­­­­­­­

Team Bricksquad

**Final Version**

Analysis Document

April 28, 2014

Table of Contents

[1 Introduction 2](#_Toc386402908)

[*1.1* *Project Overview* 2](#_Toc386402909)

[*1.2* *Scope* 2](#_Toc386402910)

[*1.3* *Problem Statement* 2](#_Toc386402911)

[2 Requirements 3](#_Toc386402912)

[*2.1* *Technical Requirements* 3](#_Toc386402913)

[*2.2* *Functional Requirements* 3](#_Toc386402914)

[3 Interaction Models 4](#_Toc386402915)

[*3.1* *User Analysis* 4](#_Toc386402916)

[*3.3* *Task Analysis* 6](#_Toc386402917)

[*3.4* *Content Analysis* 6](#_Toc386402918)

[*3.5* *Environment (System) Analysis* 6](#_Toc386402919)

[*3.6* *Interface Analysis* 7](#_Toc386402920)

[3.6.1 General Purpose Tweet Analyzer 7](#_Toc386402921)

[3.6.2 Weather Map 7](#_Toc386402922)

[*3.7* *Configuration* 7](#_Toc386402923)

[4 Behavioral Diagrams 8](#_Toc386402924)

[*4.1* *Sequence Diagram* 8](#_Toc386402925)

[*4.2* *Weather Map Sequence Diagram* 8](#_Toc386402926)

[*4.3* *Activity Diagram* 10](#_Toc386402927)

[*4.4* *Weather Map Activity Diagram* 11](#_Toc386402928)

[*4.5* *Use Case Diagram* 12](#_Toc386402929)

[*4.6* *State Diagram* 12](#_Toc386402930)

[5 Revisions 13](#_Toc386402931)

# Introduction

## *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 a minimum of three classes: The general-purpose Tweet Analyzer class that uses the methods in its class to analyze tweets; a regular expressions class that produces a list of tweets that matches specified keywords; and the weather map class that analyzes real-time tweets across the U.S in order to produce a weather map. After the analyzer is developed BrickSquad must then create a Software Development Kit in order to assist the sophisticated customers that want to write their own Tweet Analyzer class. 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.

## *Scope*

The scope of this project is construct a general-purpose tweet analyzer that will work will simple classes 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.

## *Problem Statement*

BrickSquad will create a web application that will do the following:

* Retrieve tweets from twitter API to perform a regular expression search
* Retrieve tweets from twitter API to create a weather map based on tweets about the weather
* The data retrieved will be stored in a database. The database will be MySQL
* The user will be able to enter expressions for searching twitter to get results needed

# Requirements

BrickSquad is required to develop a program that will allow users to accurately search through Tweets on Twitter using keywords. These keywords are also known as regular expressions. The platform will use the regular expressions to pick out real-time, relevant (those consisting of one or more regular expressions) tweets from Twitter. This allows the user to obtain opinions as well as valuable information from a wide range audience (any Twitter account who tweets). Presented below, are the list of requirements for the system of our Tweet Analyzer. Technical requirements present the requirements for the hardware/software platform, whereas the functional requirements define the requirements for the developed software. The priority ranking below list the severity of completing these requirements by iteration five which is described in timeline of the project plan.

Priorities:

1=must implement

2=should implement (if time allowing)

3=be nice to have

## *Technical Requirements*

1. Technical Requirements

|  |  |  |
| --- | --- | --- |
| #No | **Description** | Priority |
| **1** | The application must run on the Windows platform | 1 |
| **2** | Tweet Analyzer must depend on stable internet connection | 1 |
| **3** | Environment must support Python applications | 1 |

## 

## *Functional Requirements*

1. Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| #No | Description | Referred Use Cases | Priority |
| **1** | The application must be able to grab tweets from API, and store into appropriate database |  | 1 |
| **2** | The application must use regular expressions to search through tweets |  | 1 |
| **3** | Application must create weather map of the US based on weather tweets |  | 1 |
| **4** | Application must allow for user to interrupt (end program on user’s demand) |  | 2 |

# Interaction Models

## *User Analysis*

BrickSquad’s tweet analyzer application will support various users which range from individuals that are familiar with twitter and how it is used to those who have never even registered for an account at Twitter. The design will allow of range users so that either a technically sophisticated customer could write their own class that would be linked with the general purpose Tweet Analyzer to perform a very sophisticated form of analysis; or a non-technical customer could use a “team written” class that accepts regular expressions written by the customer to perform a simple match analysis of tweets. The application will have clear and concise graphical interface making it easy for any user to understand and use. The application will have a clearly marked interface in which users can navigate from a list of streaming tweets depending on the users’ regular expression entered and our (proof of concept) weather map which will be presented to the user based tweets regarding weather. The application can be used by any gender or race but requires necessary knowledge of the English language as well as basic computer skills.

BrickSquad’s application will assist users that:

Have no idea how Twitter is operates or who have never registered for an account (note: BrickSquad’s application does not require a user to register for a Twitter account)

Are requesting for knowledge using tweets streaming from twitter on a particular subject (i.e. noteworthy news, sports, popular cultural etc.) represented by the users regular expression that he or she may enter in

Are requesting for knowledge of weather by region within the Continental United States (i.e. rain, snow, sunny, hot, windy etc.) which will be represented by with a live visual map.

*3.2 Component Analysis*

**3.2.1 Streaming Tweets:** BrickSquad has been assigned the task to develop a general purpose Tweet analyzer program that uses methods to analyze Tweets. In order to perform this function, after the connection has been established, the program must then have a way to search through Twitter. Tweepy is a Twitter API. This API class provides access to the entire twitter RESTful API methods as well as grants us the HTTP request after proper authorization. Each method defined in the API can accept various parameters and return responses. Bricksquad will use this API to carry out required commands that obtain information about each relevant Tweet on Twitter.

This API will be able to use the Twitter API that uses the REST format to provide the relevant tweets in order to search the past for information such as terrorism attacks. It also allows you to use the Streaming API that produces real time tweets once the connection is established and authorized in order to provide current information. Streaming APIs gives a low latency access to twitter’s global stream of Tweet data.

The application gets delivered a feed of Tweets, once a connection is established to a streaming endpoint. An HTTP method for streaming endpoints is called POST statuses/filter. The endpoints for this method returns public statuses, in JSON format, that match one or more filter predicates.

**3.2.2 Filter Twitter:** The Tweet Analyzer program is required to filter the stream of Tweets to provide only those including a regular expression. In order to filter each Tweets, BrickSquad plans to use the POST request because the GET request are often rejected due to excessive URL length. One of the predicate parameters in the POST request is the Track. The Track Parameters are keywords and phrases (regular expressions) that are specified by a comma separated list.

Parse Tweets: information received from tweets: When the main class receives the Tweet from Twitter using the API commands, the tweet is returned in a JSON format. This format is used to be easily interchangeable between languages, and can be deciphered once you parse through the format in a language dependent method. In python to parse through JSON format there are many different implementations you can use. For this specific project we will be using SimpleJSON. SimpleJSON is a simple, fast, complete, correct and extensible JSON encoder and decoder for Python. The decoder will handle incoming JSON strings of the tweets we will be fetching and will help gather only the useful data that the Bricksquad team needs from the tweet which includes the location, an index of matched regular expressions, as well as the actual character string of the corresponding tweet. The number of retweets a tweet is also a measure of engagement. The more users retweet a tweet the more valuable that tweet is.

Tweet Processing Algorithm: An algorithm is used to handle the heavy stream of tweets. This algorithm is responsible for fetching and processing tweets. Each tweet process will recheck the API’s filter in order to ensure the exact regular expression’s match, as well as storing the proper information that needs to be obtained by each tweet in the correct place.

Store Tweets: BrickSquad will be using MySQL in order to store the Tweets we are streaming real-time from Twitter. MySQL is a relational database management system (RDBMS), users use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. After each tweet is arranged accordingly, the tweet will then be stored in MySQL.

Parse regular expressions: Part of the requirements asks BrickSquad to develop a regular expressions class that reads in a file to produce a list regular expression. This class will obtain those keywords by reading in a file containing the regular expressions. After reading the file, each line, containing a regular expression, will be parsed to an array. This array will be used to identify exactly how many times the keyword has been used, as well as for indexing purposes. This list will then been transferred to the analyzer class in order to define the keywords before the twitter search process begins. After each tweet is found an alert must be sent to display the list of found regular expressions that is constantly updated.

Produce list of relevant (Tweet matching keywords) Tweets: The regular expressions class for the non-sophisticated users will produce a list of Tweets that match any of the regular expressions. To produce a visual list of the tweets found using the regular expressions class, an interface is needed for the user to view them. Flask is a free, microframework written in Python, a Graphical User Interface toolkit that BrickSquad has elected to use to display said Tweets.

Display map of real-time US weather map based on predefined regular expressions: Another class must be generated for sophisticated users. This class must use a list of predetermined regular expressions from all over the Continental United States and produce a weather map for the country. Google Maps provides an API for Bricksquad to use. With Google Maps Bricksquad has the ability to design and implement a custom map that displays weather conditions on the US map for specified regions/cities/states. It already provides a weather layer with weather updates, but we are also able to customize that information. Using the Google Maps Javascript API, Bricksquad is able to implement an algorithm that fetches JSON text for different weather conditions in the Continental United States. The Tweet Analysis class will perform an analysis for tweets about the weather related regular expressions and store this information in JSON format for the Google Maps API to fetch. Once it fetches this information it will iterate through the information and populate the different weather conditions on its own weather layer that Google allows us to customize. Google already has a weather layer that uses information from weather.com.

Developing an SDK: Our Software Development Kit (SDK) will assist any user who wants to write their own class in order to implement a Tweet Analyzer class. Software Development kit or sometimes called devkit is a set of software development tools that allows for the creation of application for a certain software package, in this case it is BrickSquad’s Tweet Analyzer class. This will be very necessary in our implementation of BrickSquad’s application.

## *Task Analysis*

**Regular Expression Task list:**

1. Add a text file for the regular expression in specified directory
2. Start the application
   1. Program verifies text file to user
   2. Program displays list of regular expressions before proceeding to analyze tweets
   3. Display to user that tweets are currently being analyzed
3. View of currently processed tweets that match regular expressions
   1. View will include location of tweets
   2. Index of matched regular expressions
   3. Character string of corresponding tweet
4. Constant update of matching tweets

**Weather Map Task list**

1. Start application
2. Process predetermined list of regular expression relating to weather
3. Display to user that tweets are currently being analyzed
4. View of weather map produced by currently processed tweets
5. Constantly update of weather map according to real-time tweets

## *Content Analysis*

The content of Bricksquad Tweet Analysis will be determined by the stakeholders as well as the development team. Any data that will be retrieved from the Tweet Analyzer will be pre-approved with the requirements of the stakeholder. The content will contain the tweets pulled from the search of the user. According to what the user searched will determine how the data will be displayed. The user will be able to perform an efficient search of tweets whether it be simple or complex. The graphics that will be available for the Bricksquad Tweet Analyzer will consist of a graphical weather map of the US showing the weather in a fashion similar to the weather maps shown on national television news shows.

## *Environment (System) Analysis*

Regarding the system, our Tweet Analyzer application will be created for the Windows Operating System and development is being handled on the Windows 7 platform. The Tweet Analyzer will depend on an internet connection to grab tweets from Twitter API. Data will be pulled from the web to be placed within our database for retrieval and storage when needed.

## *Interface Analysis*

## General Purpose Tweet Analyzer

The general-purpose tweet analyzer will be the most important feature of the application. The Tweet Analyzer class for the non-technical customer that will read a file containing regular expressions (note that a keyword is also a regular expression) and applies the regular expressions to tweets producing a list of tweets that matches. Extracted from the full Twitter firehouse, the application stream will deliver full coverage of the Tweets the customer wants based on the filtering criteria they provide.

## Weather Map

The Weather Map component will serve the user by displaying a map of the United States. Within this map there will also be a weather layer on top of the map which will populate the weather forecast from each individual state. The forecast will provide an approximate temperature for each state as well as an image for each state. As the application continues to update the weather map the temperatures may change in different states depending on what data it reads from Twitter. The weather component will work directly with the tweet analyzer class. There are both vital of our user-interface, as the user can let the program run and still be kept up to date with the weather forecast.

## *Configuration*

This project’s configuration is dependent on our stakeholder’s operating system, and therefore must be designed on a window’s environment. The design for the interface used for the weather map, as well as the one used in order to produce the list of tweets matching regular expressions will be implemented in a way that the user will be needed as little as possible to perform any function. The interface will display the list of relevant Tweets, where the Tweets are retrieved from the MySQL framework. The large scale-storing framework will retrieve its information with Twitter Stream API and will use our functions to grab and format the retrieved data into the organized storing framework. The user has the ability to alter the results stored revising and/or replacing the text file with new regular expressions. The interface displaying the real-time weather map of the United States will be displayed after using an algorithm to group regional weather tweets to conclude the weather condition. After the algorithm is complete, the project will use the Google maps API to design and implement the custom map.

# Behavioral Diagrams

## *Sequence Diagram*

This section contains the sequence diagram for BrickSquad’s Tweet Analyzer. This sequence diagram displays how the processes operate with one another and in what order they will be performed.

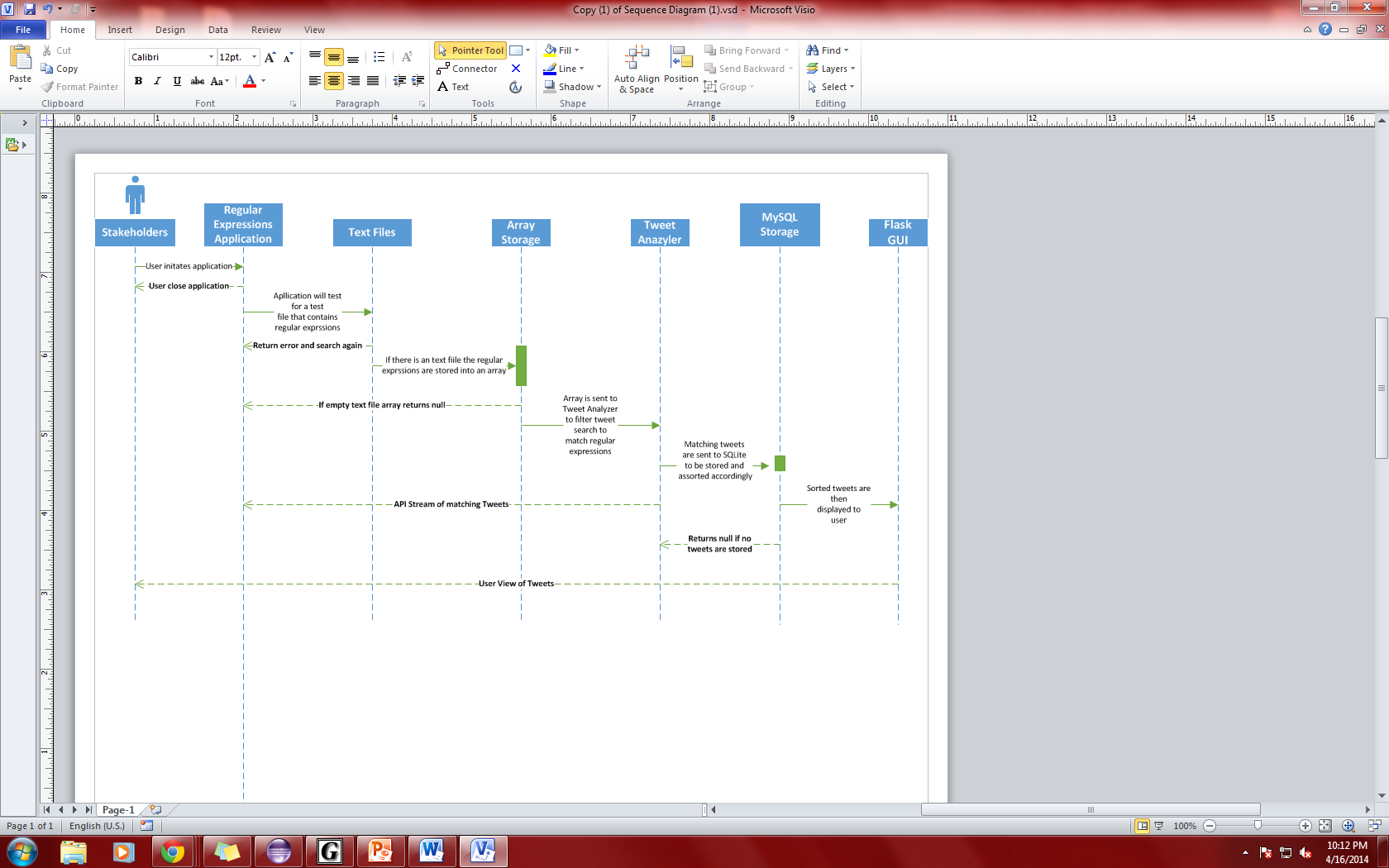


Figure 1: Sequence Diagram

## *Weather Map Sequence Diagram*

This section contains the sequence diagram for the weather portion of BrickSquad’s Tweet Analyzer. This sequence weather map diagram displays how the processes operate with one another and in what order they will be performed.

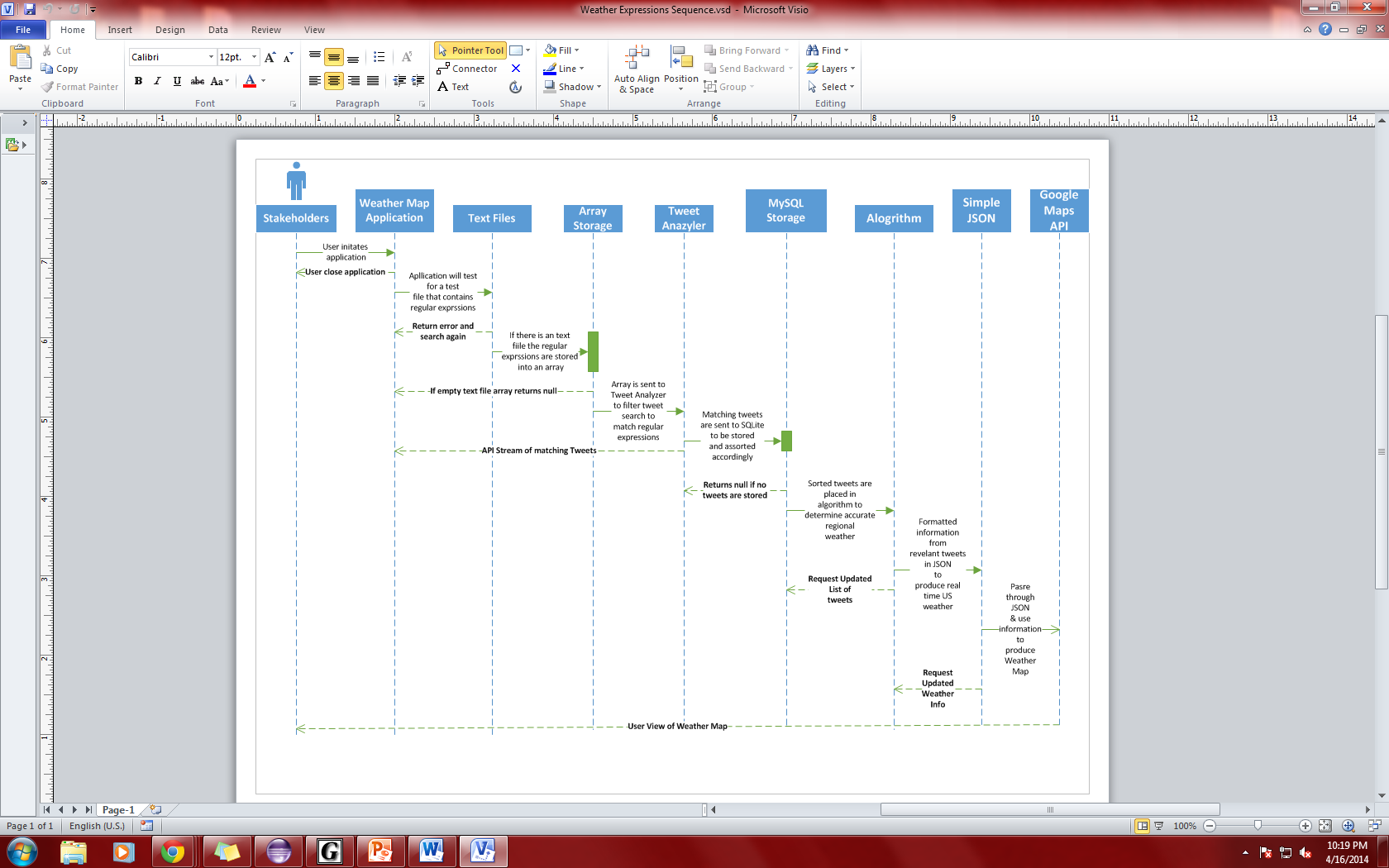


Figure 2: Weather Map Sequence Diagram

This sequence diagram describes that the user is responsible for opening the application and viewing the different pages. The user can make changes to the watchlist, but the user simply views the rest of the application for updated information.

**Actors**: Stakeholder (owner of application)

**Preconditions**: User should have a basic knowledge of twitter but it is not necessary.

**Procedure**: The user starts the service. The user can then view tweets matching the regular expressions of the text file and the weather map produced by the weather tweets.

**Exceptions**: [1] The user has no internet connection.

[2] Twitter is down or over capacity

[3] The user having no file to read from

**Post-conditions**: The user either finds tweets matching the regular expressions read in from the file or no tweets are returned matching expressions. The weather map produced shows the weather based on tweets in the United States

## *Activity Diagram*

The activity diagram shows a graphical representation of user scenarios with the BrickSquad’s Tweet Analyzer.

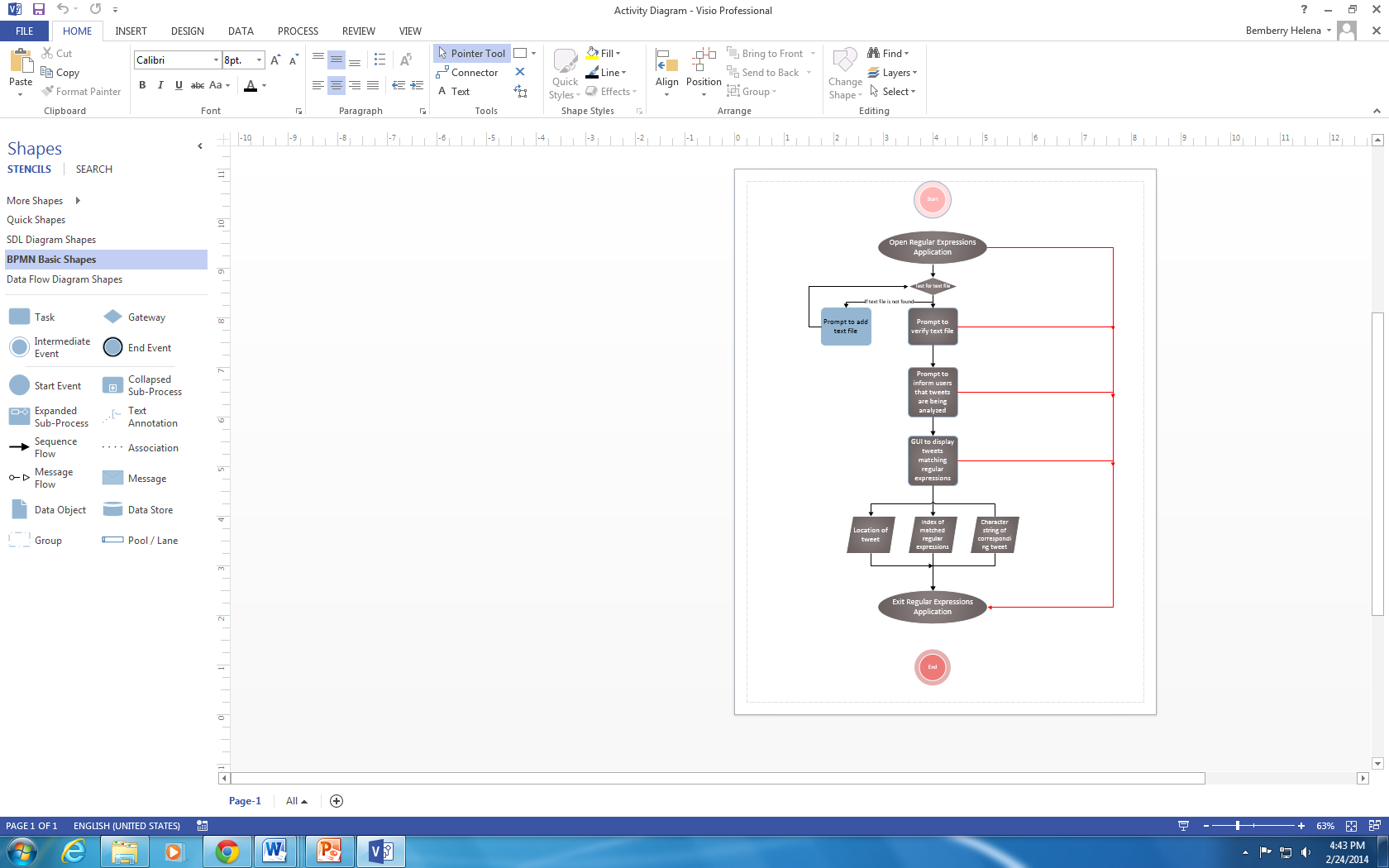


Figure 3: Activity Diagram

## *Weather Map Activity Diagram*

The weather map activity diagram shows a graphical representation of user scenarios with the BrickSquad’s Tweet Analyzer.

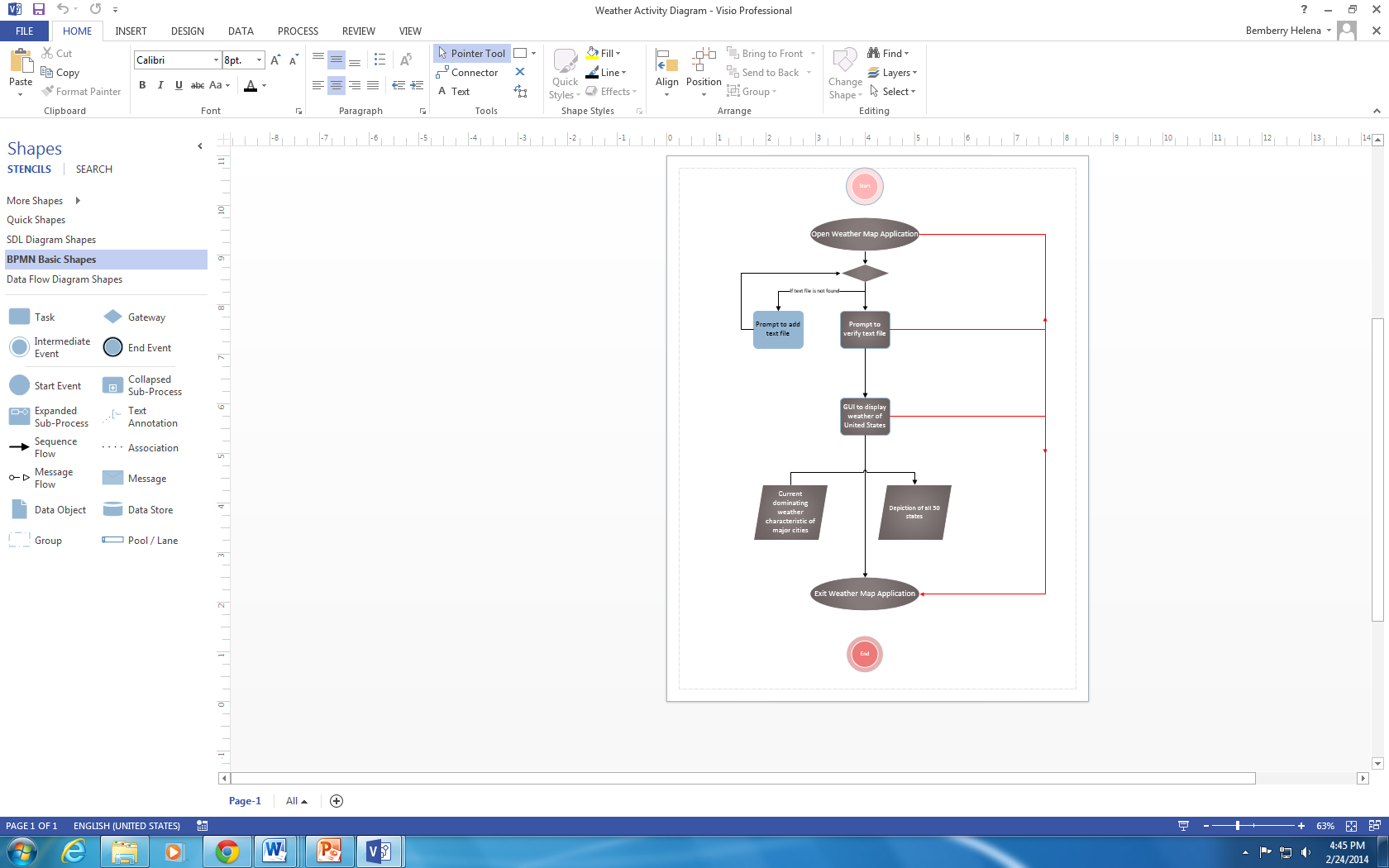


Figure 4: Weather Map Activity Diagram

## *Use Case Diagram*

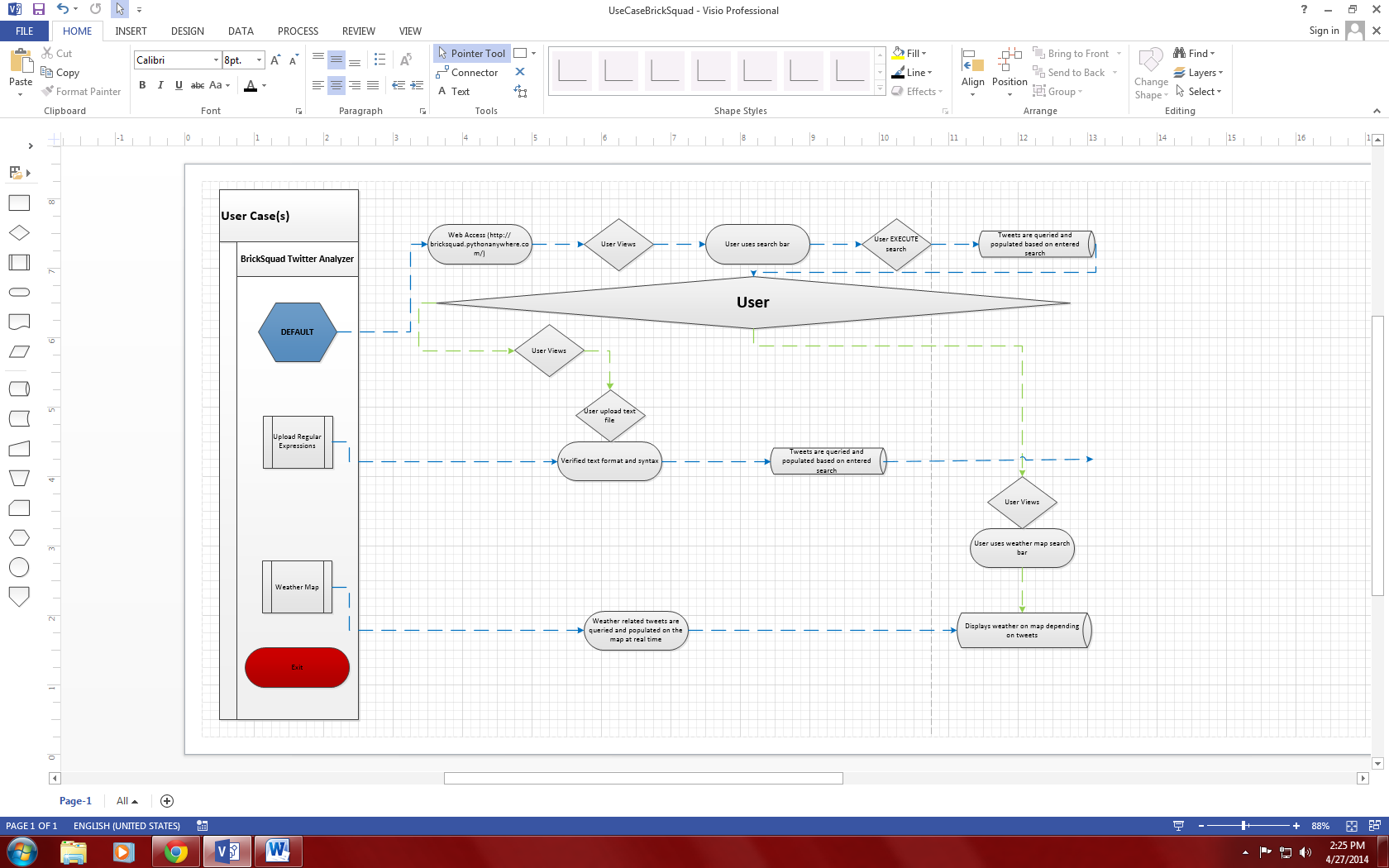


Figure 3: Use Case Diagram

## *State Diagram*

The purpose of the state diagram is to illustrate the behavior of the Tweet Analyzer. The diagram displays the system composed of a finite number of states.

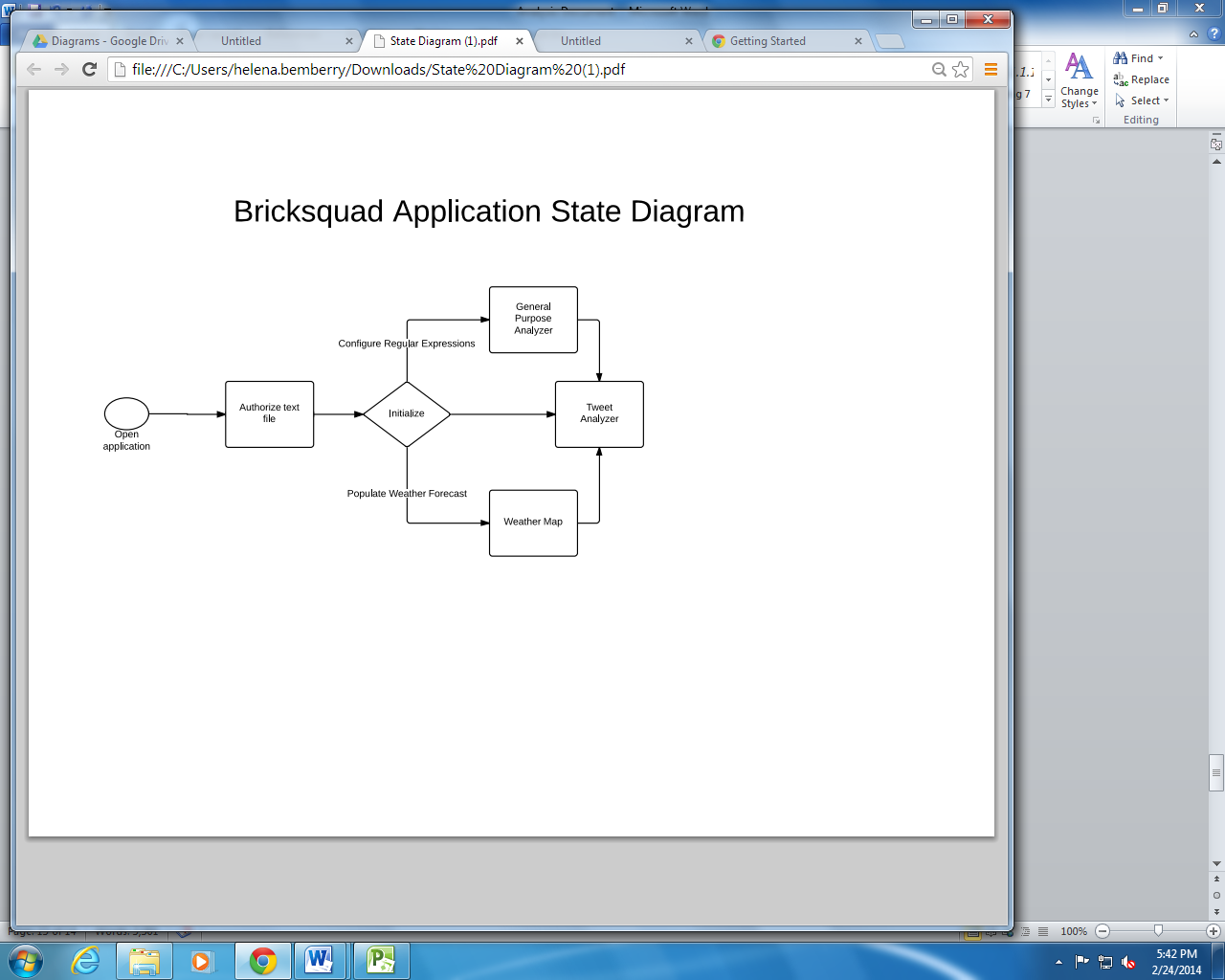


Figure 3: State Diagram

# Revisions

| **Version** | **Date** | **Name** | **Description** |
| --- | --- | --- | --- |
| **1.0** | **2/4/2014** | **Document #1** | **First document** |
| **2.0** | **3/20/2014** | **Document #2** | **Revised document** |
| **3.0** | **4/8/2014** | **Document #3** | **Revised (2)** |
| **Final version** | **4/28/2014** | **Analysis Document** | **Final document** |