Software Test Plan

Version 1.0
10.30.2012

Political Mood Gauge Based on Social Media

Team Vikings:

Gresham Schlect
Serendel Macphereson
Ryan Sacksteder

Prepared for CS383
University of Idaho - Dr. Paul Oman
Fall 2012

Table of Contents

1.0 Introduction

1.1 Purpose

The purpose of this document is to show tests applied to the Political Mood Gauge application. The intended audience includes the University of Idaho, specifically Software Engineering 383 (instructed by Dr. Oman), developers, and anyone wishing to run maintenance on the system.

1.2 Scope

This document will describe the tests to be performed on the application. The tests will include individual unit testing of each section of the application, integrative testing where multiple units will be tested together, interface testing, performance testing, regression testing, and plans for customer acceptance testing. For each type of testing, specifics on scheduling, resources, and deliverables are also discussed.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Social Media - technologies that create online communities/networking capabilities, e.g. Twitter, Facebook, MySpace and/or internet blogs

1.3.2 Twitter - a social media website that allows users to publicly share short sentences, or Tweets

1.3.3 Tweet - a text string posted by a Twitter user (140 characters or fewer)

1.3.4 Retweet - the process of reposting someone else's tweet on one's own profile

1.3.5 Follower - one user on Twitter can 'follow' another in order to directly receive their Tweets

1.3.6 Sentiment Analysis - the process of analyzing a statement and determining its degree of positivity or negativity (also known as sentiment)

1.3.7 HTML - HyperText Markup Language, a basic web page language

1.3.8 JavaScript - a language used in web sites

1.3.9 JSON - JavaScript Object Notation, the blocks of information returned from Twitter

1.4 References

Twitter: https://twitter.com/about
Alchemy: http://www.alchemyapi.com
JSON: http://www.json.org/
SRS: Political Mood Gauge Based on Social Media
SDD: Political Mood Gauge Based on Social Media
STP Template**:** www.epmo.scio.nc.gov/library/docs/testplan.doc (State of North Carolina)

1.5 Test Overview

1.5.1 Organization

Software testing is a process that will continue throughout the duration of the project. It will be both expanded and improved upon as more functionality and complexity is added to the application. Most testing will take place in the form of Unit Tests, Integration Tests, and Regression Tests, with other testing suites being implemented where appropriate.

Unit Tests will aim to cover the highest percentage possible of the overall code base. They will test individual modules of the software. Integration testing will be employed over multiple unit tested code blocks. They will test that a collection of units work together correctly. These two forms of testing will form the basis of our Regression Testing which will be executed whenever new source code is to be committed for the project. The Regression Testing will ensure that new code doesn't hurt the integrity of the existing functionality. Other forms of testing will include Performance Testing and Interface which will likely be implemented manually and less often than the previously mentioned testing suites. Also Acceptance Testing will be conducted with the customer at the end of the development cycle. The following sections of this document outline each stage of testing in greater detail.

1.5.2 Environment

The testing environment will mirror the expected system environment of the software as closely as possible. Testing will be run within a Linux environment on a local machine. Output will be tested within common web browsers. If time permits the implementation of a multi-user system over a server, the software will be tested on a server as well.

1.5.3 Responsibilities

Project team members will share the testing responsibilities equally. Code that is ready to be committed will be tested by the individual performing the commit. The design and implementation of these tests, as well as the scheduled running of these tests shall consist of an equal contribution from each team member within the group.

1.5.4 Master Test Schedule

The construction of initial Unit Tests will be completed by Thursday, November 8, 2012 with additional tests being added as needed. The construction of initial Integration Tests will follow the same schedule as the Unit Tests. Regression Testing should be added as a new build rule by Saturday, November 10, 2012. Manual tests such as Interface, and possibly Performance, Testing will be agreed upon and constructed by the project team by Saturday November 10 as well. The software code complete goal is set for Thursday, November 15, 2012 with testing continuing up until (and after) release. Acceptance Testing is expected to take place sometime during the week of November 29, 2012 through December 6, 2012.

1.5.5 Tools, Techniques, Methods, and Metrics

Bash scripts will be used to run the unit testing on the individual modules of the application. A new Makefile build rule will allow for these tests to be run under Regression Testing. Other forms of testing such as Interface Testing and Performance Testing will more than likely be run manually, and tested visually, but may be implemented through some sort of script as well.

1.5.6 Items to be Tested

- Tweet Retrieval/Input
- Tweet Filtering
- Tweet Weight Calculation
- Sentiment Analysis Score Retrieval
- Tweet Value Calculation (Weight x Sentiment Score)
- Daily Average Calculation
- Recovery of Daily Statistics
- User Interface
- CSV Output
- Product Performance

1.5.7 Items Not to be Tested

- Sentiment Score Calculation (API Dependent)
- Twitter Keyword Search (API Dependent)
- Accuracy of Graphical Outputs (API Dependent)

2.0 Unit Testing

2.1 Overview

Unit Tests (white box testing) will be written and performed for each module of source code. Tests will be written to exercise the maximum number of cases and branches in the code; the goal should be to execute every line of code during the unit testing process.

2.2 Resources

A Bash script will be used to run tests on each module. The script will run a module with a given input file, and compare its output to a given output file. If the outputs match, then the module is working correctly - if not, the test failed.

2.3 Deliverables

Each full run of the unit tests will produce text report detailing the results for each test; if a test fails, the report will show the expected and actual result of the test. Finally, the output will show the total number of failing and passing tests, and the pass rate.

2.4 Schedule

The Unit Tests will be run as part of the Regression Testing - every time a change is made to the code and committed to the source repository. The test suite will also be run if any tests are added or modified.

3.0 Integration Testing

3.1 Overview

Integration testing will be performed manually. Multiple modules will be piped together, either using static input, or using the beginning input supplied by the Twitter API in the tweet retrieval class. The output will be manually inspected. If the html output module is being tested, the results may be manually inspected on the webpage.

3.2 Resources

Integration testing will require only the current build of the project.

3.3 Deliverables

The deliverables for integration testing will be a list of all faults exhibited by the software. These bugs should be logged in the bug tracker and fixed. If integration testing uncovers a bug during active development, the bug need not be logged if it is fixed immediately.

3.4 Schedule

Integration testing should be run throughout development, and for the three weeks leading up to release when development has stopped (except for bug fixes). Integration tests may also be run during regression testing.

4.0 Regression Testing

4.1 Overview

Regression tests should be run for each build of the project (before each commit). The regression tests will ensure that the latest changes to the code to not break pre-existing functionality. As more functionality is added, more cases should be added to the regression testing suite to ensure that they are preserved in future versions. The regression testing will consist of unit testing, and one pass of integration testing (running all the modules together for the full application).

4.2 Resources

The regression test will require the unit test suite and the most recent build of the project.

4.3 Deliverables

The regression test should show pass/fail for each unit test. If the regression tests do not pass, the developer should not commit the code unless the tests need to be altered or requirements for functional requirements change.

4.4 Schedule

Regression tests will be run before each commit.

5.0 Performance Testing

5.1 Overview

Performance testing will be performed manually. Performance Testing will test the overall execution speed of the software as well as the throughput of data through the modules. The overall tweet throughput of the software (rate at which tweets are processed and results are output) will be a good measure of product performance. Performance testing should be carried out before all new commits. This will prevent performance declines which result from poorly written code and algorithms.

5.2 Resources

The performance testing will require the latest build of the project and knowledge/documentation of previous performance statistics.

5.3 Deliverables

Performance testing should deliver a list of anomalies and discrepancies with performance requirements placed on the software. These should be entered into the bug tracker, unless they are found during active development and can be immediately resolved. General statistics such as tweet throughput and execution speed should be noted for comparison with later builds.

5.4 Schedule

Performance testing will be carried out throughout the duration of the project, especially before each new commit. This will allow for quick correction of performance issues that result from new features/ implementations.

6.0 Interface Testing

6.1 Overview

Interface testing will be performed manually. The testing should cover three areas: the webpage, csv output, and configuration. To test the webpage, a project member will run the project watch the web output, ensuring that there are no anomalies. Zooming and hovering functionality on the graph should also be tested.

6.2 Resources

Interface testing will require the latest build of the project.

6.3 Deliverables

The interface testing should deliver a list of anomalies that will be entered into the bug tracker. If the anomaly is discovered during the immediate development progress, and can be fixed immediately, it need not be entered in the bug tracker.

6.4 Schedule

Interface testing should be performed as the interface is developed, and during the last three weeks of the project after active development has stopped (except bug fixing).

7.0 Acceptance Testing

7.1 Overview

Acceptance testing will be carried out manually while meeting with the customer. The customer will be presented with a demo of the application, and his feedback will determine future development, and if any is necessary.

7.2 Resources

Acceptance testing will require the latest prototype of the software.

7.3 Deliverables

The deliverables of acceptance testing will be the feedback of the customer, translated into requirements specification modifications, or anomalies to be entered in the bug tracker.

7.4 Schedule

Acceptance testing will be performed at the end of the project, primarily during the final presentation. Due to the academic environment, the feedback of the customer may or may not be enacted in any future development.