**Twitter Fingers**

Functional Specification

Julian Mathis (PM)

Dorian Wood

Anisha Carter

Taylor Clark

Dominique Collins

Sydney Parker



**Contents**

**1.0 Introduction** 3

*1.1 Goals and Objectives* 3

*1.2 Statement of Scope* 4

**2.0 Product Features** 5

*2.1 Generate a Webpage to Display Tweets* 5

*2.2 Store Tweets to the Database* 6

*2.3 Analyze the Tweets* 6

**3.0 User Interaction** 9

*3.1 Web Interface* 9

*3.2 List of Twitter Accounts to Stream* 9

**4.0 Software Specification** 10

*4.1 MongoDB* 10

*4.2 GitHub* 10

*4.3 XAMPPS* 11

*4.4 Text Editor* 11

**5.0 Software Function Priorities** 11

**6.0 Performance/Behavior Issues** 12

*6.1 Management and Technical Constraints* 12

**7.0 Version History** 13

# **1.0 Introduction**

There are many different ways to gain the financial knowledge to make successful decisions while investing in the stock market. From being a financial advisor yourself or hiring someone to manage your stock portfolio. What would be a good way to learn and watch your stocks and stocks of companies you may want to invest with? With social media being such a large part in people’s lives, currently financial advisors and investors have moved to Twitter as an outlet to inform people of the company’s stock. The clients have tasked our team to gather the important financial tweets from different financial Twitter accounts to be viewed on a webpage that can be scrolled through as they are tweeted in real-time. In addition, the task our team was given was to store the tweets into a NoSQL database so the tweets can be traversed through by pausing the incoming feed, and grouped by relation. This information from all the tweets will be analyzed to give a projection of the effects of the stock market. This document outlines our team’s functional approach to delivering this product, and how we will plan and execute these functions.

## *1.1 Goals and Objectives*

The goal of our team’s project is to create an application that will pull financial and stock market related tweets form different financial tweeters, and display the information on a webpage in a scrollable panel. The tweets will then be stored in a database, and will be able to be displayed to the user based on queries and other unique identifiers such as their stock market symbol, Twitter account, and other attributes related to the tweet we see to be important. Information from the tweets will arrive on the webpage in real time and continuously update to handle the incoming new tweets. The purpose of this project is to make users aware of current events in the financial and stock market world, so that investment decisions can be effortless and made swiftly.

## *1.2 Statement of Scope*

The team received an outline of the tasks they must complete.

* Create a fully interactive interface
* Pull tweets in real-time using a Twitter Streaming API
* Plan and program a Database that stores the tweets as well as information within tweets
* Display previously presented tweets from the database to the user
* Organize information within the database
* Have this stored financial information be accessible using a company’s stock symbol and various other queries
* Analyze tweets and information to provide statistics to the user

The scope of this project will be creating an application that will provide the user with new and relevant financial information. We will develop a program that easily and elegantly displays finance news to the user. To handle all of the information coming in from Twitter, we will need to store and organize this information in a database. We will use a NoSQL database to access our information. One aspect of this program will be that the user will be able to choose what Twitter accounts they will pull the information from, thus giving the user a personalized experience. The target audience for our program is for anyone who wants to be aware of the latest financial information. This application will be useful for experiences finance minded individuals, and newcomers to the world of finance.

# **2.0 Product Features**

This section describes the major features that will be implemented in the application.

## *2.1 Generate a Webpage to Display Tweets*

Currently the web application is accessed via the web server, by typing the address of the PHP script in the address bar of the web browser (ex. localhost/webapp.php). The program that will generate the webpage must be capable of displaying the Twitter data from certain Twitter accounts that tweet finance, investment, and stock related information. In addition, this program will display these tweets in real time. This feature will work by using an API to gather the specific tweets to display. The script to connect to the API pulls tweets from the Twitter servers and feeds the tweets and information to the webpage to display to the user.

Once the tweets have been displayed on the webpage, the user will be able to scroll a display feed on the website to view tweets. The webpage serves as the base of operations for the user. From the webpage, the user can start and stop the stream of tweets, scroll through the displayed tweets to read them more closely, filter what tweets they want to have displayed, and perform a statistical analysis on the tweets.

The webpage can be launched by following the instructions listed in the readME file that is delivered along with the source code for the application. Once the webpage is displayed, the team logo, names of the team members, and all of the features of the application will be presented to the user. This application will only be one webpage, which will serve as the control page, where all features of the application can be executed. The control page will also contain the window where the incoming tweets will be shown. This feed of incoming tweets can be started and stopped with the click of a button on the webpage.

## *2.2 Store Tweets to the Database*

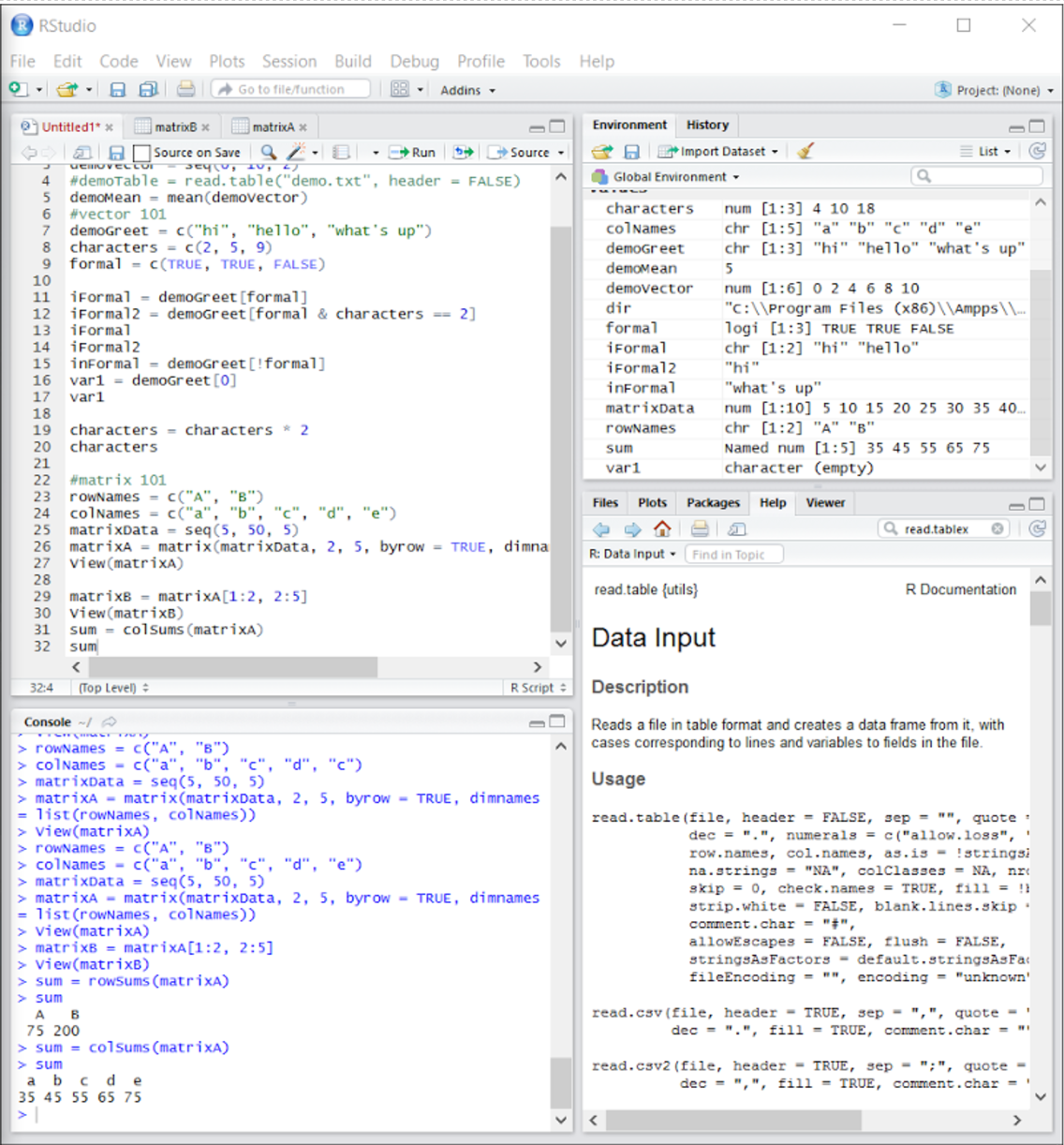
In addition to displaying tweets to the user, these tweets will be stored in a NoSQL database. Since the tweets will be displayed in real-time, we cannot display all tweets on the screen at once. We will have the tweets in a scrollable panel, which will remove tweets from the display once a certain amount is on the screen, to make room for newly tweeted tweets to display. This may cause a user to not see a tweet, or perhaps a user would like to revisit a tweet that was already displayed. Storing the tweets in a NoSQL database after they have been shown to the user allow for tweets to be referenced again even if they are no longer in the current display. With our information stored in a database, we will be able to choose which tweets we would like to display. The user will be able to query which type of tweets they would like to display, based on the content of the tweets, and the Twitter account that tweeted the tweet. We will design this NoSQL database using MongoDB. This will allow us to store the tweets in JSON format, which make accessing and parsing the content of the tweets much easier.

## *2.3 Analyze the Tweets*

Our application will have a function that will analyze tweets that are stored in the database, and provide statistics about the tweets, and content within the tweets. This will allow for the user to have useful information about the tweets. For example, this information can guide the user in making important decisions when it comes to buying stock and investing. These statistics we plan to provide will be computed using the R programming language. We will provide numerical statistics as well as graphs depicting these statistics.

Our application will provide useful statistics and graphical representations of data to the user to help them make handle financial decisions. Analysis of the data would include:

* Number of tweets per hour
* Most frequent tweeter
* Most popular object of tweets
* Frequency of tweeters per day or per week



*The image above shows the RStudio IDE used by the team to provide analytics using R. The window labeled ‘Untitled1’ in the top left shows code for R basics including vectors, mean, matrices, and more. The Global Environment window in the top right shows the developer all the variables and their values. The Console window in the bottom left show the output from the code in Untitled1. The bottom right has several tabs that are useful for the developer including: Files, Plots, Packages, Help, and Viewer. The Help tab allows developers to search R documentation for deeper explanations on R methods and attributes*

# **3.0 User Interaction**

This section will describe the different aspects of the application that the user will interact with. The user will not be able to interact with the database, or any back-end code.

## *3.1 Web Interface*

The user will interact directly with the web interface. The interface will be a webpage that contains the application. The web interface has access to the three main functions of the application; starting the streaming tweets, choosing which tweets to display from the database, and analyzing the tweets. The web interface will connect to all the back-end code that will contain the functions. We will design this interface using HTML, JavaScript, CSS, and PHP.

## *3.2 List of Twitter Accounts to Stream*

The user will be able to choose which Twitter accounts they would like to pull tweets from. There will be a list that the user can append to as well as subtract from. The user will be able to type in the Twitter account they want add to the list in a text box on the web interface. The account will then be added to the file containing the list of Twitter Accounts. The user will then be able to stream tweets from the newly added account almost instantly. This list is referenced by the PHP script that makes the calls to the Twitter Streaming API, so that it knows exactly which users to pull tweets from.

# **4.0 Software Specification**

This section specifies the software and tools used by Twitter Fingers during the development of the application.

## *4.1 MongoDB*

We will use MongoDB as our database to store our tweets. The advantages and disadvantages of MongoDB are listed below.

**MongoDB Advantages:**

* **Easy to Use:** Easy to install, and thanks to a bevy of third-party tools that the database can use, setting up an implementation is a relatively simple task.
* **Inexpensive:** The community edition, which is the one we are using, is free.
* **Adaptable:** It is compatible with virtually every operating system.

**MongoDB Disadvantages:**

* **Functionality:** Tends to be very dependent on add-ons. Certain features, such as text search, are dependent not on the core engine but on applications and add-ons.
* **Scaling:** Handles well with very high volumes of data

## *4.2 GitHub*

Twitter Fingers will be using GitHub for version control. There will be several files of code that will need to be organized so that our code does not get overwritten and we lose work. GitHub is the best option for version control, because many of us have experience using GitHub. Committing and pulling code will be easy, and can be done by using command line, the GitHub website, or the GitHub desktop application.

## *4.3 XAMPPS*

Our application will need a web server in order communicate with the Twitter Streaming API. We will use XAMPP as web server. XAMPP is a web server allowing script to be run in MySQL, PHP, and Perl. XAMPP can be downloaded for free and used with Windows, Mac OS X and Linux. Using our own locally hosted web server will make testing our product much easier, and will give us quick results when pulling information from the APIs. Instructions on how to run our program from the XAMPP web server will be provided in the readME for our product.

## *4.4 Text Editor*

We will be using a variety of text editors in order to code. Our PHP, JavaScript, HTML, and various other types of code will be written in text editors like Sublime Text and Notepad++.

# **5.0 Software Function Priorities**

The following software functions are ranked in terms of their priority and how important they are to the complete functioning of the project.

|  |  |  |
| --- | --- | --- |
| **Major Functional Tasks/Requirements** | | |
| ***Number*** | ***App Function*** | ***Priority*** |
| 1 | Connect to Twitter Streaming API | 1 |
| 2 | Displaying tweets in real-time | 1 |
| 3 | Storing Tweets in a NoSQL database | 1 |
| 4 | Organize display of tweets based on content | 2 |
| 5 | Start and stop stream of tweets | 2 |
| 6 | Manipulate list of Twitter accounts being streamed | 1 |
| 7 | Analyze tweets and provide statistical information | 2 |
| 8 | Create a visually appealing interface | 3 |

The priorities of each task range from “1” being of the highest priority, to “5” being the lowest priority.  The functional tasks listed above will ensure that the software application will adhere to our client’s requirements of this project.

# **6.0 Performance/Behavior Issues**

This is not applicable as of now since the application is not in its testing/implementation phase.

## *6.1 Management and Technical Constraints*

Main Constraints:

* + 1. Final Delivery Date scheduled for May 2nd, 2017

1. Users will not be able to access the database or manage it.
2. For the database, a NoSQL database has to be implemented.

# **7.0 Version History**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Document** |
| 1.0 | 2/16/2017 | Functional Specification Version 1 |
| 2.0 | 3/16/2017 | Functional Specification Version 2 |
| 3.0 | 4/6/2017 | Functional Specification Version 3 |
| 4.0 | 5/2/2017 | Functional Specification Version 4 |