# UNIVERSITY DEPOT

Logo, company name

Description automatically generated

**Testing Manual**

**Slippery Rock University**

Stephen Simpson - sjs1035@sru.edu

Cody Kuntz - cak1033@sru.edu

Heidi Bednarz - heb1010@sru.edu

Dustin Annarumo-dja1011@sru.edu

Zack Colich - zjc1006@sru.edu

Jenna Josselyn jmj1022@sru.edu

# Table of Contents

# 

# Manual Testing ........................ 3

# Automated Unit/Integration Testing ............................... 3

# Purchase Pages Testing .............................................. 3

# General Pages Testing ........................5

# Model Controller Testing ............................... 5

# Unimplemented Testing Areas .............................................. 6

Manual Testing

A full run through of the project was completed which entailed going through the process of logging in, viewing, selling, and buying listings, messaging and more. Many features were manually tested as they were being developed such as the search functionality. Each time changes were made; the feature was manually tested for common errors. Filtering through search was also tested for errors as it was being developed to ensure the results would be accurate.

The full run through revealed a few issues and bugs that needed to be fixed. It also showed that the website wasn’t seamless throughout. There was issues with sizing, some pages having back buttons while others don’t, different coloring schemes and more. Once the full run through was completed, these were corrected so that the pages would display in the correct sizes. Images were also resized as they were very large when doing the run through.

Automated Unit/Integration Testing

The project uses the jUnit 5 testing library for running automated tests. These tests can be accessed by opening the project in Eclipse and then opening the classes in the src/test/java project folder. Each testing class is under the package name corresponding to the package name that the actual java code is under. Each class holds a set of tests for a different component, and they are run individually.

This section will discuss some of these classes in turn. This will include information about the functionality being tested, and the purpose of each testing case included in the testing class.

Purchase Pages Testing

ConfirmPurchasePage Testing

The package for handling testing of the ConfirmPurchasePage is edu.sru.cpsc.webshopping.controller.purchase. The ConfirmPurchasePage is used to enter payment information and finalize purchases of a quantity of an item.

The first set of tests is used to ensure that entering valid information leads to a successful purchase, with its associated database updates.

* CCPurchaseSuccess() tests that a purchase with a credit card is successful if the user’s credit card information is added, and if their provided information is valid. This also tests that the database is successfully updated.
* PaypalPurchaseSuccess() tests that a Paypal purchase is successful if valid information is provided, and it tests that the associated database updates are made. A user does not have saved Paypal information, so it is not checked against any existing records.

There are also several tests for checking validation and purchase failure:

* CCPurchaseFailure() tests all cases of credit card validation failure, and ensures that the database is not updated.
* PaypalPurchaseFailure() performs the same test, but for Paypal data.
* CancelPurchase() checks that canceling a purchase returns to the original MarketListing they attempted to purchase.

MarketListingPage Testing

The package for handling testing of the MarketListingPage is

edu.sru.cpsc.webshopping.controller.purchase. The MarketListingPage is used to show information on a specific listing that a user can purchase from, and to allow them to begin the process of purchasing a quantity of items.

The first set of tests verifies behavior for when the web page is initially loaded. These include:

* The LoadPageSuccess() test, which ensures that the page is loading successfully.
* LoadPageListingDeleted() test, which ensures that, in the event that the user attempts to access a deleted webpage, an error is thrown.
* LoadPageNotLoggedIn() tests that an exception is thrown when the user attempts to access the page while not logged in.

The next set of tests are used to verify that users can successfully begin purchasing, and to check the validation functions for user input.

* The AttemptPurchaseSuccess() page tests that, if the user attempts to purchase a correct number of items, that they are then moved to the confirm shipping address page.
* AttemptPurchaseFailureTooManyBought() tests that the user is unable to proceed if they attempt to purchase too many items.
* AttemptPurchaseFailureFormErrors() tests that the user is unable to proceed if their input has form errors.

TransactionDetailsPageController Testing

The package for handling testing of the TransactionDetailsPageController is edu.sru.cpsc.webshopping.controller. This is a form of whitebox testing, where mock users and purchases are made, and the effects on the state of the database are compared to what is expected as a result of various actions.

The major tests are:

* submitShippingUpdateSuccess(), which tests that a seller can successfully update the shipping information, whenever the provided information is valid.
* submitShippingUpdateFailure(), which tests that a shipping update fails whenever invalid information is provided by the seller for the update.
* deleteTransactionSuccessful(), which tests that the transaction is successfully deleted, whenever the user attempts to delete the item before it has shipping information attached
* deleteTransactionFailure(), which tests that a transaction is preserved, whenever the user attempts to delete the item after shipping information is attached

General Pages Testing

UserDetailsPage Testing

The UserDetailsPage is used by the user in order to update their personal and payment information. This set of tests is primarily based around testing that updates are successfully made to user information, and that the required validation works correctly.

The first test is for testing successfully page loading, which is beneficial for ensuring that the page has not been accidentally broken.

* This test is initializeTest(), which just checks that no errors have been thrown

The next set of tests verify the behavior of the user attempting to update their payment and direct deposit information.

* updatePaymentDetailsSuccess() and updateDirectDepositDetailsSuccess() both test that if the user attempts to update with valid information, that the associated data is either updated or created
* updatePaymentDetailsFailure() and updateDirectDepositDetailsFailure() both check that the database will not have data added if invalid information is submitted. It also checks all of the validations individually in order to ensure that they are functioning.

Model Controller Testing

MarketListingDomainController Tests

The MarketListingDomainController handles database interactions with the MarketListing class, which is used to hold information on posted listings. This set of testing cases is used to check the validity of MarketListingDomainController functions that interact with the database.

There are several tests for this functionality:

* PurchaseQuantityValid(), which tests that updates to the MarketListing as a result of the purchase are successfully applied when the purchase information is valid.
* PurchaseQuantityInvalid(), which tests that updates to the MarketListing as a result of the purchase are *not* applied whenever the purchase information is invalid.
* PurchaseQuantityDeleted(), which tests that the function throws an exception whenever an attempt is made to update a deleted listing.

Unimplemented Testing Areas

There are several testing classes within the different packages that have not been implemented, but still exist as classes. Each of these classes exists only as a stub, and should be implemented during future development. These classes include:

* RepositoryBillingSpringTest
* RepositoryUserSpringTest
* WebshoppingControllerAllTests