after login. Search input can be submitted either by pressing the enter key while the search box is the active element or by clicking the search icon in the search box. Search will show only products whose names are similar to your search query. Price filtering is handled by two slider bars, min price and max price, located in the filters section of the sidebar. Both slider bars have an attached input text box that both displays the value associated with the slider bar position as well as allows you to input to specify exact values. Upon releasing the mouse on the slider bar, pressing enter in the input box or incrementing the value in the input box, the selection of products will be filtered accordingly. Finally is a category selection filter, also located in the sidebar, which gives you the ability to display only the category of products which you would like to view. All of the filters can be used together. The final selection options is a sort by dropdown box which controls how the products displayed are sorted. The products shown each contain an image, name, price, description and a configure button which will redirect you to the product page which contains all variations of that item that are present in the product database.

Page Header
Each page after login has a header bar that spans the top of the page. This bar contains a home button which will take you back to the product list page, a search bar which will take you to the product list page and return the result of the search filter, an account button, an orders button and a cart button. The account button expands when moused over revealing a logout button and an account page button which will redirect you to the account page. The orders button will take you to a page that will allow you to look at past completed orders and the cart button will take you to your cart page.

Product Page
The product page displays the product image, name, description, price, and stock as well as the selection buttons for color, size and sku of the currently selected item. In the event that there are multiple color, size or SKU variants for the same item, additional selection options will be shown for color, size or SKU. Clicking these buttons will redirect you the product page associated with the selection. Finally, there is an add to cart button along with a quantity selection input box. Clicking on the add to cart button will that however many items to the cart in line with the value in the quantity select box. Stock quantity is considered when adding to the cart so in the event that the number of items added to the cart combined with the number of that item currently in the cart exceeds the stock, the add to cart request will be denied. Basic feedback is provided when clicking the add to cart button but to view the cart contents, you must use the cart button from the header to view details.

Cart
The cart page details all of the products that are currently in the cart. All products have a quantity selection option in the event that you wish to alter the number of an item in your cart, however once changes are made, you must click the “Save” button or the “Checkout” button located at the bottom right of the cart contents listing. To remove an item completely from the cart, changing this quantity to zero and clicking save will remove the item from the cart. Below the list of cart contents is the total price of the cart including tax. Once you are satisfied with the contents of the cart and wish to checkout, immediately beside the save button is a checkout button which will redirect you to the checkout page where order information is finalized.

Checkout
The checkout page contains input forms for billing information as well as shipping speed selection. Changing the shipping speed from the standard rate will incur an additional charge and the total order price is displayed. Once all information has been provided, clicking the checkout button will place the order and take you to a receipt page which highlights all of the details of the order.

Account Page
This page can be gotten to by way of the header, mousing over the Hello, <Your Name> button will reveal a button called view account. This page contains a listing of all of the account information you have provided. In the event that any of the information needs to be changed, by simply changing the field of interest and clicking the update button at the bottom, the newly input data will be updated in the system.

Past Orders
Also linked in the header, this page can be accessed by clicking the order button in the header. This page will display a minimalistic display of all of your orders, each with a button to view additional information related to this order. Clicking on this button will display a full list of the order contents as well as the price of the order.

General
All interactive elements such as buttons, input fields and slider bars possess an element id unique to the page so that automated testing can be done to interact with specific and consistent page elements. While assigned bugs will vary, the most common type of bug will be that interactive page elements do not respond as anticipated or described in the sections above.

1.3.2 Storefront Administrator

Bug Assignment
The bug assignment page displays all of the registered users and all of the bugs in the database. By clicking into the user list, the list of users will be displayed as "Last Name, First Name". You can type into the user list, and it will automatically display matching entries as you type. The list of bugs is organized alphabetically by the functional area, or page and feature, that the bug affects. You can multiselect by holding the control button while making your selections. Bug assignments can have start and end dates. The end date is null, but the start date will default to the day you are assigning the bug.

When you click Review Bug Assignment, the user and the bugs selected during this session will be displayed. Clicking the Save Assignment button will add these bug assignments to the database. If the bug assignment already exists for that user, you will receive an alert as it continues to save any remaining bugs.

Updating Bug Assignment
The update bug assignment page displays a table of bug assignments when a user is selected. The table will allow you to change the start and end dates for an assignment and then either update the assignment or delete the assignment from the database.

Bug Review, Editing and Creation
This page allows you to view all of the existing bugs along with their details and relevant code block as well as allowing the insertion of new bug definitions. By clicking on the field on the left side of the screen that says "Enter a bug name" you will see all of the bugs that are in the system. This will populate the fields on the right side of the screen with the details of name, functional area, descriptions and code block. In the event that any of this information needs to be edited, simply changing the contents of these fields and clicking the "Update Bug" button will update the information saved. In the event you wish to add a new bug, typing in values for bug name, functional area, description and code block with no bug currently selected and pressing the "Save New Bug" button will save this bug into the system. However, due to the nature of hooking the bug into the code, the insertion of a bug will still require making a few simple additions to the core php files in order to support the bug. This process is described in detail in the System Administrator's Manual. Therefore, **if you are not also a system administrator or are not working directly with a system administrator during the process of adding a new bug, you should not use the "Save New Bug" feature.**

1.4 System Administrator's Manual

Getting Started

Our project was developed for use on a LAMP stack (Linux, Apache, MySQL, PHP). Assuming that this requirement has been met, simply navigate to the directory that you wish to use and from the terminal type “sudo git clone https://github.com/nawa236/StorefrontTestingApp.git”. This will download all of the required files for execution and you may continue to the next step.

Database Initialization
This project relies on the existence of a MySQL database accessible by the server where the code is executing. In order to function, the database connection attribute will need to be updated as appropriate for your environment. Both in the file “dbConnect.php” and “database.php” are variable definitions for servername, username, password and database name. These variables will need to be manually changed to match the appropriate login credential for your MySQL. Once the values have been altered, navigating to http://40.71.228.49/StorefrontTestingApp/index.php substituting the initial IP address prefix for the location of your server and directory path. Assuming that the message “Welcome to the Employee Store.” is displayed, the database settings have been properly provided and the initialization of the database for use is complete.

Account Verification Email host
The PHP server the website is hosted on must have a mail sending or relay service that is properly configured for the website verification mailing options to function correctly. This will be highly variable depending on the current server configuration and desired functionality, so it will be left to the administrator to determine the best path to take regarding this service.

In addition, the mailing system was tested by the team using a relay to the Gmail SMTP servers with the account trissentialbugsite@gmail.com. There are references to this email (as the from entry for the email) in register.php, forgotpassword.php and reverify.php. These entries will need to be changed depending on the intended email to be used for this purpose.

Additionally, the email link is currently setup to use our webserver but will need to be changed to the server IP of the PHP server the website in installed on.

If it is desired to use the Gmail account in the future, the details are below:

Account: trissentialbugsite@gmail.com

PW: BugsAreFeaturesToo

Maintenance - Adding New Bugs
The primary functionality additions for this project will be the ability to add new bugs into the system. As mentioned in the Storefront Administrator section of the user manual, adding a new bug is composed of two major tasks. The first should be to determine what you want the bug to do and to navigate to the appropriate php file that contains the page

element that you wish to add a bug to. Once on the page, select the line or lines of code that you wish to add a bug for and encapsulate it similar to this example. *Text in "< >" used to describe the contents.*

```
//***** Bug Start *****//
$bugCode = bug_check(<bug id>);
if(is_null($bugCode))
        <Original Line/Lines of code>
else
        eval($bugCode);
//***** Bug End *****//
```

The example above is for adding bugs to a php section of code and should not produce an error as long as the php includes either header.php or bugCheck.php. If neither file is included, you will need to include bugCheck.php. While similar, because bugCheck function is a php function, in the event you wish to add a bug to a JavaScript section of code, an example would look like:

In many cases you may not need a branch for both bug or no bug, as you may want to

```
//*****  Bug Start ****//
var bugCode = "<?php echo bug_check(<bug id>);?>";
if(bugCode == "")
        <Original Line/Lines of code>
Else
        eval(bugCode);
//*****   Bug End  ****//
```

add a line of code if there is a bug, or only run the correct line in the event of no bug, without explicitly having additional bugged code to run. All of these methods are acceptable, so consider what is needed when actually inserting the bug branch handling code.

The other task is to add this bug into the database. This can be accomplished through the web interface by using the admin.php page; however, as this is just a GUI for interacting with the database, it is also possible to manually insert new rows into the bug table in the database. On the admin.php page clicking on the "Bug Creation and Editing" tab will allow you to enter in a new bug name, functional area, description and code block. Clicking the "Save New Bug" button will add this bug definition to the database. Once saved in the database, it is now possible to go back to the "Bug Assignment" tab and select users to assign this bug to.

Maintenance - Adding New Web Page Features
All existing web pages are designed as single php files that when needed, call auxiliary php files through ajax or are "connected" through the use of page navigation hyperlinks.

Due to the lack of inherent coupling, editing or adding features to these pages should be easy. While exact method is up to you, simply adding the additional elements into the HTML section of the php file, ensuring that the output is as expected, then adding whatever dynamic page interaction is needed is likely the best course of action. In the event of making new pages, unless there is a need for your new pages to directly interact with a previous page, such as through get/post, it is recommended to update or query from the shared database for most inter-page interactions.

2. Testing

2.1 Test Plan

Our tests primarily exist to demonstrate the core functionality of the individual pages as well as ensure their interactions together function as expected, that our bugs work as expected and that the admin can properly control the assignment of bugs. Where functioning as expected is primarily defined by the expectations of the customer, which is surveyed in the acceptance testing. These tests can be automated through Selenium thanks to robust use of page element tagging however many were tested manually by each developer upon developing and committing code.

2.2 Acceptance Testing

Acceptance testing is used to test software for a desired acceptability and display core functionality requirements. These tests are used to determine the state of compliance between interacting implementations and pages towards the finalization of the deliverable product. Our tests were focused around ensuring that our project meets the functional requirements given to us by the client as well as prove separately developed components interact in the expected manner. The figure below is a list of the tests we performed to ensure that prime aspects of the website function completely and were integrated successfully. These tests can be automated through Selenium test scripts, however most functional components of our project were manually tested with an assigned Pass/Fail value given based on the output and its similarity to the desired results.

ID	Objective	Description	Expected Result	Result
1	Can create an account	User is able to use register.php to create a new user account.	User gets confirmation of account creation message and asked to verify email.	Pass
2	Registration requires email verification	After entering email and password in registration page an email sent to the user’s email will allow a user to verify their email and finish	An email is sent to the users email and login cannot happen until email verification is complete.	Pass

		registration.		
3	Verified user can log in	A user should only be able to access webstore after email verification is complete.	Users that have not completed email verification cannot access webstore but users that have completed email verification can access the webstore.	Pass
4	Can view products	User should be able to select and view individual items in the store.	When the user selects an object in ProductList.php, they will be redirected to an Item page for that item.	Pass
5	Can add items to a cart	On an item’s page the user can indicate a quantity of items and sku	Items added by a user to the cart will appear in cart.php in the correct quantity.	Pass
6	Can purchase the cart	Hitting checkout in the cart page will take the user through checkout and at the end submit the order.	At the end of checkout, the cart will be empty and the order will appear on the users order page.	Pass
7	Can view account information	The users account info should be available on the account page.	Hitting the account button in the header should take the user to their account page.	Pass
8	Can review past orders	The user should be able to view previously submitted orders.	Hitting the orders button in the header will bring the user to a page displaying their previous order.	Pass
9	Can easily perform Selenium-based testing	All pages should be setup with ids embedded in their tags for Selenium.	Selenium can input into fields properly and read data from page	Pass
10	Admin can assign bugs to specific users	Admins should be able to assign a bug to appear for a specific user.	Bug appears when assigned user loads the relevant page but not for any other user unless they were also assigned that bug.	Pass
11	Admin can add new bugs into the system	Admins should be able to create bugs for the webstore to be assigned to users using the bug interface.	New bugs are added to the database and can be assigned and edited by an admin at a later date.	Pass

12	Admin can update an existing bug	Admins should be able to edit bugs in the database using the bug editor	Bugs edited by admin save to database and apply in their new form to assigned users.	Pass
13	Can edit cart quantities	Editing a quantity field in the cart and hitting save changes the quantity in the cart.	If the user changes a quantity in the cart and hits save the cart will reflect that upon the automatic reload of the page.	Pass
14	Can remove item from cart	Setting a quantity in the cart to 0 and saving removes the item.	If the user changes a quantity in the cart to 0 and hits save the cart will no longer have that item upon the automatic reload of the page.	Pass

2.3 Unit Testing

Unit tests were implemented throughout the course of the project by each individual developer on the pages they were working on. A unit test is used to help ensure that individual components of the project work before integration into the software and validate that each component functions according to its design. As our project is a website-based application, we were able to test functionality immediately when code was pushed on the server throughout the course of development. The figure below lists test cases that were performed as well as a description of their intended functions with an assigned Pass/Fail value given based on the output and its similarity to the expected results.

2.3.1 Core Website Unit Testing

ID	Objective	Description	Expected Result	Result
1	Ensure new account can be made.	From register.php, input new account information	Success message appears and requested to check email for verification	Pass
2	Ensure new account verification is required	Login in with correct credentials before completing email verification	Error message displayed about being unverified	Pass
3	Ensure verification activates account	Click the link from the email provided when registering the account	Account activated and user is taken to the login page	Pass
4	Ensure previously created account is accessible.	Login with previously created account credentials.	Login successful and user name shown in the top bar	Pass

5	Ensure user info can be updated.	Click to edit a field in account information.	Submitted change persists after a page reload.	Pass
6	Ensure product filters function as intended.	Click a product filter on a known query. (Repeat for all single filters)	Only specific product should remain.	Pass
7	Ensure product search function as intended.	Search for an item. (Repeat for a couple items)	The item card should be displayed.	Pass
8	Ensure product configurations function as intended.	Change configuration (size, quantity etc.) on a specific product page.	Price displayed on add to cart should match expected value.	Pass
9	Ensure cart status is updated on adding item to cart.	Click to add an item to the cart.	Cart status indicator is incremented by the number of items added.	Pass
10	Ensure items can be removed from the cart.	Click to remove an item from the cart.	Item is no longer displayed in cart and the cart total is correct.	Pass
11	Ensure checkout page validates input.	Click to submit an order for a populated shopping cart without putting in payment.	Order not accepted. User requested to fill missing information.	**Fail**
12	Ensure all interactable elements have tags.	For each interactable element on the page, inspect for tag.	Each element will have a unique tag. (Manual test)	Pass

2.3.2 Bug Framework Unit Testing

ID	Objective	Description	Expected Result	Result
1	Ensure setting a bug works as intended.	A bug is set to active. (Repeat for all defined bugs)	The bug has the intended impact.	Pass
2	Ensure Admin can set a bug for a user.	Admin sets a bug for user. That user logs into their account.	User’s experience is impacted in line with bug’s intended impact.	Pass
3	Ensure set bug does not impact other users.	Admin sets bug for User1. User2 logs into account.	User2 does not experience any bugs.	Pass
4	Ensure that new bugs can be added.	A bug is added to the database and the bug is set to active.	The new bug has the intended impact.	Pass
5	Ensure admin can see a list of all the bugs	On bug assignment and editing screens, all bugs are displayed.	The list of bugs on these pages both match each other as well as the list	Pass

			of bugs on the MySQL server.	
6	Ensure admin can view all bug assignments	On update bug assignment, selecting a user displays all assigned bugs.	The list of bugs displayed and their data matches the bugs in the MySQL server.	Pass
7	Ensure admin can remove a bug from a user	On the update bug assignment, an assignment can either be expired or deleted.	Changing the end date or deleting the bug properly updates the MySQL server.	Pass
8	Ensure unsetting a bug actually restores user functionality	Changing the expiration date or deleting the bug turns off the bug code.	User functionality is restored for each tested bug.	Pass
9	Ensure that selecting a bug displays its data in the editing tool	The bug data updates upon change.	The bug data is retrieved and the DOM is updated with the correct data.	Pass

2.4 Testing Quality Review

2.4.1 Issues Found

- Original admin tests did not completely survey all features of this page.
  - Additional tests were added.
- Some tests present since architecture assignment were no longer valid for the current state of the project.
  - Tests were removed.
- Some cart functionality was not being tested
  - Additional tests were added.

3. Metrics

3.1 Estimated Story Points

Based on the combined estimates of all of our team members, we believe that our project is 90 story points, of which we completed 84. Story points were assigned relative to scoring the product list page as 8 points. The list of user stories is available at the following link where unless explicitly stated, was implemented.

https://github.com/nawa236/StorefrontTestingApp/wiki/User-Stories

3.2 Actual Lines of Code

We have approximately 2,700 lines of code for our project. This estimate does not include empty lines or comments.

3.3 Module Complexity

Average Complexity per LLOC:	0.21
Average Complexity per Class:	14.50
Minimum Class Complexity:	1.00
Maximum Class Complexity:	28.00
Average Complexity per Method:	2.50
Minimum Method Complexity:	1.00
Maximum Method Complexity:	14.00

- None of our modules are explicitly coupled.

3.4 System Complexity

- None of our classes are explicitly coupled.
- Each of our pages are functionally independent from one another. All direct interactions come from arguments submitted via get/post submissions on page redirection or by accessing a shared database. Changes made to the database are seen on other pages by way of database query results independently generated on a per-page basis.

3.5 Product Size

- 84 user story points implemented
- 55 test cases
  - 9 Acceptance Tests
  - 31 Unit Tests
  - 12 Integration Tests
  - 3 System Tests
- 30 code containing files produced
  - 4 database building resource files
  - 120 image files
- 45 discrete methods inside of files
- 22 files that are single method

3.6 Product effort

- Team/Customer Meetings - 35 Hours (7 Hours each)
- User Story Preparation - 4 Hours
- Project Plan Preparation - 22 Hours
- GitHub wiki/Develop Notebook - 8 Hours
- Architecture Assignment - 8 Hours
- Code Creation - 150 Hours
- Coding Documentation Assignment - 6 Hours
- **Total - 233 Hours**

3.7 Defects

- Shipping speed attribute does not exist in the database, as such, there is no way to view that information in past orders
- Currently there maybe sever instabilities within the server setup causing issues with non-SSH connections
- Currently there is an issue with the SQL database setup causing occasional issues were the root password is not accepted, this can be fixed by a server restart.
- Input to order such as shipping address is allowed to be blank
- Updating cart does not ensure quantity is valid within stock limitations
- Placing an order does not deduct from product stock

4. GitHub Wiki/Develop Notebook

https://github.com/nawa236/StorefrontTestingApp/wiki

5. Demo Video

https://www.youtube.com/watch?v=zCwRGG9yVG4

6. Word Count (This document)

Seanna Lea LoBue - 384

Michael Murray - 1652

Eric Prewitt - 573

Nicholas Wade - 2093

Kee West - 660