**Twitter Fingers**

Design Specification

Julian Mathis (PM)

Dorian Wood

Anisha Carter

Taylor Clark

Dominique Collins

Sydney Parker



Contents

[1.0 Introduction 2](#_Toc481443427)

[1.1 Goals and Objectives 2](#_Toc481443428)

[1.2 Statement of Scope 2](#_Toc481443429)

[2.0 Product Design 3](#_Toc481443430)

[2.1 Overview 3](#_Toc481443431)

[2.2 User Interface 3](#_Toc481443432)

[2.2.1 Expected Input 4](#_Toc481443433)

[2.2.2 Expected Output 4](#_Toc481443434)

[2.3 Global Data Structure 4](#_Toc481443435)

[3.0 Architectural Design 4](#_Toc481443436)

[3.1 MongoDB (Database) 4](#_Toc481443437)

[3.2 Web Application (Server-Side) 5](#_Toc481443438)

[3.2.1 Web Application Architectural Design 6](#_Toc481443439)

[3.2.2 The Twitter Streaming API (RESTful API) 6](#_Toc481443440)

[3.2.3 The tmhOAuth Library 7](#_Toc481443441)

[3.3 Statistical Analysis Program (Rtools) 8](#_Toc481443442)

[5.0 Restrictions, Limitations, and Constraints 12](#_Toc481443443)

[6.0 Testing 13](#_Toc481443444)

[7.0 Version History 13](#_Toc481443445)

# **Introduction**

Twitter Fingers has been tasked with creating an application capable of procuring and analyzing posts found on Twitter which provide information about the financial status of companies and stocks. The software will provide the user with useful stock information from the top financial Twitter accounts and allow customization of display functions.

*1.1 Goals and Objectives*

The overall goal of this software application is to provide the user with useful financial information regarding company financial status and stock prices. In addition, the software will perform statistical analysis on all gathered data to provide the user with a more detailed understanding of the information. The program must also provide a level of customization for users to specify from which financial information is gathered.

*1.2 Statement of Scope*

The scope of this project is to create a website application that pulls Twitter posts from financially focused accounts, and displays them to the user in real-time as they are posted, and enters them into the database for future reference. Once entered the database the gathered information will be analyzed to provide more concise and relevant statistical information, as opposed to just raw financial data.

**2.0 Product Design**

## *2.1 Overview*

From the onset of development, the product will be designed with the requirements in mind. Design of the product will consist of two main areas: The database, and the web-application. The database will be designed in a way to facilitate storage and query of large amounts of data from real-time tweets. The web-application will be designed to allow users to easily retrieve and display financial tweets, as well as statistical information.

## *2.2 User Interface*

The user interfaces with the product via a web-application. The user will utilize a combination of text input and buttons in order to perform operations. Throughout development of the product, ease of use will be a factor in determining the layout and presentation of the user-interface and its functions.

The user interface of the product is realized through the web-application. The web-application contains all functions available to the user which allow manipulation of the product. Such functions include: communicating with the API (“The Twitter Streaming API”) by stopping or starting the Twitter feed, modification of search parameters, display of tweets in real-time from Twitter accounts, and display of statistical data gathered from tweets.

The interface for our web-application is in the form a website written in PHP, HTML, and CSS. This web interface will display the stream of tweets, along with the button to start and stop the stream. This interface will also allow the user to execute other features such as retrieving tweets from the database and performing the statistical analysis.

## *2.2.1 Expected Input*

At any time the product may receive the following input from the user:

*Start the Twitter Feed (button)*

*Stop the Twitter Feed (button)*

*Enter search keywords (text)*

## *2.2.2 Expected Output*

The user receives a continuously scrolling Twitter feed displaying stock market related Tweets, and Tweets containing any keywords they may have specified.

## *2.3 Global Data Structure*

The data structure that would best characterize this product is the JSON format. JSON is an object based description language, just like XML. The JSON format has been chosen based on its versatility as well as the fact that Java and PHP are supported. The Twitter streaming API will retrieve tweets in JSON format, display to the user, and ultimately permanently store them in the MongoDB database.

**3.0 Architectural Design**

*3.1 MongoDB (Database)*

Gathered tweets from chosen Twitter accounts will be transferred to the display feed, and subsequently to the database. Real-time procurement of tweets will undoubtedly yield a sufficient amount of data, but to provide more useful information to the user, statistical analysis will be performed. The database will serve as a means of storing the large amounts of data to be analyzed, after which it will be displayed.

We will be using a NoSQL database to store the tweets we have displayed to the user. MongoDB is a document style database, so our information will be stored as a markup language. We look to store the tweets/information as JSON format. This will allow us to reference our data explicitly as each piece of information will have a unique ID.

Using a NoSQL database is effective for streaming tweets in real-time because they are scalable. As we look to the future, it is important to have a scalable database for this project, as there are millions of tweets tweeted every day, and for storage and analysis of these tweets, it is important that our database is able to grow.

*3.2 Web Application (Server-Side)*

The web application will be the only client-side component of the product. The web-application will serve as the interface between the user and the various background components of the product; these components include the scripts responsible for the web application and its functions, The Twitter Streaming API, the database and its input/output files, and the statistical program. At this stage of development, the application will be designed using the following languages: HTML, JavaScript, PHP. The web application will be simple in nature and organized so that all functions are easily accessible useable.

### *3.2.1 Web Application Architectural Design*

The web application will contain the following components:

*Start Twitter feed (button): Starts the scrolling Tweets*

*Stop Twitter feed (button): Stops the scrolling Tweets*

*Search bar: The user inputs text to specify keywords to search on Twitter*

*Scrolling Twitter feed: The visual report showing queries tweets*

The operation of starting the feed, gathering tweets, automatically starts a process of connecting to the MongoDB database, and storing gathered Tweets (in the form of JSON data) to MongoDB Collections. Storing data to the database makes it possible to perform statistical analysis and discern financial and stock related information. The following are the main classes utilized to perform this process:

**Class MongoClient**

*Function: A connection manager for PHP and MongoDB used to manage connections with MongoDB databases*

**Class MongoCollection**

*Function: A grouping of MongoDB documents*

### *3.2.2 The Twitter Streaming API (RESTful API)*

Connection to the Twitter Streaming API will be implemented using the PHP Scripting Language. The script file is located server-side, and its main function is to allow communication between our web-application and the Twitter Streaming API. Implementation of the script file is covered in further detail in the following section (3.2.2 The tmhOAuth Twitter Library).

### *3.2.3 The tmhOAuth Library*

The tmhOAuth Twitter Library provides several functions which allow communication to the Twitter Streaming API. The library is located within the public directory of our server (ex. XAMPP/htdocs folder). This library is available on GitHub, in the tmhOAuth repository by *themattharris*.

In order to make calls to the Twitter Streaming API authorization is required. Applications attempting to communicate with the Twitter Streaming API must obtain access keys and/or tokens. In our case, both the unique “consumer keys” and “user tokens” were provided along with the Twitter Developer Account. These unique keys and tokens, tied to each Twitter Developer Account, allow Twitter to verify who is trying to connect to the API, and authenticate the request. A screenshot of this script is below:



*A block of code showing how the script uses the tmhOAuth PHP library, and makes a request to stream tweets from the Twitter Streaming API.*

The tmhOAuth library packages are inherited using the PHP command **“require”**, as shown on line 10 above. Following this, on line 12 of the example, a new tmhOAuth object is instantiated, and all access keys and tokens discussed earlier are passed to this object. Once passed the tmhOAuth library does the bulk of the work required to stream tweets. The script will make calls to the streaming API, and retrieve tweet information in the JSON data format. This data will be translated and parsed in order to retrieve the exact tweets and information that is desired by the user.

*3.3 Statistical Analysis Program (Rtools)*

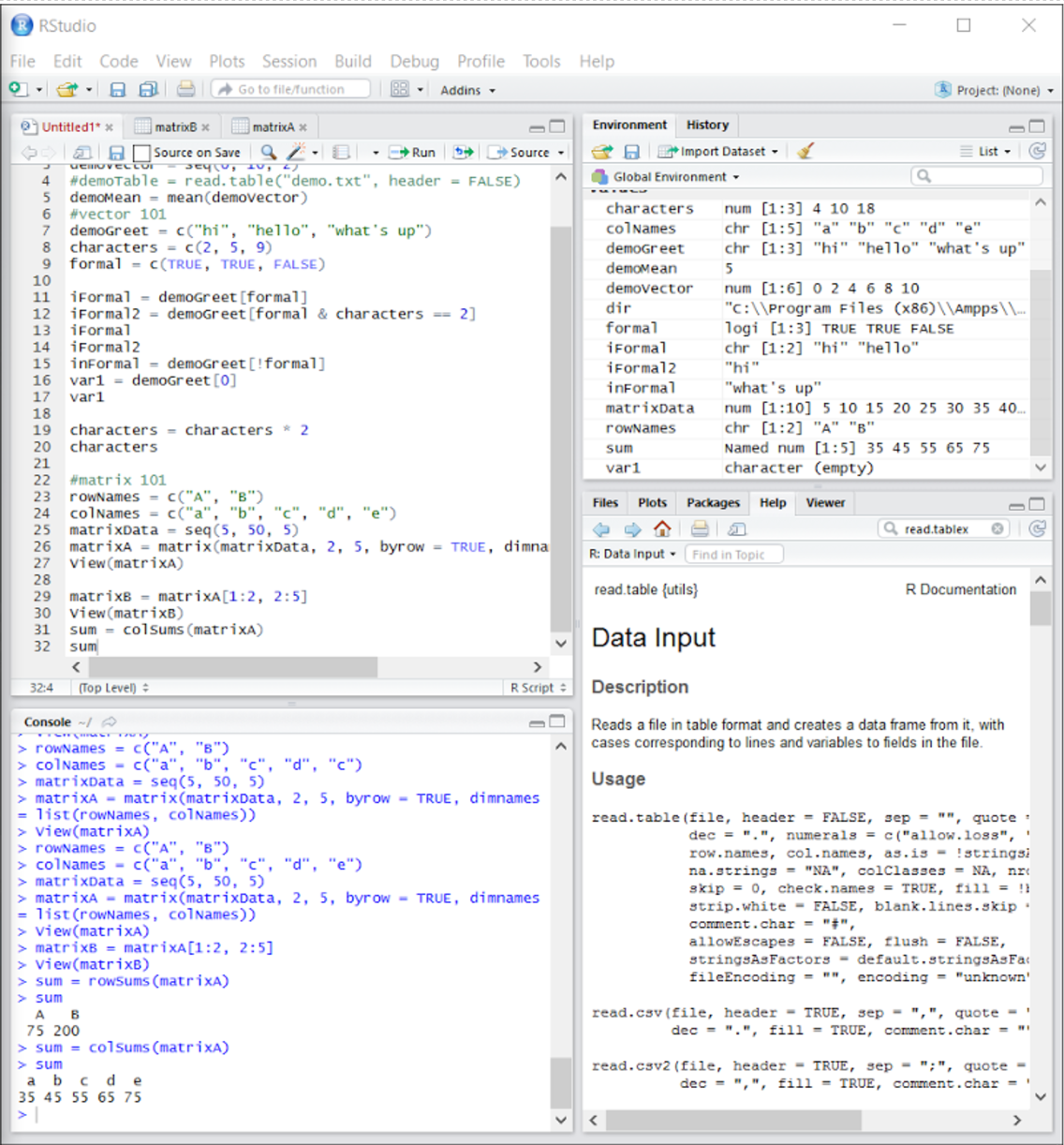
Our program will analyze the data (tweets) we pull from Twitter and provide statistical information to the user. This program will use R programming language. The R programming language is useful for providing statistics and graphical representations of data. Our program will provide useful statistics to the user in order to help them make informed decisions when investing and handling finances. Predetermined analytics will include:

1. Number of tweets per hour
   1. Identify timestamp of latest tweet
   2. Count number of tweets until timestamp is 1 hr less
      1. Pseudocode: while (currentTime > LatestTime - 1) {tweets++};
   3. Keep running average of counts per hour in vector.
2. Most frequent tweeter(s)
   * 1. Add all present tweeters in data to vector
   1. Pseudocode: while (count < all tweets) {increment count for tweeters in vector};
   2. Return max in vector
3. Most popular object of tweets (ex. SnapChat)
   1. Add all present tweets(hashtags) in data to vector
   2. Pseudocode: while (count < all tweets) {increment count for tweets in vector};
   3. Return max in vector
4. Frequency of tweeters per day or per week
   1. Keep running total of tweeters per day
   2. Return range and average of tweeters each day for the week

Some screenshots of our program that will perform these statistical analyses and the output are below:



*Image above shows Rtools and a demo provided by RGui to create a pie chart. R allows developers to produce visuals in HTML. This HTML code can be added to the website via a PHP script and users will have the capability to visualize data.*



*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.*

Below is an architectural design of how our application will be implemented:



*Architectural design and data flow of web-application, database, and corresponding components.* ***“Twitter Fingers Website”*** *encompasses all Client-side components.*

# **5.0 Restrictions, Limitations, and Constraints**

* This product has a strict release date of May 2nd, 2017.
* User-interaction of this product, and consequently graphical design, will be limited to the web-application only. All other functions are considered back-end, and will not require visual design.
* Data gathered from Twitter must be displayed in raw format before it is transferred to the database.
* Gathered financial data stored in the database must be translated from raw format, analyzed, and displayed to allow inference of useful information.
* Data gathered from Twitter that is relevant to stock, finance, and investing must be designated in an easily identifiable manner.

# **6.0 Testing**

Testing methods have been specified in the Test Specification document.

# **7.0 Version History**

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