Online Auctioning

Testing Manual

**Slippery Rock University**

Spring 2024

Jacob Johnston – jrj1014@sru.edu

Tiffany Morrow – tbm1005@sru.edu

Nicholas Vioral – ncv1005@sru.edu

Wolfgang Zoppelt – wsz1001@sru.edu

Objectives and Scope: The purpose of this testing plan is to guarantee the functionality, performance, and security of our Spring Boot application. It encompasses unit testing of separate components, integration testing of interconnected components, and user acceptance testing to confirm the overall performance and usability of the application.

Testing Strategy: Our strategy adopts a phased methodology. We commence with unit testing utilizing JUnit and Mockito, proceed to integration testing to assess component interactions, and culminate with user acceptance testing for comprehensive evaluation of the application's end-to-end functionality.

**System Requirements:**

Specifications:

Windows OS

MySQL Workbench 8.x

JDK: 17

Tomcat 10.0.x

Spring Boot 3.1.x

**Unit Testing:**

Setting Up the Test Environment: We configured the unit testing environment using the Eclipse JUnit plugin and employed Mockito to generate mock objects, facilitating the isolation of individual component testing.

Test Cases and Coverage: Our unit testing strategy encompassed the creation of test cases for essential functionalities, such as setters and getters for domain entities. These tests spanned both positive and negative scenarios to achieve thorough coverage, extending to domain classes, form classes, and DTO objects. We also performed unit testing on the majority of units in the service package to ensure that methods of the services worked as intended.

**Integration Testing:**

Integration Approach: Our integration testing methodology emphasized examining the interaction between controllers, services, and entities, aiming to replicate real-world usage scenarios.

Test Scenarios: Key test scenarios comprised listing items for sale, logging in, and purchasing items, with a primary objective of ensuring robust and error-free component interactions.

**Conclusion:**

Upon receiving the project, it initially exhibited 3 failures and 41 errors among 272 tests. We diligently enhanced, updated, and refined numerous tests, resulting in a reduction of failures to 2 and errors to 38 out of 380 tests. Throughout the process, we encountered various complications and obstacles in testing. However, the remaining 340 tests function as intended, including the newly added discussion board tests. Although we were not able to test everything, we did manage to complete the testing of domains, and services. But due to the complexity of testing controllers, we were not able to completely test them and that is where many failures and errors exist. These errors and failures do not necessarily mean that the code is incorrect but that we were unable to test them correctly.