Social Media Mapper

Final Project Report

Prepared by Matthew Silverman and Rebecca Kasinger-Landstreet  
for CSCI 210, George Washington University

November 29th, 2010

**Table of Contents**

1 Introduction 1

1.1 Motivation 1

1.2 Problem Statement 1

2 Requirements Analysis 1

2.1 Product Scope 1

2.2 Overall Description 1

2.2.1 Product Perspective 1

2.2.2 Product Functions 2

2.2.3 User Classes and Characteristics 2

2.2.4 Operating Environment 2

2.2.5 Design and Implementation Constraints 3

2.2.6 Assumptions and Dependencies 3

2.3 External Interface Requirements 3

2.3.1 User Interfaces 3

2.3.2 Hardware Interfaces 3

2.3.3 Software Interfaces 3

2.4 System Features 4

2.4.1 Enter Search Criteria 4

2.4.2 Execute Search 4

2.4.3 List the Search Results 5

2.4.4 Display a Map 5

2.4.5 Option to Search Again 6

2.4.6 Option to Select Number of Results to Return 6

2.4.7 Option to Overlay Multiple Search Results 7

2.4.8 Option to Limit to only Posts Made in the U.S. 7

2.5 Other Nonfunctional Requirements 8

2.5.1 Performance Requirements 8

2.5.2 Safety Requirements 8

2.5.3 Security Requirements 8

3 Software Design 8

3.1 Design Goals 8

3.2 Architecture 8

3.2.1 Type of System 9

3.2.2 Platform 9

3.2.3 Inputs and Outputs 10

3.2.4 User Interface 10

3.2.5 Locality 10

3.2.6 Instances 10

3.3 Data 10

3.4 Communication 11

3.5 Code 11

3.5.1 Introduction 11

3.5.2 Modules 12

3.5.3 Results View 13

3.5.4 Interfaces 13

3.5.5 Installation 13

3.5.6 Upgrades 13

3.5.7 Uninstall 13

3.6 Development 14

4 Implementation Details 15

5 Test Cases 15

5.1 Testing SMM Search for Flickr 15

5.2 Testing SMM Search for Twitter 15

5.3 Testing SMM Search for Twitter and Flickr 16

5.4 Testing SMM Google Map 16

5.5 Testing SMM Search Validation 16

6 Project Plan 16

Appendix A: References 17

Appendix B: User Manual 17

# Introduction

## Motivation

As more social media communities, such as Flickr and Twitter, begin providing the capability for users to automatically tag their geographical location when posting to the site, it would be beneficial to provide a means to analyze this information, finding trends, locating hot-spots for certain topics, etc. The Social Media Mapper (SMM) goes a step further in that it allows user to analyze information posted to multiple social media sites. The product will allow a user to enter search criteria and specify which social media sites to search. It will then use that search criteria to find the most recent related posts on the selected social media sites, return links to the posts, and map the location where each post was made on a single map.

## Problem Statement

The Social Media Mapper will be an application that allows users to analyze the geographic locations of posts made to social media sites, providing search capability with results listed as links to the original post and a map of the location where each post was made.

# Requirements Analysis

## Product Scope

The first release of the Social Media Mapper (SMM) performs the following functions:

* Allow users to search one or more social media sites
* Allow the user to select which social media sites to search
* Return the most recent posts related to the entered search criteria
* Provide a link to each post returned by the search
* Map the location where each post was made
* Differentiate between the various social media sites on the map, i.e. blue for Twitter and red for Flickr

In future releases, SMM may allow the users the following additional functionality:

* Select how many search results to return from each social media site
* Overlay multiple search results, such as showing the location of “Redskins” posts in comparison to the location of “Cowboys” posts
* Limit search results to U.S. posts only

## Overall Description

### Product Perspective

The Social Media Mapper (SMM) was developed as a result of the increased popularity of tagging a location when making a post to various social media sites, such as Twitter and Flickr. Unlike the internal search engines of these sites, SMM will allow a user to simultaneously search multiple social media sites for content, and will provide a single map with the locations of the latest posts on each site.

### Product Functions

The first release of the Social Media Mapper (SMM) will performs the following functions:

* Allow users to search one or more social media sites
* Allow the user to select which social media sites to search
* Return the most recent posts related to the entered search criteria
* Provide a link to each post returned by the search
* Map the location where each post was made
* Differentiate between the various social media sites on the map, i.e. blue for Twitter and red for Flickr

In future releases, SMM may allow the users the following additional functionality:

* Select how many search results to return from each social media site
* Overlay multiple search results, such as showing the location of “Redskins” posts in comparison to the location of “Cowboys” posts
* Limit search results to U.S. posts only

### User Classes and Characteristics

The system users will include operators (end users) and the system administrators. The operators should be familiar with performing standard web searches, and should be familiar with the types of content included in Twitter and Flickr. The system does not require a high level of expertise from the end users. The interface will consist of a web page loaded in the user’s browser, containing input elements such as text boxes and buttons as well as a Google Map area and a list of search results. The system administrators are the application developers, Matthew Silverman and Rebecca Kasinger-Landstreet.

The number of allowed users will be determined based upon bandwidth to the application server. If the number of concurrent users exceeds load capacities, these users will experience performance degradation. However, the expected number of users is only 10, including the project team, course instructor, and friends and family who may assist with testing. With such a small number of expected users, we do not anticipate any capacity problems.

### Operating Environment

The software will operate as a client/server application. The user will act as the client and connect to the server from a PC or Mac computer using a modern Internet Browser such as Internet Explorer, Safari, Firefox or Chrome. The client’s browser must support JavaScript. The client must maintain a connection to the Internet. Matt Silverman’s home computer will act as the server, hosting the application using Apache Tomcat. Matt Silverman’s home computer will from here on be referred to as the application server. Queries to the various social media outlets will originate from the application server as opposed to the client host.

The application is intended to operate on a standard desktop or laptop PC with mouse and keyboard or equivalent devices. It is not intended to operate on mobile devices such as Blackberry, iPhone or iPad at this time.

### Design and Implementation Constraints

This project must be completed by November 29th, 2010. The application must be demonstrated to Professor Bellaachia in his office.

The application must only use free software and hosting services.

The number of queries made from the application to the various social media services will be limited by the query flood control limitations imposed by those services.

The application must be written in a language familiar to the developers.

The application requires a connection to the Internet.

### Assumptions and Dependencies

It is assumed that the application running on the application server will be able to make the necessary http connections to the various social media services.

## External Interface Requirements

### User Interfaces

The user interface will include the following elements

* Text box to capture search criteria
* Checkboxes to allow user to select which social media sites to search
* Search button
* List of links to the search results with short description (where available)
* Google map of locations
* Option to search again

### Hardware Interfaces

Both the user’s computer and the applicatioin will interface with a physical connection to the Internet through hardware network adapters.

The application will interface with the system hardware it runs on through the application server’s runtime environment. The runtime environment relies on the underlying operating system’s interface to the physical resources such as memory and network adapters. Matt will manage and maintain this interface using Apache Tomcat software.

### Software Interfaces

The user’s client (web browser) will interface with the application via the http protocol and render in its browser the HTML and JavaScript returned by the server.

The application will interface with various social media services through web requests to the service’s REST interface. The web requests will be formed using third party Java libraries imported into the application. The web requests will be formed following the syntax described in the API documentation of the libraries. The map displayed by the application will be built and customized using the Google Map API.Communications Interfaces

The user’s client (web browser) will interface with the application’s web server using an http connection on TCP port 80.

The application’s communications with the third-party social media services will be made using http web requests over TCP port 80.

## System Features

### Enter Search Criteria

#### Description and Priority

This feature will allow the user to customize his/her search by entering the search terms and select which social media sites to be searched. This feature is **MANDATORY**.

#### Stimulus/Response Sequences

This interface will be presented immediately to the user when he/she accesses the application, and will be presented again any time that the user selects the option to search again.

The user enters his/her search terms and selects the appropriate checkboxes to search the desired social media sites.

This functionality ends when the user presses the Search button.

#### Functional Requirements

This feature requires the following elements

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| REQ-1 | (Mandatory) Allow the user to enter his/her search terms. |
| REQ-2 | (Mandatory) Allow the user to select which social media sites to search. |
| REQ-3 | (Mandatory) Allow the user to execute the search. |

### Execute Search

#### Description and Priority

This feature allows the user to submit his/her search terms and social media site selections for the system to begin its search for related posts. This feature is **MANDATORY**.

#### Stimulus/Response Sequences

This feature will be executed when the user presses the Search button.

The system will then build and send the REST calls to each of the social media communities’ APIs.

When a response is received from each of the social media communities, the system will parse the various response formats, and aggregate them into a common format, limiting the results to the 20 most recent posts.

This functionality ends when SMM has parsed and aggregated the search results from the social media communities.

#### Functional Requirements

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| REQ-3 | (Mandatory) Allow the user to execute the search. |
| REQ-4 | (Mandatory) Properly format the REST calls and submit them to the selected social media sites. |
| REQ-5 | (Mandatory) Receive and parse the responses from each social media site. |
| REQ-6 | (Mandatory) Aggregate the responses from the multiple social media sites. |
| REQ-7 | (Mandatory) Default the results to the 20 most recent posts. |

### List the Search Results

#### Description and Priority

This feature allows the user to view a list of links (and descriptions, where available) of the posts from the social media sites that were returned by the search. This feature is **MANDATORY**.

#### Stimulus/Response Sequences

This feature will begin once the search results from the social media communities have been parsed and aggregated.

The system will use the information from the search results to build a list of links to the social media sites so that the user can view the post.

This functionality ends when the list of links is displayed to the user.

#### Functional Requirements

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| REQ-8 | (Mandatory) Display a list of links to the user that will allow them to view the posts on the social media sites. |

### Display a Map

#### Description and Priority

This feature allows the user to view a map of the locations where the posts were made. This feature is **MANDATORY**.

#### Stimulus/Response Sequences

This feature will begin once the search results from the social media communities have been parsed and aggregated.

The system will use the information from the search results to properly format the REST call to Google Maps, and will submit the REST call.

Once a response is received from Google, the system will display the mapped results on the Google map, differentiating between the various social media sites. For example, use blue points for Twitter and red for Flickr.

This functionality ends when the map has been presented to the user.

#### Functional Requirements

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| REQ-9 | (Mandatory) Properly format the REST call to Google Maps with the locations provided from the search results of the social media sites. |
| REQ-10 | (Mandatory) Display a Google Map of the post locations to the user. |
| REQ-11 | (Mandatory) Differentiate between the various social media sites on the map. |

### Option to Search Again

#### Description and Priority

This feature allows the user to repeat the search operation. This feature is **MANDATORY**.

#### Stimulus/Response Sequences

The input fields will be present on the screen to allow the user to search again.

If the user presses the search button again, the system will process the search.

This functionality ends when the user is viewing the new results.

#### Functional Requirements

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| REQ-12 | (Mandatory) Allow the user to enter a new search. |

### Option to Select Number of Results to Return

#### Description and Priority

This feature will allow the user to set the maximum number of results to display from each social media site. For example, display the 5 most recent posts on Flickr, and the 10 most recent posts to Twitter. This feature is ***optional*** and may not be included in the first release.

#### Stimulus/Response Sequences

An additional option may be added to the Search Criteria screen that allows the user to specify the maximum number of results to return from each of the social media communities included in the search.

If the user opts to use this functionality, he/she would provide the number of results to return from each social media site.

The system would return up to the specified number of related posts from each site.

This functionality ends when the proper maximum number of results is displayed to the user.

#### Functional Requirements

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| REQ-13 | (Optional) Allow the user to specify the maximum number of results to return from each social media site. |

### Option to Overlay Multiple Search Results

#### Description and Priority

This feature will allow the user to search multiple times, overlaying the results of each search. For example the user could search for “Redskins”, and then search for “Cowboys”, and the map would display the results of both searches. This feature would allow the user to compare the locations of each set of search results. This feature is ***optional*** and may not be included in the first release.

#### Stimulus/Response Sequences

An additional option may be added to the results display to “Add another Search”.

If selected, the user would be directed to another search screen, and the search process would be repeated for the new search criteria and search options.

The results of the second search would be plotted on the same Google Map. The system would allow the user to compare the locations of the posts about the different topics by differentiating the results in some manner, i.e. using different color schemes.

This functionality ends when the search results for multiple searches are displayed on the same map, and the search results are distinguished from each other.

#### Functional Requirements

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| REQ-14 | (Optional) Allow the user to plot the results of multiple searches on the same map. |

### Option to Limit to only Posts Made in the U.S.

#### Description and Priority

This feature would provide another option on the search screen to allow the user to filter the search results to only include posts from within the U.S. This feature is ***optional*** and may not be included in the first release.

#### Stimulus/Response Sequences

An additional option may be added to the search criteria screen to “Limit to U.S. Posts”.

If selected, the search results would be limited to only the posts that were made in the U.S., and the map displayed would only be a U.S. map.

This functionality ends when the search results are limited to only posts made from the U.S.

#### Functional Requirements

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| REQ-15 | (Optional) Allow the user to limit search results to only posts made from the U.S. |

## Other Nonfunctional Requirements

### Performance Requirements

The Social Media Mapper (SMM) performance requirement is simply to display the search results in minimal time to the user. While SMM cannot control the search time required by the social media community sites to process the requests, it will output the results on the list and map within 5 seconds of receiving the results from the social media sites.

### Safety Requirements

The Social Media Mapper (SMM) does not pose a safety risk to the user, and has no safety requirements.

### Security Requirements

There are no security requirements for this system. All information will be maintained solely by the social media community sites. No information will be saved on the Social Media Mapper (SMM) itself.

# Software Design

## Design Goals

The design of the Social Media Mapper should align with the innovative nature of the application realm which it evolved from. Forward thinking, the application should be extensible and provide a framework for future addition of features and functionality. The application must be designed in such a way that it meets the requirements laid forth in the Project Requirements Document. It must be designed such that it meets the constraints outlined in that document including use of freely available development tools and hosting services, and completion by the project deadline.

## Architecture

### Type of System

The Social Media Mapper will be an N-tier application. Components of the system will include the user-side client, application server, 3rd party REST services and map visualization provider. These parts will be referred to loosely as the client component, server components, and the 3rd party services.

### Platform

The client component will run in a web browser on the user’s computer.

The server components will be run by Matt Silverman’s home computer using Apache Tomcat, an open source servlet container.

The 3rd party services are run on various social media website platforms.

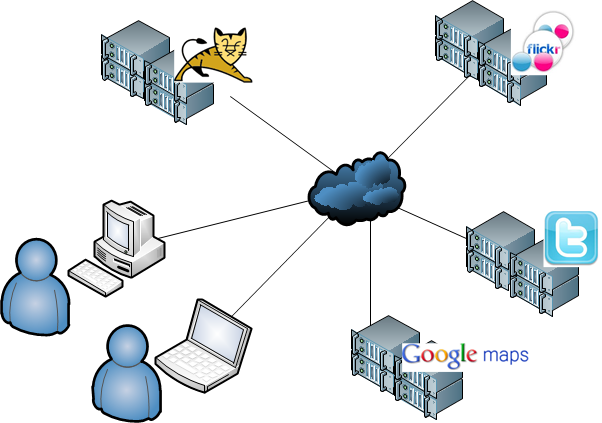


Figure 1 Physical layout

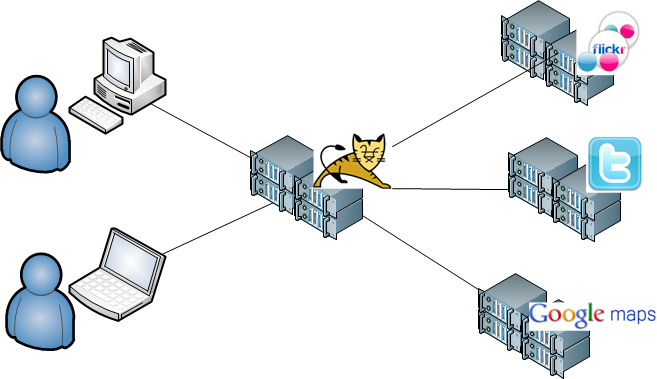


Figure 2 Logical Layout

### Inputs and Outputs

The major inputs to the system include the user’s search term and search result data from the 3rd party services. The major outputs from the system are the web requests containing the user’s search term submitted to the 3rd party services and the visualization of search results on a map presented to the user.

### User Interface

The user interface will be in the form of a web page loaded in the user’s browser, consisting of input elements such as text boxes and buttons as well as a Google map area.

### Locality

The three tiers described above will each reside on different systems. The 3rd party services will be hosted at their respective data centers. Twitter’s data center is in Salt Lake City, Utah. Flickr is located in San Francisco, California. The application server is located in Bethesda Maryland. The client may be located anywhere worldwide with access to the internet.

### Instances

There will be one instance of the server component deployed to Apache Tomcat. Although the application will be hosted on a home computer, it will support the anticipated level of usage for the course of the semester.

## Data

At this time there are no requirements for data persistence.

## Communication

The user’s client (web browser) will interface with the SMM’s web server using an http connection on TCP port 80. The user’s web browser will initiate the connection to the web server. The web server will serve a web page to the user. The user will submit information through forms on the web page back to the server.

The SMM’s communications with the 3rd party social media services will be made using http web requests over TCP port 80. If authentication and secure communication is required to access a service TCP port 443 will be used. The SMM will initiate connections to the 3rd party services and submit query information. The 3rd party services will return query results.

Finally, the SMM will return a results page to the user’s web browser and close the communication session.

## Code

### Introduction

Several factors were evaluated in the formulation of the code design. The code has to be developed in a language familiar to the developers. The hosting platform must be compatible with such programming language. There should also be compatible software components available.

A driving factor in the decision process was the desire to use Google’s App Engine. This component provides development, deployment and hosting features that make it a desirable choice. GAE provides a wealth of documentation as well as a plug-in for the Eclipse IDE making development and deployment simple. There are two software development kits available for GAE: Java SDK and Python SDK. There are a number of software components available for both Python and Java, therefore the choice of which language to use ultimately came down to which language the developers felt more comfortable working in, and that language is Java. App Engine Java applications use the Java Servlet standard for interacting with the web server environment. An application's files, including compiled classes, JARs, static files and configuration files are arranged in a directory structure using the WAR standard layout for Java web applications.[[1]](#footnote-1)

Although the final implementation does not include Google’s App Engine (refer to the Implementation Details section for more information), Google’s App Engine continued to serve as a code repository for the developers to collaborate more easily on the coding.

Special considerations have been made for leveraging existing software components to reduce development time and increase reliability. Specifically, there are a number of Java API packages for social media applications. These packages allow the developer to create connections, create and perform queries, and receive and process responses in an object oriented manner. The design of this project will seek to utilize these existing packages and not attempt to create lower level web requests and network connections.

### Modules

#### Java Server Pages (JSP)

While we could output the HTML for our user interface directly from the Java servlet code, this would be difficult to maintain, as the HTML gets complicated. It's better to use a template system, with the user interface designed and implemented in separate files with placeholders and logic to insert data provided by the application. We'll use JavaServer Pages (JSPs) to implement the user interface for the social media mapper. JSPs are part of the servlet standard. Apache Tomcat compiles JSP files in the application's WAR automatically, as configured in the web.xml file and maps them to URL paths.

The JSP file uses forms for the input components. Javascript is used for processing events. For example, when the body of the page loads, a call is made to the function initMap() to display the blank google map. Also, when the seach button is clicked, or enter is pressed while the search text box is being edited, the function ajaxLoadMap() is called. Ajax techniques are used to submit the search criteria to the Java servlet, specifying the handleMapResponse() function is to be called when the response is received. This allows for a single page to be used for both the search and results. The javascript library JQuery is used for displaying a mask over the page while the search is being performed. This prevents the user from accessing the controls and also shows that “something is happening”. When the results are returned the mask is hidden and the user can once again interact with the page. JQuery also provides a notation for easily reference div components of the DOM.

#### Servlet Class- QueryServlet

The application will need a web form so the user can post their search term and a way to process that form. The HTML of the form will go into the JSP. The destination of the form will be a new URL, /query, to be handled by a servlet class, QueryServlet. QueryServlet will process the form, and then send the response to the response handling function of the JSP in the user's browser.

The QueryServlet will perform the main processing of the application. It takes the user input and generates the queries to the 3rd party services, processes the results and generates the map output to be returned to the user.

The QueryServlet will import the following libraries:

Flickr Java API- flickrj[[2]](#footnote-2) Flickrj is Java API which wraps the REST-based Flickr API

Twitter Java API- Twitter4J[[3]](#footnote-3) by Yusuke Yamamoto. An open-sourced and mavenized Java library for the Twitter API, released under the BSD license.

**Internal Functions:** doPost- The main functionality will take place in the doPost method. This method will take the user input and generate the queries to the 3rd party services, processes the results and generate the map output to be returned to the user.

searchFlickr- Function takes the query text string as input, submits the query to Flickr, parses the response and returns a Collection of GenericResult objects.

searchTwitter- Function takes the query text string as input, submits the query to Twitter, parses the response and returns a Collection of GenericResult objects.

buildMapResponsePage- Takes as input an ArrayList of GenericResult objects that holds all of the response objects from all of the services that were searched, reads in the file baseResponse.txt which contains a template for generating the Google Map and fills in the response code template with the results data.

#### Object Class – GenericResult.java

The GenericResult class is an object class that will hold an abstracted version of the results from various social media providers. The object contains fields that are mapped to data returned from the social media sites, such as User ID, Title, Text, Link, Image URL, Latitude, Longitude and Place Name. There is also a field for Service Type. This field is implemented as an enum. When extending the application to use a new social media service, a new entry is added to the enum.

### Results View

The main component of the results view is the Google Map displaying the mapped location of search results. The Google Maps API is a JavaScript component.[[4]](#footnote-4) The webpage will initially contain a blank Google Map. Code will be created by the query servlet in accordance with the Google Maps API and returned in the Ajax response. The browser will interpret the JavaScript results code loaded in the user’s browser and display the results map.

### Interfaces

The Social Media Mapper interfaces with several components. The SMM will follow the API documentation for each of these components which describes the interface protocol.

### Installation

The system administrators will deploy a WAR file on the Apache Tomcat server.

### Upgrades

Upgrades will be distributed to the end users by updating the application on Google’s App Engine, downloading the files from GAE, and redeploying to Apache Tomcat on Matt’s home computer. Authentication and authorization to the code repository will be managed by registering the project team members’ Google accounts with the project on Google’s App Engine. The developers will make and maintain backups of previous versions in order to rollback if necessary.

### Uninstall

Since this application is web-based, end users will have no need to uninstall it. When the semester has ended, the project team may decide to uninstall the application by removing it from the Apache Tomcat server, or disabling the Tomcat software on the hosting computer.

## Development

Development will be performed using the Eclipse IDE[[5]](#footnote-5).

Required components for development include:

* Java JDK version 6 [[6]](#footnote-6)
* Flickrj version 1.2[[7]](#footnote-9)
* Twitter4j version 2.1.7[[8]](#footnote-10)
* JQuery Javascript library
* Apache Tomcat

# Implementation Details

As development and testing progressed, it was necessary to make a major deviation from the original planned implementation. As outlined in the original requirements and software design documents, we intended to use Google’s App Engine to host our application. During unit testing, we began running into errors when attempting to retrieve information from the social media sites. Further investigation uncovered that the social media sites limit the number of requests that can be made from a single IP address. Our application usage did not approach this limit, but since Google shares a pool of IP addresses amongst the myriad of applications hosted, our requests combined with those of the other applications on the IP address exceeded the limits.

The workaround to the rate limiting situation was to host the application on a private web server. The number of queries generated from the private web server should not cause rate limiting to occur.

# Test Cases

Testing for the Social Media Mapper (SMM) will involve executing searches against Flickr, Twitter, and SMM and then comparing the results.

To begin, select a search term (i.e. “Sears Tower” or “Washington Redskins”) to use as your search term for all of the test cases.

## Testing SMM Search for Flickr

This test case proves that SMM can search for Flickr posts, and that SMM’s Google Map markers are the Flickr icon.

1. Execute the search on SMM (<http://cs210-smm.info/smm/>), using the option to search only Flickr and to display the search results in a list and on the map.
2. The map markers should look like this:  
   C:\Users\Rebel\AppData\Local\Temp\video.png
3. In a new tab or window, execute the same search on Flickr (<http://www.flickr.com/>).
4. Compare the search results. They should be similar, but in most cases, they will not be identical.

NOTE: There will be differences, because not every Flickr user allows their location to be captured when they post. The SMM only retrieves results with a geographic location.

## Testing SMM Search for Twitter

This test case proves that SMM can search for Twitter posts, and that the SMM’s Google Map markers are the Twitter icon.

1. Execute the search on SMM (<http://cs210-smm.info/smm/>), using the option to search only Twitter, and to display the search results in a list and on the map.
2. The map markers should look like this:  
   C:\Users\Rebel\AppData\Local\Temp\text.png
3. In a new tab or window, execute the same search on Twitter (<http://twitter.com/>).
4. Compare the search results. They should be similar, but in most cases, they will not be identical.

NOTE: There will be differences in the search results, because not every Twitter user allows Twitter to capture their location when they post. The SMM only retrieves results with a geographic location.

## Testing SMM Search for Twitter and Flickr

This test case proves that SMM can search for both Twitter and Flickr posts at the same time.

1. Open a new tab or window and execute the search on SMM, using the option to search both Twitter and Flickr.
2. Compare the search results to the previous SMM searches. The results should be similar, but may not contain all of the results from both searches, depending upon the number of results retrieved by each search.

## Testing SMM Google Map

This test case proves that the markers are placed correctly on SMM’s Google Map.

1. Execute the search on SMM (<http://cs210-smm.info/smm/>), using the option to search both Twitter and Flickr, and to display the results on the map as well as in the list.
2. Copy and paste some of the latitudes and longitudes from the search results list.
3. In a new tab or window, search for the coordinates on Google Maps (<http://maps.google.com/maps?hl=en&tab=wl>).
4. Compare the mapped locations on Google against the map displayed by SMM.

## Testing SMM Search Validation

In order for SMM to submit a search to the social media sites, the user must enter a search term, and must select at least one social media site to search.

1. Click the “Search” button on SMM without entering any search terms.   
   An error should be displayed reminding you to enter a search term.
2. Enter a search term and deselect all social media sites.   
   An error should be displayed reminding you to select at least one social media site to search.

# Project Plan

The following table outlines the milestones for the project, timelines for meeting these milestones, and identifies a lead for the milestones. In some cases, it does not make sense to assign only one lead. These milestones are assigned to “All”.

| **Date** | **Milestone** | **Lead** |
| --- | --- | --- |
| September 27, 2010 | Submit one page abstract | Silverman |
| October 11, 2010 | Submit requirements document | Kasinger-Landstreet |
| October 25, 2010 | Submit design document, timeline, and tasks distribution | Silverman |
| Begin development of initial prototype (interface with social media sites) | Silverman |
| Begin development of initial prototype (interface to Google Maps) | Kasinger-Landstreet |
| November 8, 2010 | Deploy initial prototypes | Silverman |
| Continue development | All |
| Begin development of test plan | Kasinger-Landstreet |
| November 20, 2010 | Code freeze and testing | All |
| Bug fixing and regression testing | Silverman |
| November 29, 2010 | Submit and demo project | All |

# Appendix A: References

The format for the requirements analysis section is modified from Karl E. Wiegers’s format, © 1999, with granted permission to use, modify, and distribute, found online at: [www.csc.villanova.edu/~tway/courses/csc4181/srs\_template-1.doc](http://www.csc.villanova.edu/~tway/courses/csc4181/srs_template-1.doc)

The format for this software design section is modified from Ofer Faigan’s template, © 2005, with granted permission to use, modify, and distribute, found online at: <http://www.bitformation.com/art/sample_sw_design_doc.html>

# Appendix B: User Manual

1. [↑](#footnote-ref-1)
2. <http://flickrj.sourceforge.net/> [↑](#footnote-ref-2)
3. <http://twitter4j.org/> [↑](#footnote-ref-3)
4. <http://code.google.com/apis/maps/documentation/javascript/> [↑](#footnote-ref-4)
5. <http://www.eclipse.org/downloads/> [↑](#footnote-ref-5)
6. <http://www.oracle.com/technetwork/java/javase/downloads/index.html> [↑](#footnote-ref-6)
7. <http://sourceforge.net/projects/flickrj/> [↑](#footnote-ref-9)
8. <http://twitter4j.org/en/index.html#download> [↑](#footnote-ref-10)