**Twitter Fingers**

Testing Specification

Julian Mathis (PM)

Dorian Wood

Anisha Carter

Taylor Clark

Dominique Collins

Sydney Parker



**Contents**

[1.0 Introduction 3](#_Toc479278356)

[1.1 Goals and Objectives 3](#_Toc479278357)

[1.2 Constraints 4](#_Toc479278358)

[2.0 Testing Methods 4](#_Toc479278359)

[2.1 Unit Test 4](#_Toc479278360)

[2.2 Integration Test 5](#_Toc479278361)

[2.3 Regression Test 5](#_Toc479278362)

[3.0 Testing Strategy 5](#_Toc479278363)

[3.1 Test Items 5](#_Toc479278364)

[3.2 Test Descriptions 6](#_Toc479278365)

[3.3 Test Checklist 8](#_Toc479278366)

[3.4 Desired Test Results 8](#_Toc479278367)

[4.0 Testing Tools and Environment 9](#_Toc479278368)

[5.0 Response to Failure 10](#_Toc479278369)

[6.0 Version History 11](#_Toc479278370)

# **Introduction**

Twitter Fingers was tasked with creating a program/application that will display tweets from pre-approved Twitter accounts that contain financial information, and store these tweets/information to a database so that they can be referenced at any moment of using the program. Our interface will display the information from these tweets in a scrolling fashion, similar to a stock market ticker. This program will be make the user more aware of the latest and most relevant financial news happening in real time. This document will give a precise description of how Twitter Fingers will test our program so that it functions in accordance to features and requirements the client has requested.

## *1.1 Goals and Objectives*

The purpose of this testing specification document is to list all components of the product that need to be tested, give detailed descriptions of how each component will be tested, and specify how test results may affect our product. A primary objective is to have this document provide clear and detailed descriptions of testing methods, so that the client can understand how the product was developed and tested before delivery. Another important objective is that this document can reflect the changes that will constantly be undergoing during the development of this product. It is possible that some testing mechanisms may be added, or removed as more features are added to the product.

## *1.2 Constraints*

Possible constraints with the testing of our product are:

* **Time:** Time can be a constraint seeing as how some features may need to be tested more frequently than others in order to ensure that it works properly. This is a major concern, seeing as how we will be working with the Twitter Streaming API, which only allows a limited number of connections in a short period of time.
* **Frequency:** Because our application pulls tweets from Twitter in real time, the testing processes may depend on how frequent the user we follow are spitting out information.

# **2.0 Testing Methods**

## *2.1 Unit Test*

Unit testing involves isolating the components of the software, and testing each piece so that it works correctly on its own. Unit tests are run by themselves before the components of the software are put back together and tested as one product. Seeing as the product will be made up of several different programs, unit testing will be important in making sure every single component of the product is working independently. We will look to use PHPUnit for a lot of out unit testing, since much of our product will be written in PHP. PHPUnit is a programmer-oriented testing framework for PHP. It is an open source framework and acts alone as a good tool for testing classes or a set of functions. Its intention is to ease the development cycle and help to avoid endless debug sessions.

## *2.2 Integration Test*

Integration testing involves combining two or more components that have already been tested (via unit testing) and testing the interface between them. The goal of integration testing is to continuously combine the different parts of the product and test them until you end up with one model. Integration testing points out problems that occur when different components of the software are put together. A test plan that calls for the testing of every component ensures how viable the unit is before combining with other units. Errors discovered when combining units are likely related to the interface between units. This method reduces the number of possibilities to a far simpler level of analysis.

## *2.3 Regression Test*

When making changes to the program, it is important to perform regression tests, to ensure that the new changes do not affect the functionality of our program in undesired ways. Regression testing will be important considering that constant changes will be made to our programs as we receive feedback from the client at the end of each iteration.

# **3.0 Testing Strategy**

## *3.1 Test Items*

* Program that connects to the Twitter Streaming API
* Program that manages the database of tweets
* Web interface
* Program to perform statistical analysis

## *3.2 Test Descriptions*

The following section describes the various components of the product that must be tested in order to ensure the product is working as intended. These tests include descriptions and instructions for how to execute them. The date when the test was successfully passed is also given when completed.

**Program/Component**

Connect to Twitter Streaming API

**Test Name**

Connection and Streaming Test

**Test Description**

This test is to ensure that information can be pulled from the Twitter Streaming API.

**Test Instructions**

From your internet browser, go to the path that streamin\_api2.php file is in (refer to readME for instructions to access this file). Tweets should be appearing on the screen in real time

**Result**

Passed 04/05/2016

**Program/Component**

Display tweets to the User

**Test Name**

Tweet Display Test

**Test Description**

Verify that tweets are being displayed by making streaming request and ensure output from the stream is directed to the display.

**Test Instructions**

From the main webpage, check that tweets are being displayed in a readable format. If tweets are not in JSON format and appear on the screen then the display feed is working.

**Result**

Not yet tested

**Program/Component**

Store tweets in the database

**Test Name**

Database Storage Test

**Test Description**

Connect to the database throughout the running of the application to verify that data is being stored

**Test Instructions**

Access the database and ensure that Twitter data has been stored

**Result**

Not yet tested

**Program/Component**

Retrieve tweets from the database

**Test Name**

Database Retrieval Test

**Test Description**

Ensure tweets can be successfully pulled from the database and displayed to the user

**Test Instructions**

Use search term or stock symbol in order to reference tweets stored in the database and display to the user in a scrolling fashion

**Result**

Not yet tested

## *3.3 Test Checklist*

A checklist for testing will also be used. This checklist gives a description of all of the test and a grading system for whether or not the tests pass or fail. This checklist is delivered alongside the Test Specification and all of the other documents.

## *3.4 Desired Test Results*

The following lists the desired functions of the tested programs once they have all been tested.

* Program that connects to the Twitter Streaming API
  + The script connects to the Twitter Streaming API
  + The script
* Program that manages the database of tweets
  + Tweets correctly flow into the database
  + Tweets correctly be retrieved from the database.
  + Data in the database is accessible via a query to filter types of tweets
* Web interface
  + Correctly launches and presents to the user the website for streaming tweets and performing analysis
  + Connects to all other components of the software to display required tweets and statistical analysis to user
* Program to perform statistical analysis
  + Collects all information about tweets to prepare for analysis
  + Performs precise statistical analysis
  + Connects with the web interface to display analysis to the user

# **4.0 Testing Tools and Environment**

All software testing will be done locally on the personal computers of all team members, or the computers in the lab of S&T 127. When connecting to the Twitter Streaming API, tests are run on a local web server, such as XAMPP.

# **5.0 Response to Failure**

Any tests that we have ruled failed will be noted and will be added to a backlog of problems that need to be fixed. All problems will be ranked based on the severity of their functionality. An example of the log is below:

|  |  |  |
| --- | --- | --- |
| **Unresolved Issues** | | |
| ***Number*** | ***Test Failure*** | ***Priority*** |
| 1 | Connecting to streaming API times out | 1 |
| 2 | Statistical analysis returns incorrect data | 3 |
| 3 | When stream is stopped, tweets that were made during the period the stream was not live do not get displayed to the user | 3 |

The priorities of each task range from “1” being of the highest priority, to “5” being the lowest priority.

At the end of each iteration, all unresolved issues will be noted in the documentation and README file before product delivery to the client.

# **6.0 Version History**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Document** |
| 1.0 | 3/16/2017 | Test Specification Version 1 |
| 2.0 | 4/6/2017 | Test Specification Version 2 |
| 3.0 | 5/2/2017 | Test Specification Version 3 |