OpenBazaar - Test Report

The Fair Traders
Daniel Mandel - mandeldr - 1303865
Shandelle Murray - 1303109 - murras25
Connor Sheehan - 1330964 - sheehacg

Friday, November 27th, 2015

Abstract

This report contains the test results for the OpenBazaar redevelopment project, for McMaster University SFWR 3XA3.

Contents

Introduction

1 Objective

Testing the OpenBazaar application has several main goals and will span several testing domains. At the lowest level, unit testing must be completed to ensure accuracy of the application's dataflow. It is important to ensure that testing includes both white box and black box tests. White box tests will ensure that a module's secrets have been implemented adequately, and black box tests will ensure that the modules as a whole operate correctly. Further, usability testing will demonstrate the application's functionality from the perspective of use cases. Finally, security testing is essential for the application as confidential and sensitive data can be stored and transmitted via the application.

2 Approach

Our approach involves the use of the unittest2 library that python has available. Test cases will be written using the same design as any NUnit or JUnit implementation (Arrange Act Assert). Meaning all variables at first will be created, manipulated and then verified. This is the most effective way of organizing unit tests and as well as extremely modular. All test cases will be ran from the command line by simply running the test fixture.

3 Definitions

Black Box Testing	The process of testing a function or module without taking the internal implementation into account.
White Box Testing	The process of testing which takes the internal implementation into account, and ensures it's correctness.
Usability Testing	The process of evaluating a program by testing it on users. Provides direct input on how users will interact with the system.
Ricardian Contract	A contractual agreement for a trade which aims to be exact and unchangeable. Any contract is comprised of several modules which provide information about the interaction, as well as a hash of the data to ensure that the contract is not changed retroactively. Ricardian Contract's are human readable, as well as machine parsable.
Kademlia DHT	A distributed hash table for decentralized peer-to-peer networks. This is the infrastructure used in the OpenBazaar application.

Test Results

4 Black Box Testing

5 White Box Testing

This section contains all unit test results that do not cover usability, performance or security. This is because the internal design of the software is known to the tester.

Class: Ricardian Contract Tests

- Test Case Name: `test_constructor`
- Initial State: Nothing initialized
- Input: RicardianContract(`contract_dict`, `seller_settings`)
- Expected Results: RicardianContract Object Instance
- Results Match: Yes

- Test Case Name: `test_get_module`
- Initial State: testRicardianContract, `seller_settings`, `contract_dict` objects initialized
- Input:
  - tradeDict = testRicardianContract.`get_module('trade')`
  - price = tradeDict['price']
  - `item_name` = tradeDict['name']
  - keywords = tradeDict['keywords']
- Expected Results:
  - keywords == 'Head'
  - `item_name` == 'Hat'
  - price == "50"
- Results Match: Yes

6 Usability Testing

7 Performance and Usability Tests

8 Security Testing

Test Summary

9 Unit Test Summary

10 Performance Test Summary

11 Security Test Summary

12 Changes Derived from Tests

After running through all of our tests some changes were necessary. The following is a list of documented changes to our code after testing:

- Class: RicardianContract
- Test: test contract hash
- Output: Hash Object
- Expected: String

After dicovering that the contract hash was returning the wrong object, we went back and reimplemented the contract hash function for the RicardianContract class.