Massmine for the Masses

Project Team: Morgan McIntyre, Logan Hornbuckle, Patricia Tanzer, Joshua Moore

Table of Contents

1. Project Definition

1.1 Why

Massmine (www.massmine.org) is a command-line tool useful in the data analysis field in that it can pull massive amounts of data from sources like Twitter and give them to the user for further analysis. However, because it is a command-line tool in the Linux operating system, not every user understands how to work with it. Additionally, it produces that data in a form which is difficult to parse for the non-technically oriented. This makes massmine very difficult to work with, despite its usefulness.

1.2 What

The goal of this project it to simplify this process for the average user by creating a web application that allows a user to create queries for Twitter data in a simple, easily understandable user interface. The data will also be presented in a simple user interface to allow for basic analysis without the user ever having to directly interact with it.

1.3 How

The web application will take query information from the user such as search terms and time periods, and send this information to a massmine instance running on the host server. The data retrieved from this massmine instance will be parsed and stored in a database which can then be accessed at a later date through the web application.

2. Project Requirements

2.1 Functional

- Web application and associated databases to store user information, store collected Twitter data, create and update new users, and allow users to gather and analyze Twitter data.

2.2 Usability

2.2.1 User interface

- Login interface: should allow the creation of accounts, as well as signing in. Optional but useful: ‘forgot password’.
- User account interface: Allow user to enter or change their Twitter authentication code. Optional but useful: ‘change password’.
- Study interface: Lists the studies that a user has access to and allows them to open Analysis interfaces on those studies. Another link allows them to create a new study or remove an old one.

Optional but useful: Allow renaming of studies and inviting/removing additional users from viewing a study.

- Collection interface: Allow the user to create queries or ‘studies’.
- Analysis interface: Allow the user to run basic analyses, such as aggregate by hour, on a study. (Optional: allow user to export analysis for presentation, etc).

2.2.2 Performance

- The user should not notice any significant time delay when running analyses on data already gathered. Depending on the query, data likely will not be available immediately after a query is given, but this should not impact the user from signing out, back in, or running additional queries and analyses in the meanwhile. A collection database will keep past queries to maintain efficient analysis.

2.3 System

2.3.1 Hardware

- A host Linux server with undefined storage resources in the UNCG library system.

2.3.2 Software

- Web application, likely displayed in Javascript and HTML.
- Additional modules in Python or similar languages.

2.3.3 Database

- MongoDB

2.4 Security

- At-rest encryption of user authentication keys, usernames, and Twitter data. Passwords used to sign in to the account will be salted and hashed. The web application will be accessed through an https:// page.
- The framework used, Django, has built-in protections and security features. Django guards against cross site scripting, cross site request forgery, SQL injection, and clickjacking.
- Additional measures to be taken include input validation on client and server side

3. Project Specification

3.1 Focus

- Develop an open source, web-based application to collect and analyze social network data using MassMine.

3.2 Domain

- The intended user for this application would be academic researchers and universities that may host their own version of this project. This application should appeal to those interested in social media research that lack the necessary command-line skills to fully utilize massmine.

3.3 Area

- Big data/data analysis - This project will involve large sets of metadata that must be parsed through and organized efficiently.
- Database management - There will be several databases used and maintained within the scope of this project.
- Web application/web application security

3.4 Libraries

- The following are examples of Python libraries that will be used: shell_plus, clean_pyc, json encoding and decoding, create_template tags, describe_form, notes

3.5 Development Environment

- Pycharm: A dedicated python and django IDE.
- Linux host (virtual machine)

3.6 Framework

- Django, a high-level Python web framework. This framework is scalable and secure.
- Mongodb is our framework for the databases. It is a NoSQL based database system that works well with pymongo for scripting.

3.7 Platform

- Web application meant for desktop access. This platform allows for user usage on any operating system, although the server should be running a Linux distribution to run the Massmine tool.

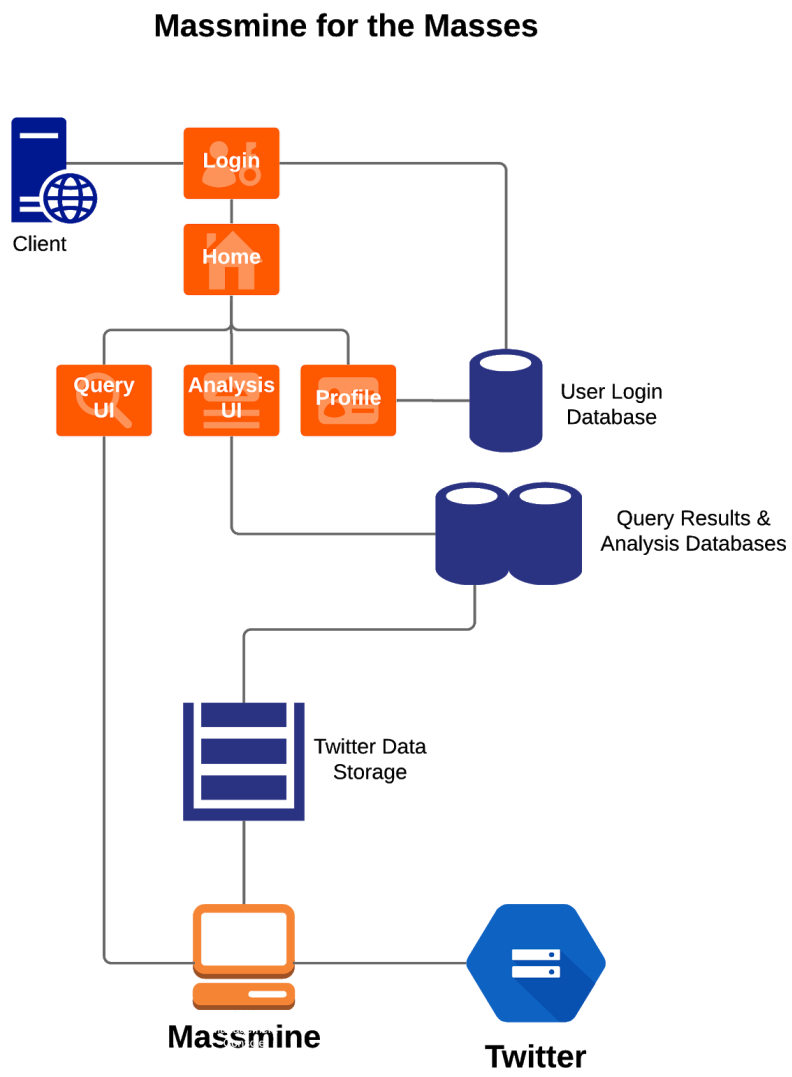
3.8 Genre

- Web application layered over the command-line tool Massmine.

4. System – Design Perspective

4.1 Identify subsystems – design point of view

- Overall:



- Database management - ER model
  - Design
  - Populating
  - Retrieving
- Analysis - use-case
- Massmine integration - sequence diagram
- Web interface - GUI

4.2 Sub-System Communication (Diagram and Description)

- Controls

- I/O
- DataFlow

4.3 Entity Relationship Model (E-R Model)

- Example - https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model

4.4 Overall operation - System Model

- Simplified Sub-system to System interaction

5. System – Analysis Perspective

5.1 Identify subsystems – analysis point of view

5.2 System (Tables and Description)

5.2.1 Data analysis

- Data dictionary (Table - Name, Data Type, Description)
-

5.2.2 Process models

5.3 Algorithm Analysis

5.3.1 Big - O analysis of overall System and Sub-Systems

6. Project Scrum Report

6.1 Product Backlog (Table / Diagram)

6.2 Sprint Backlog (Table / Diagram)

6.3 Burndown Chart

7. Subsystems

7.1 Subsystem 1 – Name 1 - *Individual responsibility*

- Initial design and model
  - Illustrate with class, use-case, UML, sequence ..... diagrams (eg if a UI, do a use-case)
  - Design choices
- Data dictionary
- If refined (changed over the course of project)

    - Reason for refinement (Pro versus Con)
    - Changes from initial model
    - Refined model analysis
    - Refined design (Diagram and Description)
- Scrum Backlog (Product and Sprint - Link to Section 6)
- Coding
    - Approach (Functional, OOP)
    - Language
- User training
    - Training / User manual (needed for final report)
- Testing

7.2 Subsystem 2 – Name 2 - *Individual responsibility*

- Initial design and model
    - Illustrate with class, use-case, UML, sequence ..... diagrams
    - Design choices
- Data dictionary
- If refined (changed over the course of project)
    - Reason for refinement (Pro versus Con)
    - Changes from initial model
    - Refined model analysis
    - Refined design (Diagram and Description)
- Scrum Backlog (Product and Sprint - Link to Section 6)
- Coding
    - Approach (Functional, OOP)
    - Language
- User training
    - Training / User manual (needed for final report)
- Testing

7.3 Subsystem 3 – Name 3 - *Individual responsibility*

- Initial design and model
    - Illustrate with class, use-case, UML, sequence ..... diagrams
    - Design choices
- Data dictionary
- If refined (changed over the course of project)
    - Reason for refinement (Pro versus Con)
    - Changes from initial model

    - Refined model analysish
    - Refined design (Diagram and Description)
- Scrum Backlog (Product and Sprint - Link to Section 6)
- Coding
    - Approach (Functional, OOP)
    - Language
- User training
    - Training / User manual (needed for final report)
- Testing

7.4 Subsystem 4 – Name 4 - *Individual responsibility*

- Initial design and model
    - Illustrate with class, use-case, UML, sequence ..... diagrams
    - Design choices
- Data dictionary
- If refined (changed over the course of project)
    - Reason for refinement (Pro versus Con)
    - Changes from initial model
    - Refined model analysis
    - Refined design (Diagram and Description)
- Scrum Backlog (Product and Sprint - Link to Section 6)
- Coding
    - Approach (Functional, OOP)
    - Language
- User training
    - Training / User manual (needed for final report)
- Testing

8. Complete System

- Final software/hardware product
- Source code and user manual – screenshots as needed - Technical report
    - Github Link
- Evaluation by client and instructor
- Team Member Descriptions