SOFTWARE TESTING DOCUMENT

Grape’s Financial Market Simulator

Abstract

This document contains information regarding the different testing mechanisms deployed during the development of Grape’s Financial Market Simulator

Grape

cos301.mainProject.grape@gmail.com

Contents

Testing tools

The Financial Market Simulator was developed purely in Java and the testing tools used were testing tools that support the Java programming language.

Unit Testing

A unit testing framework for the Java programming language. An integral part of test-driven development in Java.

Integration Testing

Integration Testing was designed and deployed by the Grape team.

Testing Built

Maven is an automated built tool created mainly for Java projects. The Maven tools also has functionality to run all unit tests before the application is built and helps to create more fault tolerant software.

Unit Testing

Test Execution

Using Maven, all the JUnit unit tests run before the project is built.

The Maven Lifecycle has the following build phases:

- Validate - validate the project is correct and all necessary information is available
- Compile - compile the source code of the project
- Test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- Package - take the compiled code and package it in its distributable format, such as a JAR.
- Verify - run any checks to verify the package is valid and meets quality criteria
- Install - install the package into the local repository, for use as a dependency in other projects locally
- Deploy - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

JUnit automatically verifies the correctness of a program's behavior. The main advantage of JUnit is that it is automated rather than you manually having to check with your print outs. Each test you write stays with your system.

Test Development

The development process followed was a test-driven development strategy. Every test was written and then run to see if it fails. The function for that test was then written and minimalistic code was written to see if the function correctly behaves like it is suppose too. The code was then refactored according to the application's requirements.

Test Coverage

All major functions were tested. All minor trivial functions were not tested.

Integration Testing

Test Execution

A special class called FinancialMarketSimulator is run to test the different components of the system integrating. At several points in the testing output of the internal system is being displayed and expected results are compared to the actual output of the system.

Test Development

All components in the application have different states during the execution of the program. The components were developed in such a fashion in order to compare states at different points in the system and compare them with expected results and these comparisons will be regarded as the testing the integrability of one component with another.

Test Coverage
All components in the system as per requirements specification were covered.

Non-functional Testing

Performance

Test Execution
A testing class was created and run to see the amount of trades occurring in minute.

Test Development
The function was created and used to test how many trades can occur within a minute. Approximately 12000 trades occurred within a minute.

Test Coverage
Only performance of trades occurring within the application was recorded.

Scalability

Test Execution
A testing class was created and run to see the amount MarketParticipants that can trade within the market.

A testing class was created and run to see the amount StockManagers that can be implemented for MarketParticipants to trade within the market.

Test Development
The test classes were created to see how many StockManagers and MarketParticipants can concurrently be run in the application. The application works best with 30 or less MarketParticipants and 3 or less StockManagers. The application tends to slow down dramatically if more than 30 MarketParticipants are trading.

Test Coverage
Only scalability tests only covered for MarketParticipants and StockManagers.

Platform

Test Execution
Application is simply run on different operating systems.

Test Development
Created an application that is compatible on most operating systems. It can run on Linux, Windows, Unix and any other platform that has Java installed.

Test Coverage
Application works only on platforms that have Java installed.

Usability Testing

Test Execution
Conducted a usability test with 20 different users. A questionnaire was also handed out after the test.

Test Development
The major functions of the application were identified and then those functions were conducted into a usability test where the user would receive instructions on how to do that particular function and then doing them. Their results were then recorded for analysis on where to improve the application.

Test Coverage
The Matching Engine as well as the Simulator were both tested for most popular functions. 95% of the application's functionality was tested.