Team Bricksquad

**Final Version**

Test Specification

**Table of Contents**

[1. Introduction 2](#_Toc384205313)

[*1.1 Project Overview* 2](#_Toc384205314)

[*1.2 Purpose* 2](#_Toc384205315)

[*1.3 Scope* 3](#_Toc384205316)

[2. Definitions, Acronyms, and Abbreviations 4](#_Toc384205317)

[3. Testing Conditions 4](#_Toc384205318)

[*3.1 Pass / Fail* 4](#_Toc384205319)

[*3.2 Suspension* 4](#_Toc384205320)

[*3.3 Resumption* 4](#_Toc384205321)

[*3.4 Approval* 4](#_Toc384205322)

[4. Testing the Tweet Analyzer 5](#_Toc384205323)

[5. System Resources 5](#_Toc384205324)

# Introduction

*1.1 Project Overview*

BrickSquad has taken on the task of constructing a general-purpose Tweet Analyzer. General purpose meaning a technical or non-technical customer could use custom or written classes to link to the tweet analyzer to perform such simple or complex analysis of tweets. The Tweet Analyzer program will contain two classes’ regular expressions and the other weather map. This document includes information on the project estimates as well as the resources, design plan, risk management, team project schedule, iteration project planning, and our milestones. Each will examine the different aspects of general-purpose tweet analyzer.

*1.2 Purpose*

The purpose of this document is to clearly and concisely lay a road map for the requirements of our project as well as establish a scope. On top of this, we'll be taking into account our potential users of the application, project estimates, stakeholders, and our overall iteration plans. Within this document are adequate details of our tweet analyzer. Also it will detail the impact this application is expected to have. This initial plan will be a skeleton that's laid out throughout the stages of our project. This will ensure that all readers understand BrickSquad’s tasks at hand regarding the development of our tweet analyzer.

*1.3 Scope*

The scope of this project is to construct a general-purpose tweet analyzer that will work will simple classes as well as complex classes. The tweet analyzer program must be able to link to the tweet analyzer class using regular expressions to find matching tweets. In addition to, the program must also be able to link to the weather class that will produce a weather map based on tweets in the United States. This is both for personal as well as business use.

Scope of BrickSquad Tweet analyzer program:

* Tweet analyzer class using regular expression
* Tweet analyzer class used to produce weather map

The target audience of BrickSquad application is estimated, but not limited to, the general population; this application is projected to be used by a variety of people, but we want it to be user-friendly enough to a degree that even those with little technology experience can use this program. This tweet analyzer is to be very simple and to use.

# Definitions, Acronyms, and Abbreviations

* API (Application Program Interface) – a programming language and messaging format that allows two different applications to communicate, or interface with each other
* JSON (JavaScript Object Notation) -is a lightweight data-interchangeable format
* Twitter- an application that is used to express a thought.
* Tweet-a message that a Twitter user posts to his or her profile page.
* Follow - to subscribe to another user's tweets.
* Retweet - to forward a tweet posted by someone else to all of your followers.
* Reply - a response to an existing tweet, posted by clicking the "reply" button. The reply text automatically begins with @username (the username of the person to whom you are replying). All replies to a user's tweets are logged under the @Mentions tab on the user's home page.
* Mention - a tweet update that contains @your username anywhere in the body of the tweet. Both new tweets and replies can be considered mentions.
* Direct messages (DM)- Messages to another users that can only be seen between the two users
* OAuth- authentication protocol to provide authorized access to its API
* Regular expressions - A regular expression is a sequence of characters that forms a search pattern, mainly for use in pattern matching with strings, or string matching.

# Testing Requirements and Conditions

*3.1 Pass / Fail*

Outlined within this section are the criteria by which a particular test item has passed or failed its testing. If the testing has failed another test will be completed until the status remains passing.

*3.2 Suspension*

A particular test activity will be suspended if the acquired result does not match expected result and has a negative impact on one or other components or relationships. This impact is to be determined by the team, and will be adjusted for purposes benefiting the user's interaction with the application.

*3.3 Resumption*

Suspended test items will be resumed upon proper evaluation of conditions. Original testing items will be repeated upon resumption to ensure both efficiency and validity.

*3.4 Approval*

Approval conditions will be met when results of the appropriate test case are as expected, and have no harm or negative effect on any of the components related. This is to be determined by the developer upon extensive testing. Project Manager should also approve all testing and test results.

# Testing Items

Components of the Bricksquad’s application will be tested accordingly and the developers will track tests and updates made as a result of testing. Each component however will have test cases that vary. The weather map, regular expression text file, website, and tweet analyzer components will interact with a user. Thus, within that components test will be user procedures and user scenarios. As for the database components of our application, each will test functionality, operations, and features as well as plans for verification and validation. Also will be procedures to ensure that each component can be supported from a production perspective.

* Website (Front End)
* Database
* Regular Expression Text File
* Tweet Analyzer
* Weather Map

*4.1 Integration Testing*

Integration testing involves the testing of each possible scenario for the application itself. These will be sequentially and systematically be tested one by one. Component integration will be tested according to component relationship in our activity diagram further explained in the analysis document.

## *4.2 Validation Testing*

Validation testing will demonstrate conformity with the application according to what has been described in the requirements document.

## *4.3 System Testing*

The purpose of tests run on the system will be to fully exercise the sample platform being used, involving operating system and hardware specifications mentioned within the requirements documentation.



# Component Testing

# *5.1 Website (Front End)*

|  |  |  |  |
| --- | --- | --- | --- |
| What will be tested | Test Dates | Pass or Fail | How it will be tested & what is needed to pass |
| Navigation | March 28  April 3  April 6  April 15  April 20  April 24  April 27 | FAIL  PASS  FAIL  FAIL  PASS  PASS  PASS | * Components and pages of the application will be opened and need to be easily accessible for user. After the application is open all the assessable icons/buttons need to have successful opening |
| Consistency | March 28  March 30  April 2  April 15  April 16  April 24  April 27 | PASS  FAIL  FAIL  PASS  PASS  PASS  PASS | * Once the application is open the interface will be tested to ensure that it is consistent simultaneously. Each window available must display and work properly |
| Response time | March 30  April 6  April 10  April 16  April 21  April 23  April 27 | FAIL  FAIL  PASS  FAIL  PASS  FAIL  PASS | * The application will be tested to ensure that that the GUI responds to user input in a quick manner. |
| Error handling | March 30  April 6  April 11  April 16  April 21  April 25  April 27 | FAIL  FAIL  FAIL  PASS  PASS  FAIL  PASS | * Once the application is open the application must not stall the application. There should be no runtime errors |

*5.2 Database (back end)*

|  |  |  |  |
| --- | --- | --- | --- |
| What will be tested | Test Dates | Pass or Fail | How it will be tested & what is needed to pass |
| Correlates correct tweets based on search | April 15  April 19  April 20  April 21  April 24  April 26  April 27 | FAIL  PASS  FAIL  FAIL  PASS  PASS  PASS | * Once the database connects to twitter, based on the twitter search the results match those related to the actual search |
| Creates tables with results | April 15  April 17  April 22  April 23  April 25  April 26  April 27 | PASS  FAIL  FAIL  PASS  PASS  PASS  PASS | * After the search is finished the results print out in formatted tables based on search results |
| Returns Locations from Tweets | April 15  April 19  April 20  April 21  April 24  April 26  April 27 | FAIL  FAIL  PASS  FAIL  PASS  FAIL  PASS | * Returns the location of the tweets retrieved from the search results |
| Store each regular expression in separate database table | April 15  April 17  April 22  April 23  April 25  April 26  April 27 | FAIL  FAIL  FAIL  PASS  PASS  PASS  PASS | * Each regular expression will be stored in appropriate table |
| Database Able to Retrieve Tweets from Twitter | April 14  April 15  April 21  April 24  April 25  April 26  April 27 | FAIL  PASS  FAIL  FAIL  FAIL  FAIL  PASS | * Using the command prompt the database can successfully retrieve tweets from twitter based on search |

*5.3 Regular Expression Text File*

|  |  |  |  |
| --- | --- | --- | --- |
| What will be tested | Test Dates | Pass or Fail | How it will be tested & what is needed to pass |
| Text File Uploads | April 20  April 21  April 24  April 26  April 27 | PASS  FAIL  FAIL  PASS  PASS | * When the text file is uploaded the list of regular expressions will be transferred to a data structure. There should be successful finding of the text file. |
| Read from Text File | April 22  April 23  April 25  April 26  April 27 | FAIL  FAIL  PASS  PASS  PASS | * Once the text file is uploaded the application can read the text file and begin execution |
| Verify File | April 20  April 21  April 24  April 26  April 27 | PASS  FAIL  PASS  FAIL  PASS | * Upon file upload, a test will be conducted to verify whether the file that has been uploaded is an actual text file |
| Empty Text File | April 22  April 23  April 25  April 26  April 27 | FAIL  PASS  PASS  FAIL  PASS | * The uploaded test file will be tested to determine whether or not the file is empty. |
| Text File contains Regular Expression | April 21  April 24  April 25  April 26  April 27 | PASS  FAIL  FAIL  FAIL  FAIL | * Upon the file upload the text file uploaded has to contain regular expressions |
| After Reading Analyzer Executes | April 20  April 21  April 24  April 26  April 27 | PASS  FAIL  FAIL  PASS  PASS | * The tweet analyzer executes search based on regular expressions found in text file |
| Search Multiple Regular Expressions | April 22  April 23  April 25  April 26  April 27 | PASS  FAIL  FAIL  PASS  PASS | * The tweet analyzer uses and or operator to search multiple regular expression |

*5.4 Tweet Analyzer (front end)*

|  |  |  |  |
| --- | --- | --- | --- |
| What will be tested | Test Dates | Pass or Fail | How it will be tested & what is needed to pass |
| Search Twitters Real-Time Feed | April 20  April 21  April 24  April 26  April 27 | FAIL  FAIL  FAIL  FAIL  FAIL | * The search being performed are connecting to twitter real-time tweets |
| Sorts Tweets Based on Search | April 22  April 23  April 25  April 26  April 27 | FAIL  FAIL  FAIL  FAIL  FAIL | * After search is performed tweets sorted based on search results |
| Able to Retrieve Tweets | April 20  April 21  April 24  April 26  April 27 | FAIL  FAIL  FAIL  FAIL  FAIL | * The analyzer will connect to twitter to search through tweets to produce search results |

*5.5 Weather Map*

|  |  |  |  |
| --- | --- | --- | --- |
| What will be tested | Test Dates | Pass or Fail | How it will be tested & what is needed to pass |
| Fetches Tweets from Twitter (front end) | April 20  April 21  April 24  April 26  April 27 | Failed  Failed  Failed  Failed  Failed | * Temperatures will based on twitter tweets and not weather.com or weather site |
| Fetches tweets from twitter (back end) | March 20  March 21  March 24  March 26  March 27 | Pass  Pass  Pass  Pass  Pass | * Temperatures will based on from twitter and not weather.com or any other weather site |
| Finds Correct City Search | April 22  April 23  April 25  April 26  April 27 | Pass  Pass  Pass  Pass  Pass | * Searching for a city on weather map, the map zooms to find the correct location |
| Matches Temperature to Tweets | April 20  April 21  April 24  April 26  April 27 | Failed  Failed  Failed  Failed  Failed | * Temperatures match tweets based on locations |
| Populates Weather Map Based on Tweets | April 20  April 21  April 24  April 26  April 27 | Failed  Failed  Failed  Failed  Failed | * Weather map connects to twitter, analyzer populates weather map with temperatures based on tweets from twitter |

# System Resources

|  |  |
| --- | --- |
| **System Resources** | |
| **Resource** | **Name / Type** |
| Computer | Project PC/ Dell Precision Intel Core 2 CPU |
| Operating System | Microsoft Windows 7 |
| Internet Connection | Hampton University  Network |
| Programs | Python Anywhere  My SQL  Eclipse  Google Maps |