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

2.1.1Web application and associated databases to:

i. Store user information (username, password, 0 to many Twitter authorization keys)

ii. Store collected Twitter data (tweet id, author, content, etc.)

iii. Create and update new users (change password,

iv. Allow users to gather Twitter data. The mass of data collected by a query is a ‘study’.

v. Allow users to perform basic analysis on this Twitter data.

2.2 Usability

2.2.1 User interface

vi. Login interface: should allow the creation of accounts, as well as signing in. Optional but useful: ‘forgot password’.

vii. User account interface: Allow user to enter or change their Twitter authentication code. Optional but useful: ‘change password’.

viii. 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.

ix. Collection interface: Allow the user to create queries or ‘studies’.

x. 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

xi. 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.

xii. Multiple uses should be able to use the web-app at once. Up to 50 accounts expected.

2.3 System

2.3.1 Hardware

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

2.3.2 Software

xiv. Web application, likely displayed in Javascript and HTML.

xv. Additional modules in Python or similar languages.

2.3.3 Database

xvi. MongoDB

2.4 Security

b. At-rest encryption of user authentication keys and usernames. Passwords used to sign in to the account will be salted and hashed. The web application will be accessed through an https:// page.

c. 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.

d. 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.
- Virtual environments for testing
- 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

- Database management
  - User database:
    - Name: user_database
    - Collections:
      - user_collection
      - oauth_collection
      - study_collection
    - Accessed by: login and user management page
  - Tweet database:
    - Name: twitter_db
    - Collections:
      - tweet_info
    - Accessed by: Query page, analysis page
  - Analysis Database
    - Name: analysis_database
    - Collections:
      - Analysis_collection
    - Accessed by: Analysis page

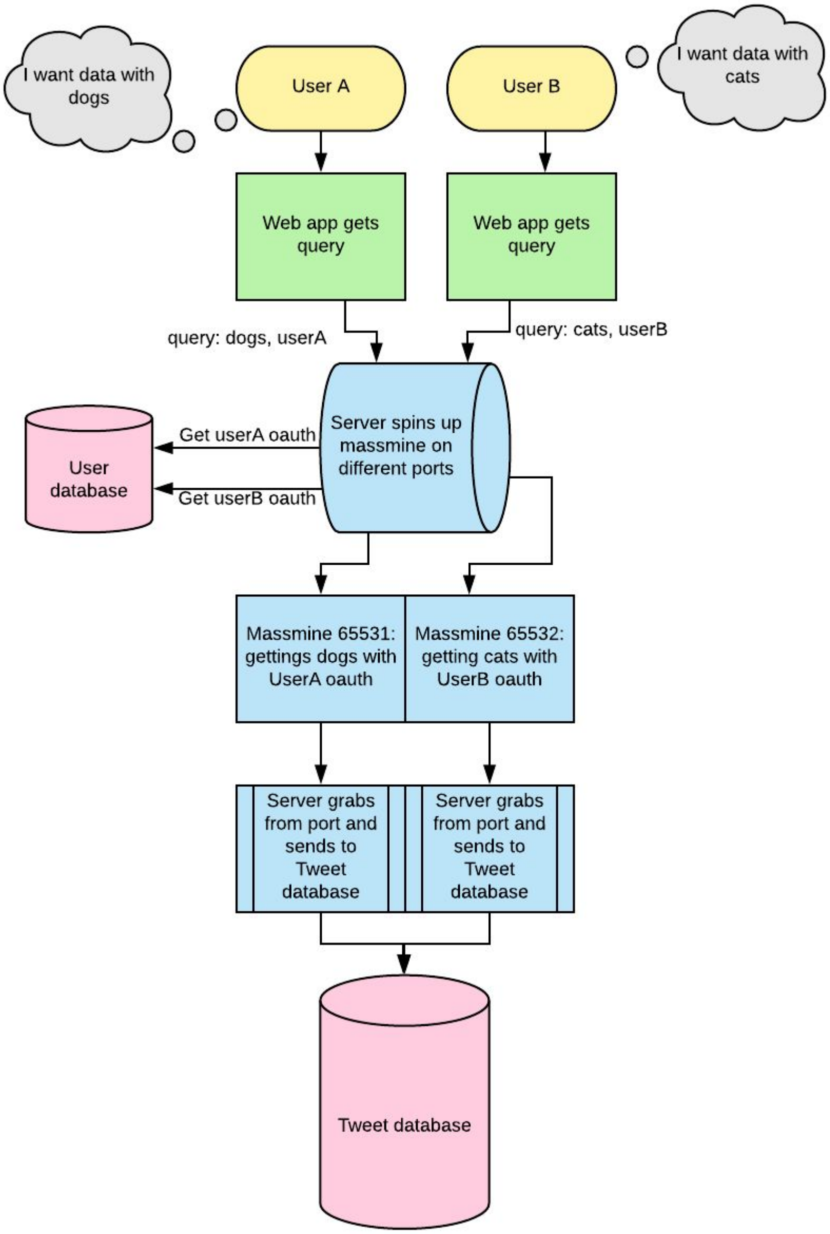
- Analysis:
  - Analysis types:
    - tweet_type - This sorts a study by tweet type (tweet, reply, retweet)
    - freq_words - Sorts by the most frequently used words in a study
    - freq_hashtags - Sorts by the most frequently used hashtags in a study
    - act_authors - Sorts by the most active users (by how many posts they have)
    - pop_authors - Sorts by authors with the most followers
    - men_accounts - Sorts by most mentioned accounts
    - lang - Lets user filter/sort languages
    - device - Sorts by source device
    - date - Sorts by date

- times_retweeted - Sorts by how many times each tweet was retweeted (most popular tweets)
- times_favorited - Sorts by how many times the tweet is favorited
- location - Sorts by country, all tweets with no geo information will be excluded

○ Website calls this database and gets list of possibly analysis for user to choose from

○ User may choose one of these analyses and feed it desired information (such as a word/phrase, or dates to view), so that the results are graphically displayed

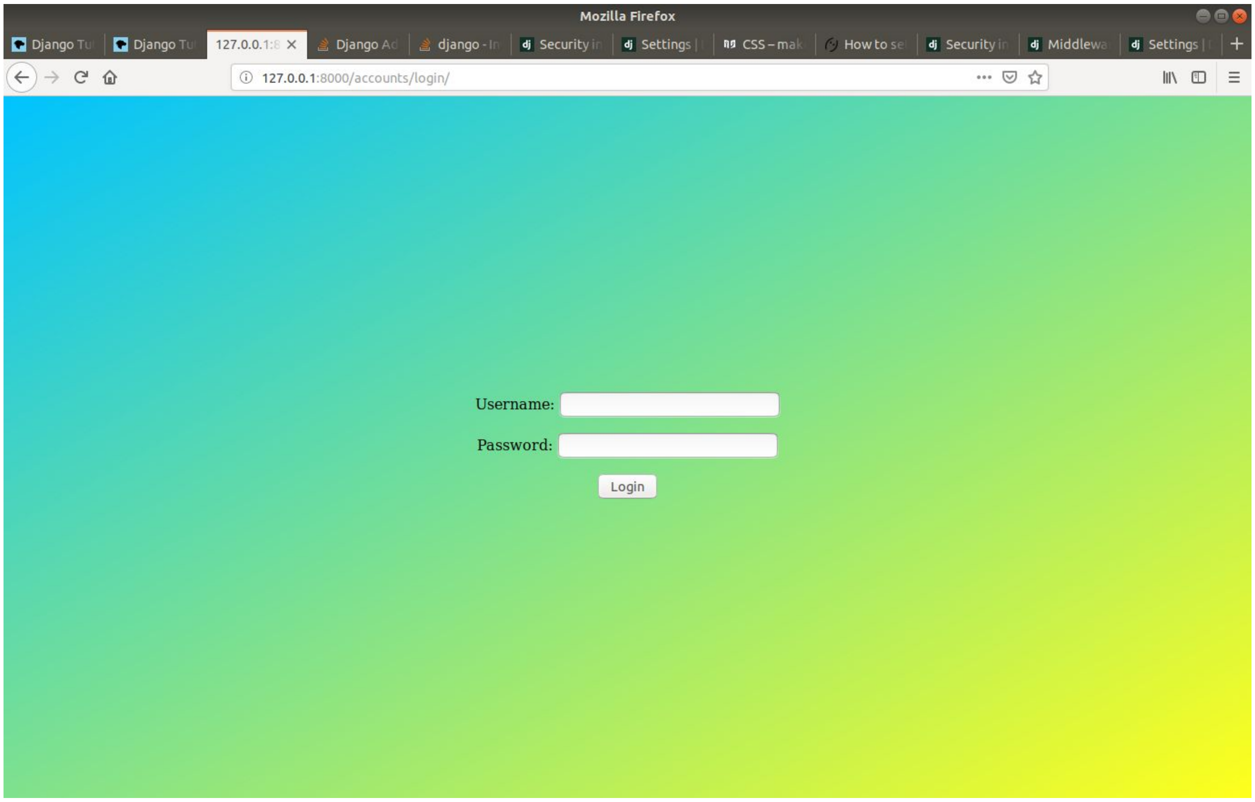
- Massmine integration:


I want data with dogs
User A
User B
I want data with cats
Web app gets query
Web app gets query
query: dogs, userA
query: cats, userB
User database
Get userA oauth
Get userB oauth
Server spins up massmine on different ports
Massmine 65531: gettings dogs with UserA oauth
Massmine 65532: getting cats with UserB oauth
Server grabs from port and sends to Tweet database
Server grabs from port and sends to Tweet database
Tweet database

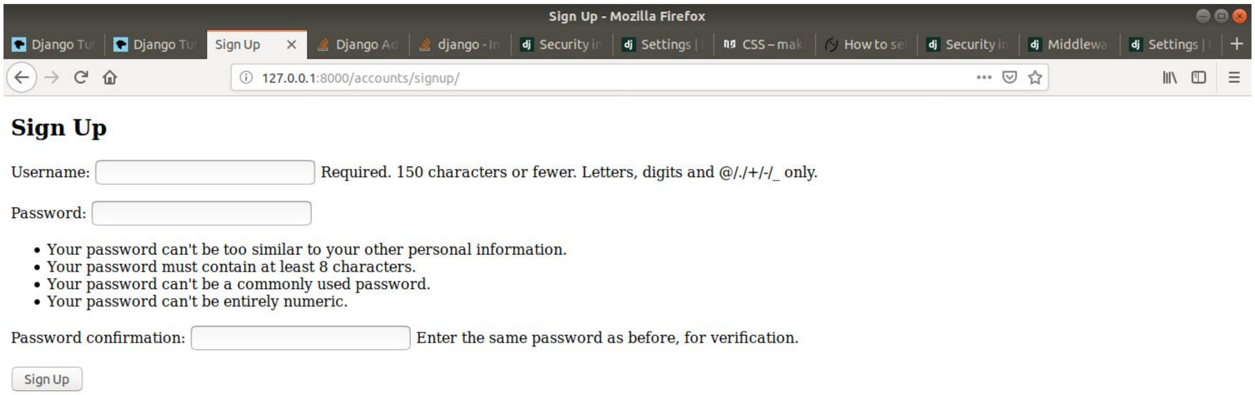
- ■

Web interface - GUI (Screenshots)

- GUI for user login



- GUI for account creation



Sign Up

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

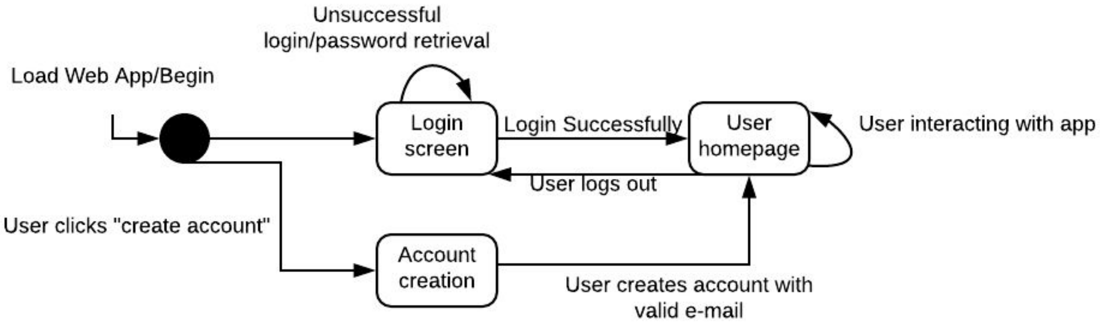
Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

4.2 Sub-System Communication

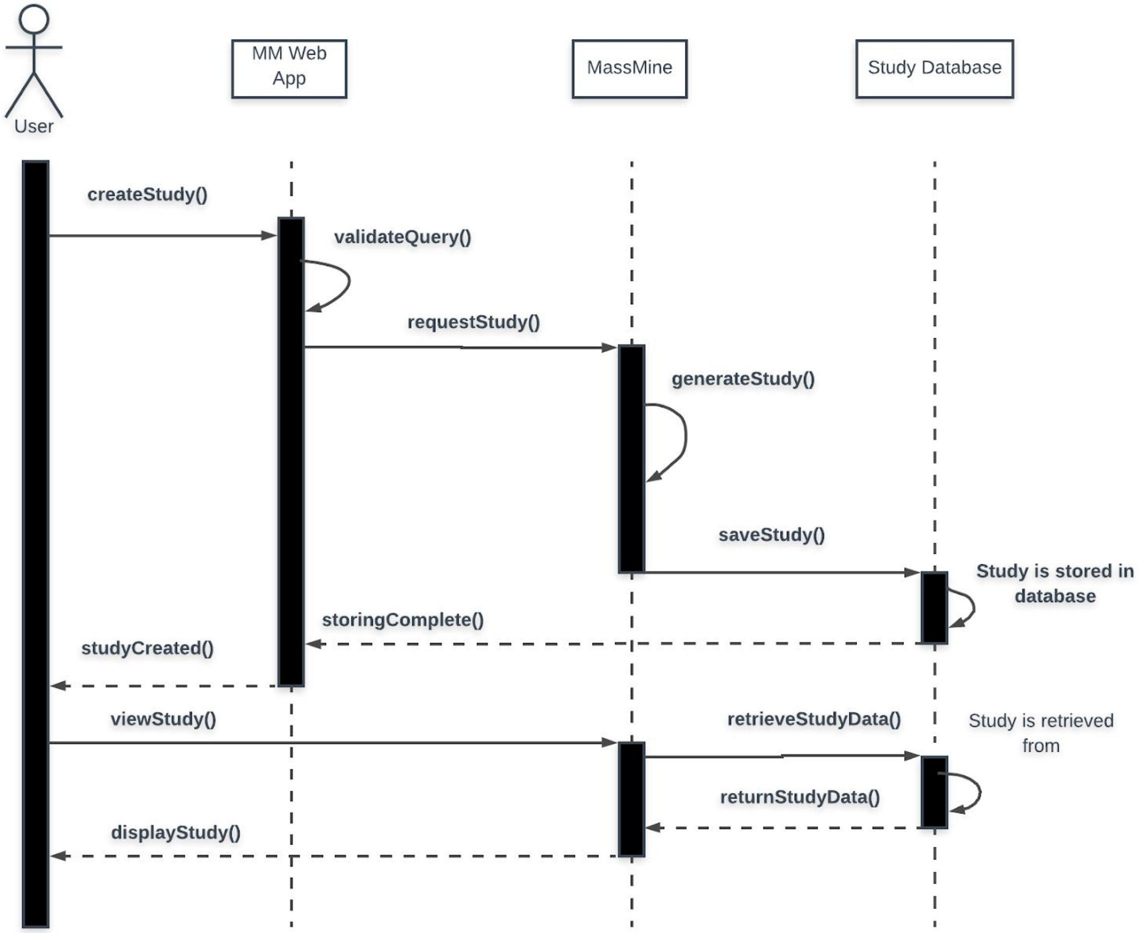
- Controls by subsystem
  - Subsystem 1: Database Management/Login and account creation
    - Most query terms will not allow the user to directly write into a box to avoid injection. For example, the date will be entered via an interactive calendar. Those fields which do allow the user to enter data (such as a search term) will be carefully checked for input validation
    - Login will require the correct username and password combination and verified in database.
    - User passwords will be hashed and salted in the user database.
  - Subsystem 2: Web GUI


Unsuccessful login/password retrieval
Load Web App/Begin
Login screen
Login Successfully
User homepage
User interacting with app
User logs out
User clicks "create account"
Account creation
User creates account with valid e-mail

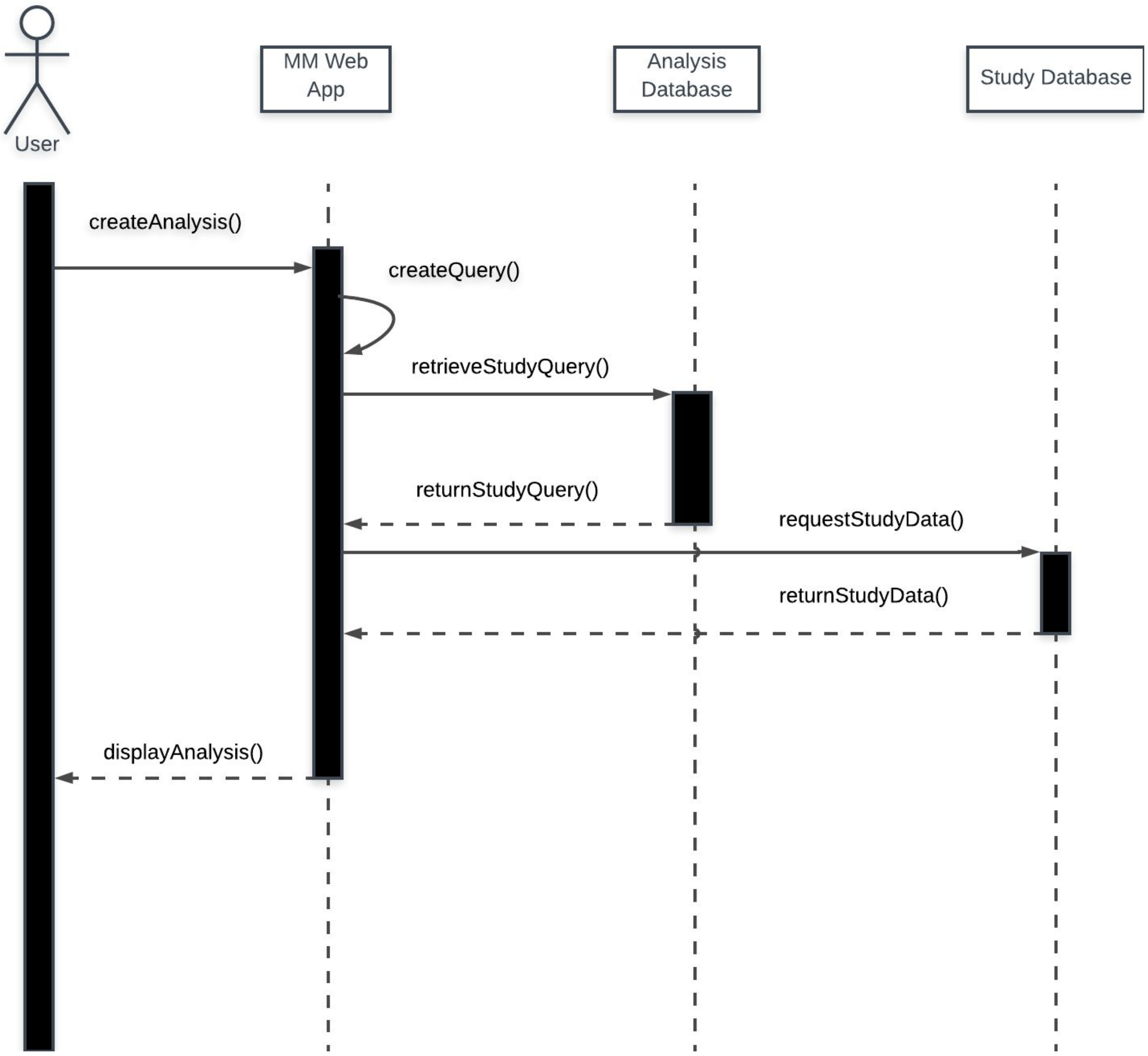
        - Django has built-in controls to prevent cross site scripting and cross site request forgery.
        - The user may navigate from page to page with buttons on the top of the screen.
        - If the user's login is incorrect (username or password) they will be temporarily denied access to their account.
        - Account creation will require a unique username and a password that fits length and complexity standards.
        - This subsystem will communicate with the Massmine integration section.
        - Each webpage verifies user credentials.
        - Django handles HTTPS requests and secure connections, but cannot serve HTTPS in production. In order for a production environment to support these security measures, a web server such as Apache must be implemented.
    - Subsystem 3: Analysis
        - The user may choose the type of analysis to be displayed from a drop-down menu.
        - This subsystem is the only one to call the analysis database.
        - Only uses tweets from user’s own studies.
    - Subsystem 4: Massmine Integration
        - Users need a Oauth keys to be able to gather twitter data. In the Massmine communication if they don’t have a valid Oauth key they can’t request a study.
        - The Oauth key will be encrypted for further security.
        - Massmine itself is a command line tool, and will be controlled directly by the server running the application.
- I/O by subsystem

- Inputs
    - Subsystem 1: Database Management
        - Username from user
        - Password from user
        - Oauth key from user
        - Randomly generated study ids
        - Twitter data from server
    - Subsystem 2: Web GUI/Study creation
        - Study requirements from user (date, search terms, etc.)
    - Subsystem 3: Analysis
        - Analysis requirements from user (choose from a drop-down menu)
    - Subsystem 4: Massmine communication
        - User information from web GUI and user database
        - User query
- Outputs
    - Subsystem 1: Database Management
        - Valid/Invalid response for authentication checking
        - Twitter data when requested by web page
    - Subsystem 2: Web GUI
        - Prompts user for study query (webpage)
    - Subsystem 3: Analysis
        - Prompts user for analysis query (webpage)
        - Analysis results to user as graphical display
    - Subsystem 4: Massmine communication
        - Results from Massmine to server and tweet storage

- Dataflow
    - Data-flow for creating a study

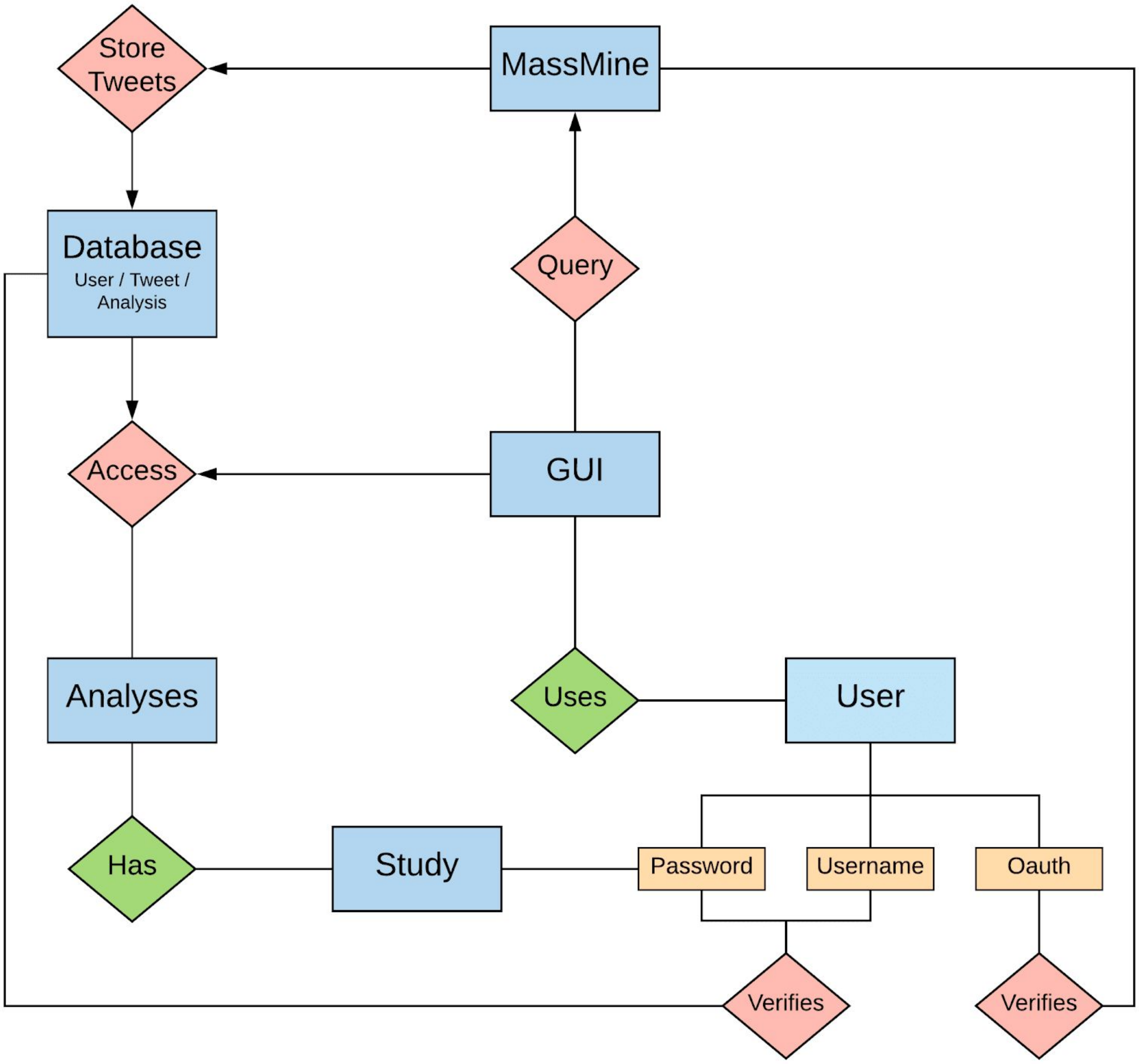


- Data-flow for creating an analysis

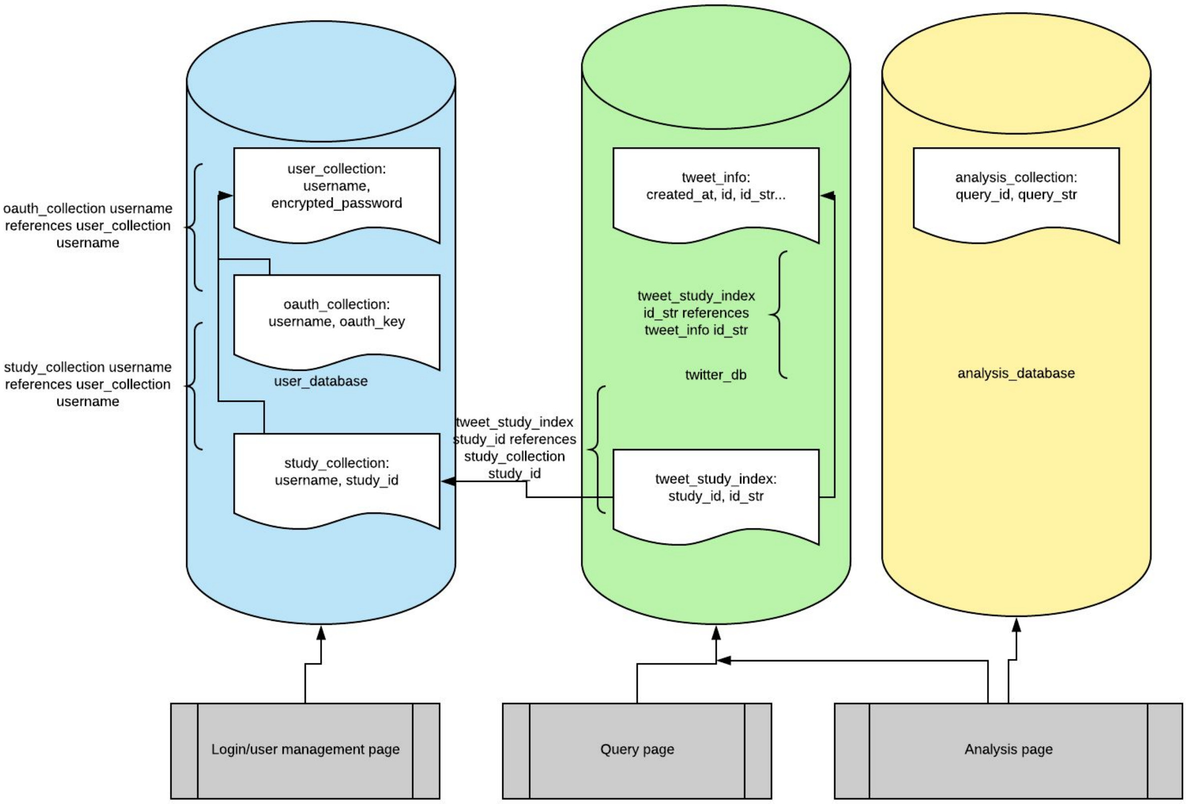


4.3 Entity Relationship Model

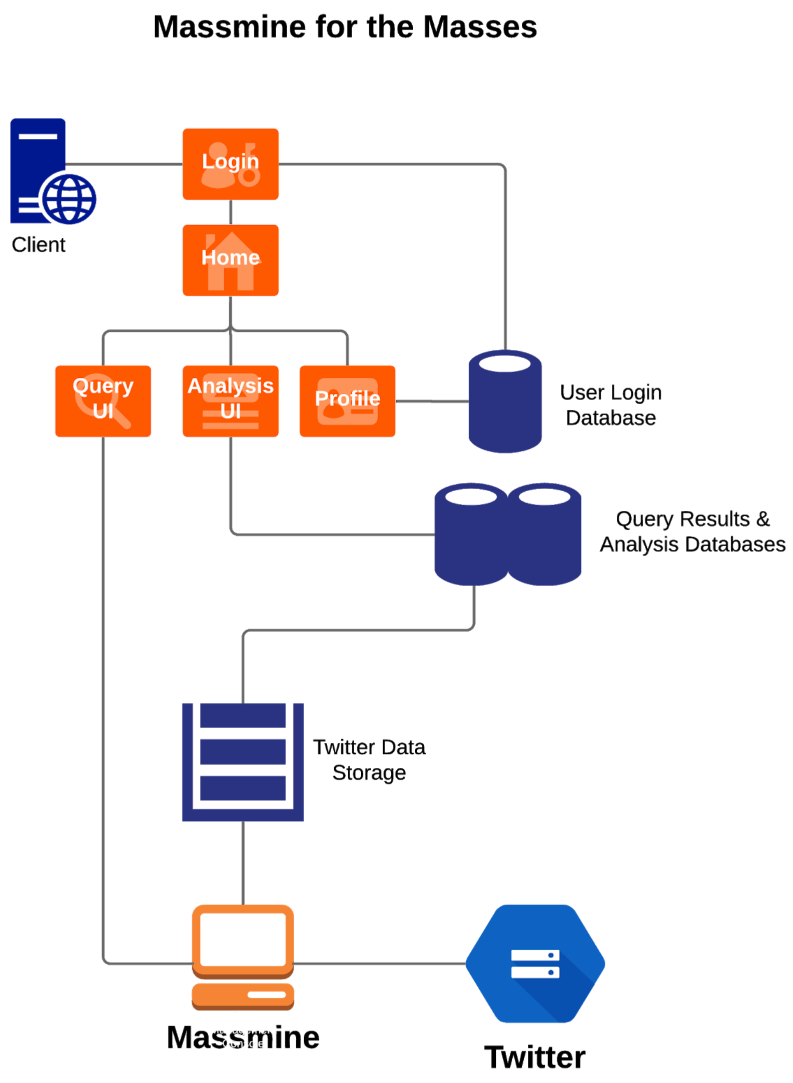
- System Wide ERM:



- Database ERM:



4.4 Overall operation - System Model



5. System – Analysis Perspective

5.1 Identify subsystems – analysis point of view

- Subsystem 1: Database Management
  - Data Storage:
    - Anticipating needing up to 100 TB for initial database storage.
  - Data Structure:
    - MongoDB allows various ways to use tree data structures to model large hierarchical or nested data relationships.

        - MongoDB collections do not require all documents to have the same schema.
        - MongoDB documents stored in JSON format.
    - Performance:
        - Allow up to 100 socket connections to insert into tweet database simultaneously.
        - Allow up to 25 users to store personal information in the user database, including up to 10 twitter authorization keys.
- Subsystem 2: Web GUI
    - GUI Structure:
        - Django-defined Models consisting of appropriate Field classes, specified by class attributes.
        - Web Server: Apache 2.4.38.
        - The only required part of a Django Model is the list of database fields it defines.
        - Web Interface should allow any open connection to interact with the login page, while internal pages require authentication.
    - Performance:
        - The server should support at least 25 simultaneous user sessions.
- Subsystem 3: Analysis
    - Analysis Structure:
        - Each analysis type is a filter that can be applied to a particular set of data
        - Multiple filters can be layered for more complex analyses
    - Performance
        - The analysis graphics should update as the analyses is working, so that users are not waiting long periods for data to display
- Subsystem 4: Massmine communication
    - Structure:
        - The server will initiate an instance of Massmine for each uncompleted user query, and close the instance when the query finishes.
        - The server should constantly monitor the massmine ports for output and stream that output to the tweet database
    - Performance:

        - The server should support 100 simultaneous Massmine instances (4 per user)

5.2 System (Tables and Description)

5.2.1 Data analysis

- Data dictionary:
- User database:
    - Name: user_database
    - Collections:
        - user_collection:

Attribute	Description
username	Name identifying a user
encrypted_password	Hashed and salted password used to confirm user login

        - oauth_collection:

Attribute	Description
username	References users_collection username
oauth_key	Twitter developer api key required for gathering tweet data

        - study_collection:

Attribute	Description
username	References username in user_collection
study_id	A user is associated with 0 or more 'studies', the collection of tweets gathered by a particular user

	query

        - Accessed by: login and user management page
    - Tweet database:
        - Name: twitter_db
        - Collections:
            - tweet_info: (all attributes except for study_id are the same as those in the tweet object found at https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object.html)

Attribute	**Description**
study_id	Identifies the user study to which the tweet belongs
created_at	UTC time when this Tweet was created. Example: "created_at":"Wed Aug 27 13:08:45 +0000 2008"

id	The integer representation of the unique identifier for this Tweet. This number is greater than 53 bits and some programming languages may have difficulty/silent defects in interpreting it. Using a signed 64 bit integer for storing this identifier is safe. Use id_str for fetching the identifier to stay on the safe side. See Twitter IDs, JSON and Snowflake . Example: "id":114749583439036416
id_str	The string representation of the unique identifier for this Tweet. Implementations should use this rather than the large integer in id. Example: "id_str":"114749583439036416"
text	The actual UTF-8 text of the status update. See twitter-text for details on what characters are currently considered valid. Example: "text":"Tweet Button, Follow Button, and Web Intents"
source	Utility used to post the Tweet, as an HTML-formatted string. Tweets from the Twitter website have a source value of web. Example: "source":"Twitter for Mac"

truncated	Indicates whether the value of the text parameter was truncated, for example, as a result of a retweet exceeding the original Tweet text length limit of 140 characters. Truncated text will end in ellipsis, like this ... Since Twitter now rejects long Tweets vs truncating them, the large majority of Tweets will have this set to false . Note that while native retweets may have their toplevel text property shortened, the original text will be available under the retweeted_status object and the truncated parameter will be set to the value of the original status (in most cases, false). Example: "truncated":true
in_reply_to_status_id	*Nullable.* If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's ID. Example: "in_reply_to_status_id":114749583439036416
in_reply_to_status_id_str	*Nullable.* If the represented Tweet is a reply, this field will contain the string representation of the original Tweet's ID. Example: "in_reply_to_status_id_str":"114749583439036416"
in_reply_to_user_id	*Nullable.* If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's author ID. This will not necessarily always be the user directly mentioned in the Tweet. Example: "in_reply_to_user_id":819797

in_reply_to_user_id_str	*Nullable.* If the represented Tweet is a reply, this field will contain the string representation of the original Tweet's author ID. This will not necessarily always be the user directly mentioned in the Tweet. Example: "in_reply_to_user_id_str":"819797"
in_reply_to_screen_name	*Nullable.* If the represented Tweet is a reply, this field will contain the screen name of the original Tweet's author. Example: "in_reply_to_screen_name":"twitterapi"

user	The user who posted this Tweet. See User data dictionary for complete list of attributes. Example highlighting select attributes:
	{
	"user": {
	"id": 2244994945,
	"id_str": "2244994945",
	"name": "TwitterDev",
	"screen_name": "TwitterDev",
	"location": "Internet",
	"url": "https://dev.twitter.com/",
	"description": "Your source for Twitter news",
	"verified": true,
	"followers_count": 477684,
	"friends_count": 1524,
	"listed_count": 1184,
	"favourites_count": 2151,
	"statuses_count": 3121,
	"created_at": "Sat Dec 14 04:35:55 +0000 2013",
	"utc_offset": null,
	"time_zone": null,
	"geo_enabled": true,
	"lang": "en",

	"profile_image_url_https": "https://pbs.twimg.com/" } }

coordinates	*Nullable.* Represents the geographic location of this Tweet as reported by the user or client application. The inner coordinates array is formatted as geoJSON (longitude first, then latitude). Example:
	"coordinates":
	{
	"coordinates":
	[
	-75.14310264,
	40.05701649
	],
	"type":"Point"
	}

place	*Nullable* When present, indicates that the tweet is associated (but not necessarily originating from) a Place . Example:
	"place":
	{
	"attributes":{},
	"bounding_box":
	{
	"coordinates":
	[[
	[-77.119759,38.791645],
	[-76.909393,38.791645],
	[-76.909393,38.995548],
	[-77.119759,38.995548]
	]],
	"type":"Polygon"
	},
	"country":"United States",
	"country_code":"US",
	"full_name":"Washington, DC",
	"id":"01fbe706f872cb32",
	"name":"Washington",
	"place_type":"city",
	"url":"http://api.twitter.com/1/geo/id/0172cb32.json"
	}

quoted_status_id	This field only surfaces when the Tweet is a quote Tweet. This field contains the integer value Tweet ID of the quoted Tweet. Example: "quoted_status_id":114749583439036416

quoted_status_id_str	This field only surfaces when the Tweet is a quote Tweet. This is the string representation Tweet ID of the quoted Tweet. Example: "quoted_status_id_str":"114749583439036416"
is_quote_status	Indicates whether this is a Quoted Tweet. Example: "is_quote_status":false
quoted_status	This field only surfaces when the Tweet is a quote Tweet. This attribute contains the Tweet object of the original Tweet that was quoted.
retweeted_status	Users can amplify the broadcast of Tweets authored by other users by retweeting . Retweets can be distinguished from typical Tweets by the existence of a retweeted_status attribute. This attribute contains a representation of the *original* Tweet that was retweeted. Note that retweets of retweets do not show representations of the intermediary retweet, but only the original Tweet. (Users can also unretweet a retweet they created by deleting their retweet.)
quote_count	*Nullable.* Indicates approximately how many times this Tweet has been quoted by Twitter users. Example: "quote_count":1138 Note: This object is only available with the Premium and Enterprise tier products.
reply_count	Number of times this Tweet has been replied to. Example: "reply_count":1585 Note: This object is only available with the Premium and Enterprise tier products.

retweet_count	Number of times this Tweet has been retweeted. Example: "retweet_count":1585
favorite_count	*Nullable.* Indicates approximately how many times this Tweet has been liked by Twitter users. Example: "favorite_count":1138

entities	Entities which have been parsed out of the text of the Tweet. Additionally see Entities in Twitter Objects . Example:
	"entities":
	{
	"hashtags":[],
	"urls":[],
	"user_mentions":[],
	"media":[],
	"symbols":[]
	"polls":[]
	}

extended_entities	When between one and four native photos or one video or one animated GIF are in Tweet, contains an array 'media' metadata. Additionally see Entities in Twitter Objects . Example:
	"entities":
	{
	"media":[]
	}

favorited	*Nullable.* Indicates whether this Tweet has been liked by the authenticating user. Example: "favorited":true

retweeted	Indicates whether this Tweet has been Retweeted by the authenticating user. Example: "retweeted":false
possibly_sensitive	*Nullable.* This field only surfaces when a Tweet contains a link. The meaning of the field doesn’t pertain to the Tweet content itself, but instead it is an indicator that the URL contained in the Tweet may contain content or media identified as sensitive content. Example: "possibly_sensitive":true
filter_level	Indicates the maximum value of the filter_level parameter which may be used and still stream this Tweet. So a value of medium will be streamed on none, low, and medium streams. Example: "filter_level": "medium"
lang	*Nullable.* When present, indicates a BCP 47 language identifier corresponding to the machine-detected language of the Tweet text, or und if no language could be detected. See more documentation HERE. Example: "lang": "en"

matching_rules	Present in *filtered* products such as Twitter Search and PowerTrack. Provides the *id* and *tag* associated with the rule that matched the Tweet. With PowerTrack, more than one rule can match a Tweet. See more documentation HERE. Example:
	"matching_rules": " [{
	"tag": "rain Tweets",
	"id": 831566737246023680,
	"id_str": "831566737246023680"
	}, {
	"tag": "snow Tweet",
	"id": 831567402366218240,
	"id_str": "831567402366218240"
	}]"

current_user_retweet	*Perspectival* Only surfaces on methods supporting the include_my_retweet parameter, when set to true. Details the Tweet ID of the user's own retweet (if existent) of this Tweet. Example:
	"current_user_retweet": {
	"id": 26815871309,
	"id_str": "26815871309"
	}

scopes	A set of key-value pairs indicating the intended contextual delivery of the containing Tweet. Currently used by Twitter's Promoted Products. Example: "scopes":{"followers":false}
withheld_copyright	When present and set to "true", it indicates that this piece of content has been withheld due to a DMCA complaint . Example: "withheld_copyright": true

withheld_in_countries	When present, indicates a list of uppercase two-letter country codes this content is withheld from. Twitter supports the following non-country values for this field: “XX” - Content is withheld in all countries “XY” - Content is withheld due to a DMCA request. Example: "withheld_in_countries": ["GR", "HK", "MY"]
withheld_scope	When present, indicates whether the content being withheld is the “status” or a “user.” Example: "withheld_scope": "status"
geo	**Deprecated.** *Nullable.* Use the coordinates field instead. This deprecated attribute has its coordinates formatted as *[lat, long]*, while all other Tweet geo is formatted as *[long, lat]*

- Accessed by: Query page, analysis page

- Analysis Database
    - Name: analysis_database
    - Collections:
        - analysis_collection

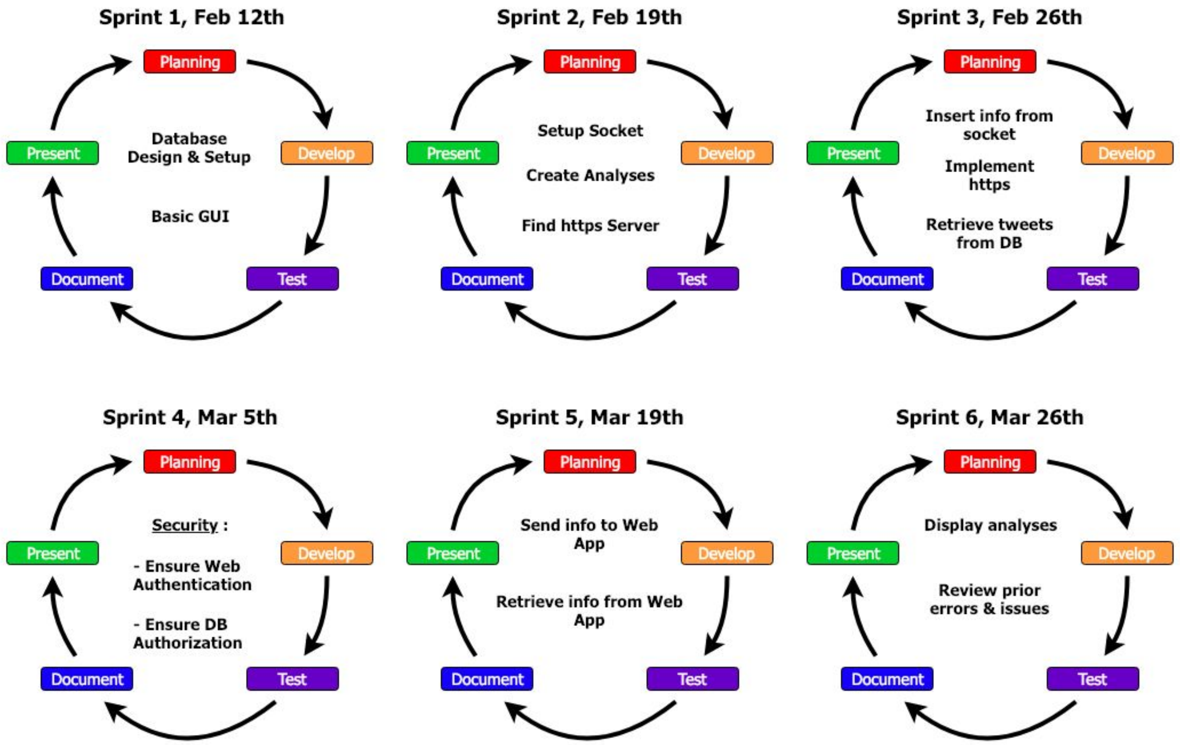
Attribute	**Description**
query_id	Identifies the particular query being used for analysis
query_str	The actual query being run on a data

	set

- ○ Accessed by: Analysis page

5.2.2 Process models

- System Process Model (Agile):



5.3 Algorithm Analysis

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

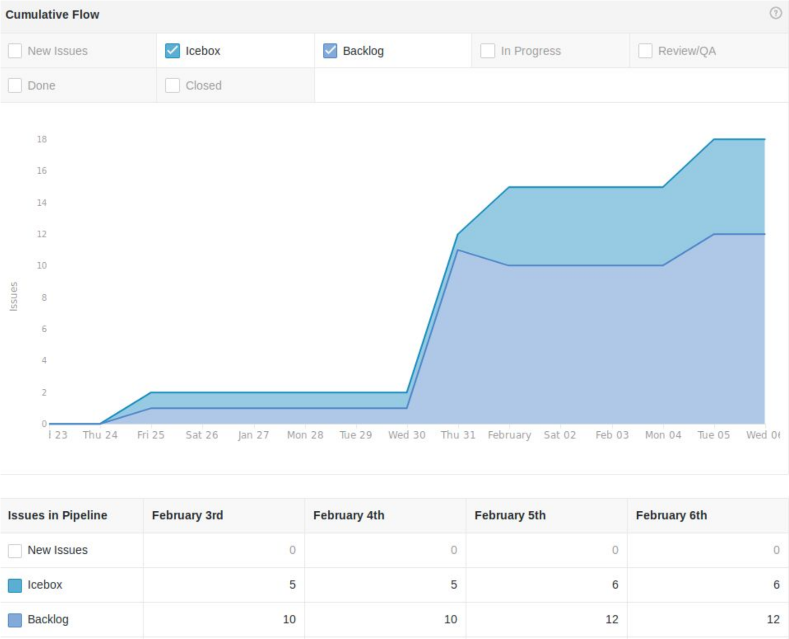
- Databases
    - ○ Updating/adding user: O(log n) - Looks first for username primary key. If username does not exist, adds entry
    - ○ Checking authentication: O(log n) - Looks for username primary key and checks hashed password
    - ○ Entering Tweet: O(1) - Assumes tweet-study pair does not exist already and enters a single entry to tweet_info and another to tweet_study_index
    - ○ Entering Tweet Group: O(n) - If there are n tweets in the Massmine output, there are n calls to the database
    - ○ Retrieving Tweet: O(n) - n is the number of tweets in the requested study

        - Must check for all tweets with a particular study id, then the attributes requested in that study
- GUI
    - Loading web page: O(1), since session cookie is validated once each time new page loads.
- Analysis Database
    - Analysis Retrieval: O(n), n is the number of fields/attributes in query.
        - Retrieving the query itself is O (log n)
- Big-O Massmine communication: O(n)
    - Sending the query to Massmine: O(1)
    - Retriving the data and inserting into the database: O(n)

6. Project Scrum Report

6.1 Product Backlog (Table / Diagram)

- Jan. 24 - Feb. 06

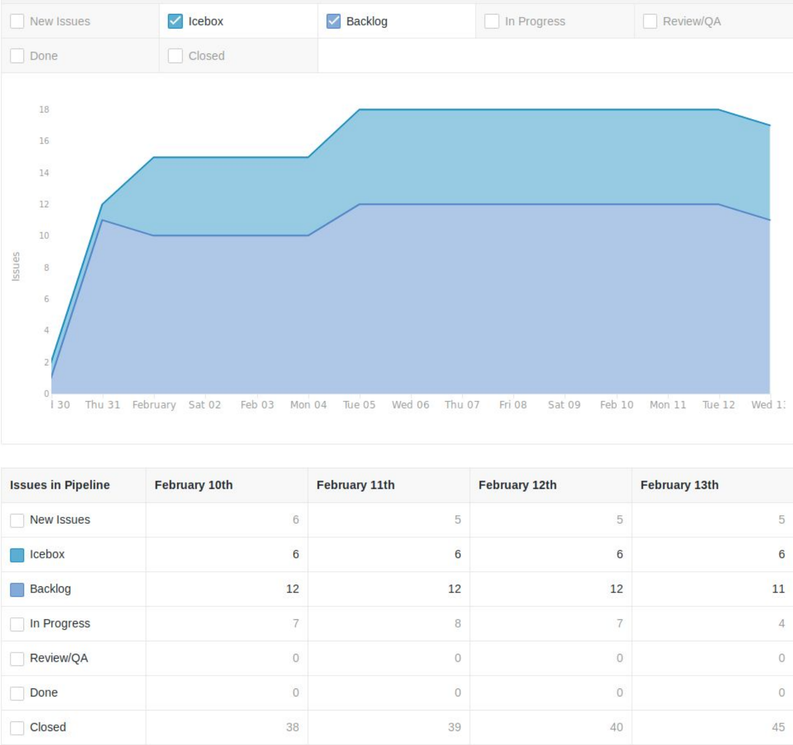


Cumulative Flow

□ New Issues ☑ Icebox ☑ Backlog □ In Progress □ Review/QA
□ Done □ Closed

Issues in Pipeline	February 3rd	February 4th	February 5th	February 6th
□ New Issues	0	0	0	0
🟦 Icebox	5	5	6	6
🟦 Backlog	10	10	12	12

- Jan. 30-Feb. 12



□ New Issues ☑ Icebox ☑ Backlog □ In Progress □ Review/QA

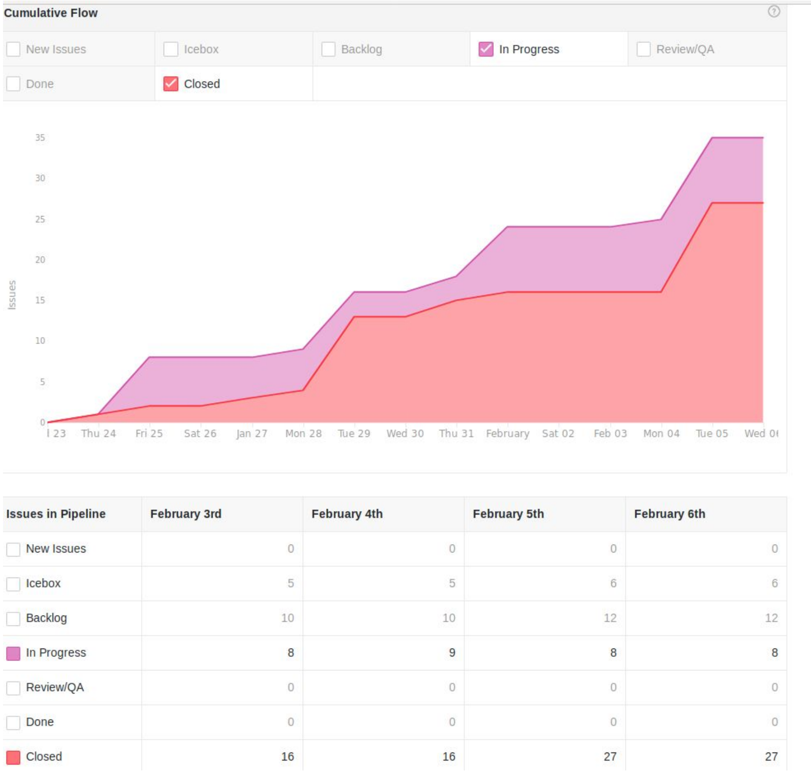
□ Done □ Closed

Issues in Pipeline	February 10th	February 11th	February 12th	February 13th
□ New Issues	6	5	5	5
🟦 Icebox	6	6	6	6
🟦 Backlog	12	12	12	11
□ In Progress	7	8	7	4
□ Review/QA	0	0	0	0
□ Done	0	0	0	0
□ Closed	38	39	40	45

-

6.2 Sprint Backlog (Table / Diagram)

- Jan. 24-Feb.06



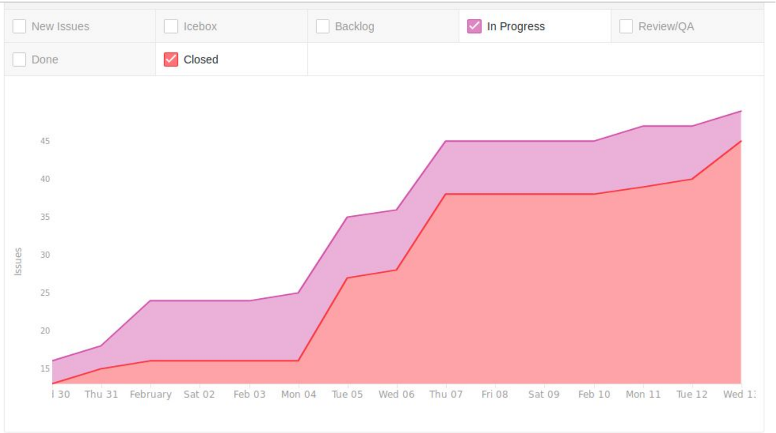
Cumulative Flow

□ New Issues □ Icebox □ Backlog ☑ In Progress □ Review/QA

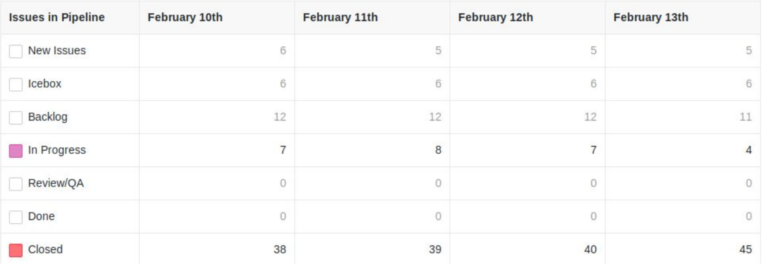
□ Done ☑ Closed

Issues in Pipeline	February 3rd	February 4th	February 5th	February 6th
□ New Issues	0	0	0	0
□ Icebox	5	5	6	6
□ Backlog	10	10	12	12
🟪 In Progress	8	9	8	8
□ Review/QA	0	0	0	0
□ Done	0	0	0	0
🟥 Closed	16	16	27	27

- Jan. 30-Feb.12



□ New Issues □ Icebox □ Backlog ☑ In Progress □ Review/QA
□ Done ☑ Closed

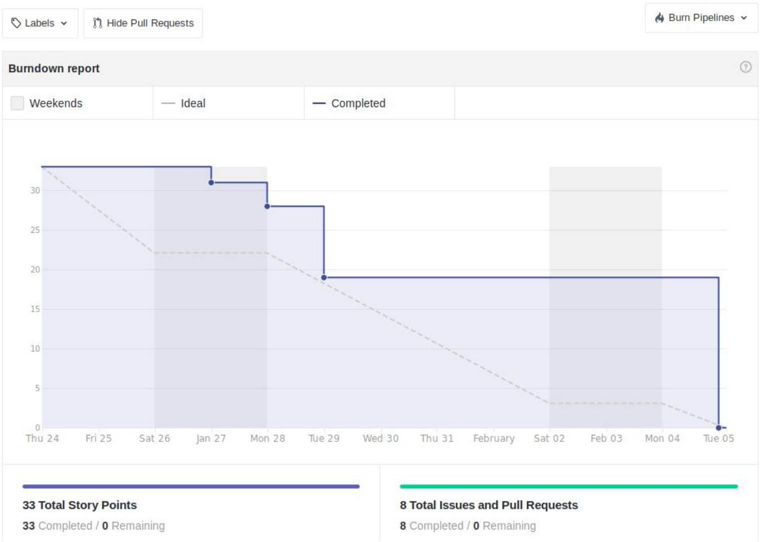


Issues in Pipeline	February 10th	February 11th	February 12th	February 13th
□ New Issues	6	5	5	5
□ Icebox	6	6	6	6
□ Backlog	12	12	12	11
🟪 In Progress	7	8	7	4
□ Review/QA	0	0	0	0
□ Done	0	0	0	0
🟥 Closed	38	39	40	45

-

6.3 Burndown Chart

- For Scrum Feb 5:

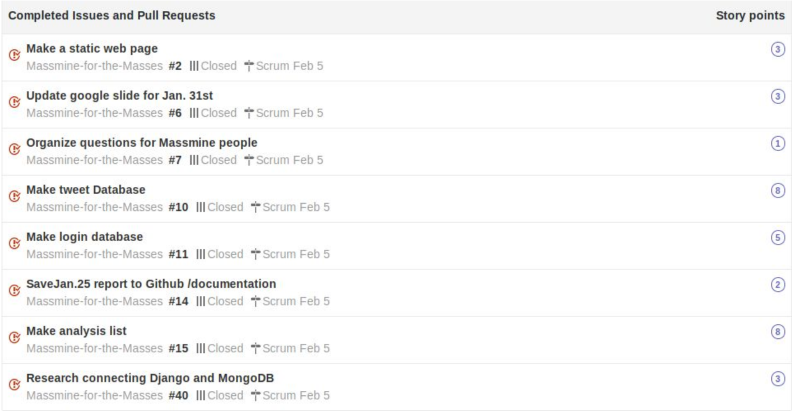


Burndown report

□ Weekends — Ideal — Completed

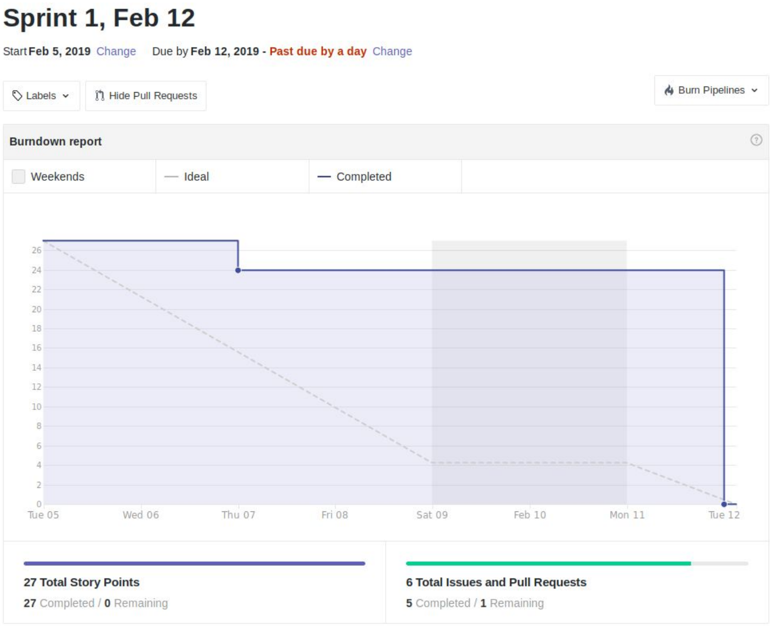
33 Total Story Points
33 Completed / 0 Remaining

8 Total Issues and Pull Requests
8 Completed / 0 Remaining



Completed Issues and Pull Requests	Story points
Make a static web page Massmine-for-the-Masses **#2** Closed Scrum Feb 5	3
Update google slide for Jan. 31st Massmine-for-the-Masses **#6** Closed Scrum Feb 5	3
Organize questions for Massmine people Massmine-for-the-Masses **#7** Closed Scrum Feb 5	1
Make tweet Database Massmine-for-the-Masses **#10** Closed Scrum Feb 5	8
Make login database Massmine-for-the-Masses **#11** Closed Scrum Feb 5	5
SaveJan.25 report to Github /documentation Massmine-for-the-Masses **#14** Closed Scrum Feb 5	2
Make analysis list Massmine-for-the-Masses **#15** Closed Scrum Feb 5	8
Research connecting Django and MongoDB Massmine-for-the-Masses **#40** Closed Scrum Feb 5	3

- For Scrum Feb 12:



Sprint 1, Feb 12

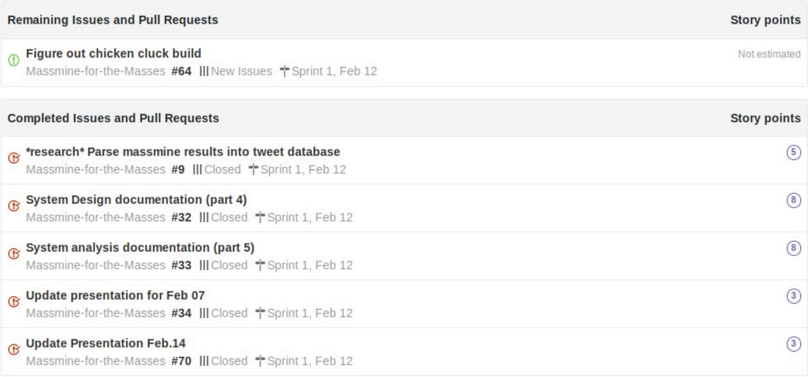
Start **Feb 5, 2019** Change Due by **Feb 12, 2019 - Past due by a day** Change

Burndown report

□ Weekends — Ideal — Completed

27 Total Story Points
27 Completed / 0 Remaining

6 Total Issues and Pull Requests
5 Completed / 1 Remaining



Remaining Issues and Pull Requests	Story points
Figure out chicken cluck build Massmine-for-the-Masses **#64** New Issues Sprint 1, Feb 12	Not estimated

Completed Issues and Pull Requests	Story points
research Parse massmine results into tweet database Massmine-for-the-Masses **#9** Closed Sprint 1, Feb 12	5
System Design documentation (part 4) Massmine-for-the-Masses **#32** Closed Sprint 1, Feb 12	8
System analysis documentation (part 5) Massmine-for-the-Masses **#33** Closed Sprint 1, Feb 12	8
Update presentation for Feb 07 Massmine-for-the-Masses **#34** Closed Sprint 1, Feb 12	3
Update Presentation Feb.14 Massmine-for-the-Masses **#70** Closed Sprint 1, Feb 12	3

-

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 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

7.4 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