Design Document

Jake Ballinger, Dibyo Mukherjee, Hawk Weisman, Gabe Kelly, Ian MacMillan

December 13, 2013

"...I have always found that plans are useless, but planning is indispensable."
- Dwight D. Eisenhower

1. Introduction to Document

1.1 Purpose of the Product

Knightingale is a *Twitter* analytics tool. Whether it's helping businesses better figure out to whom to market their products, aiding clinical psychologists understand the impact of social media on anxiety disorders, or making navigating *Twitter* easier for young adults, *Knightingale* is intended for all users.

The *Software Design Document* is intended principally for the development team and their professor at *Allegheny College*, *Dr. Gregory Kapfhammer*. This document provides a description of the functions of *Knightingale* at a low-level of detail.

1.2 Document Conventions

This document was written following *IEEE* conventions. It was formatted with LaTeX. This is the most low-level of the trio of documents; the architecture document is at a higher level of description, and the requirements document is more concerned with the high-level concepts and expectations of the procuct.

Names, classes, and methods will be italicized, and both numbers and variables are modified with the $ wrapper in LaTeX.

1.3 Scope of the Product

Knightingale is a *Twitter* analytics system. Refer to Section 1.4 of the Requirements Document.

1.4 References

1.5 Outline of the rest of the SRS

Section 2: System Overview
Section 3: Data Design
Section 3.1: Data Description
Section 3.2: Component Design
Section 4: Human Interface Design
Section 4.1: Overview of User Interface

2. System Overivew

Knightingale was, at least in theory, conceived by *Dr. Gregory Kapfhammer* at *Allegheny College*. He assigned the creation of the system to the students in his *Computer Science* 290 *Principles of Software Development* class as their final project. Four teams of five students would each develop their own *Twitter* analytics system. *Knightingale* is one of those systems.

What sets *Knightingale* apart from its competition is it's user-friendly interface. **More on this**.

To further develop an example given in Section 1.1, consider the situation of a clinical psychologist. It is common knowledge that a person's use of social media can be linked to their loneliness. The clinical psychologist might use the metrics provided by *Knightingale* to better understand her client.

3. Data Design

3.1 Data Description

The information domain of *Knightingale* is the twitter archive.*ZIP* file downloaded from *Twitter*. The zip file is parsed as an ArrayList of Tweets using the ZipParser and TweetBuilder classes that is then stored in a SQLite3 database. *Knightingale* uses a *SQLite* database to store all the information provided by *Twitter*. The database has two tables: one named `Tweets`, the other named `Users`. The `Tweets` table contains 10 columns and the `Users` table contains 2 columns. The `Tweets` tables is constructed in the same order as the information provided by *Twitter*, with matching column names. The `Users` table has *user_id* as its first column which gets populated with all replied and retweeted user, and *user_name* being the second column which is populated with a *Twitter4J* call to match user `IDs` to their *Twitter* profile name. Rows from the tweets table are usually extracted as ResultSets from the database and then converted to Tweets using the TweetBuilder class. ArrayLists of tweets are usually passed around in the various analysis methods.

3.2 Component Design

The major components of the system are as follows:

Package `Tweet` This package contains some fundamental blocks of the system:

`Tweet.java` : This class models a tweet as it is stored in the twitter archive zip file.

`TweetBuilder.java` : This class has methods to build tweets from `twitter4j Statuses`, `ResultSet`, and arrays of Strings.

`LogConfigurator.java` : This class sets up logging for the system.

4. Human Interface Design

4.1 Overview of User Interface

Describe the functional of the system from the users perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.