Massmine for the Masses

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

Table of Contents

[**1. Project Definition**](#_f6pns9jm1fk3) **4**

[1.1 Why](#_wckxeknchnp2) 4

[1.2 What](#_8pqoao6zguhn) 4

[1.3 How](#_ac67agwzegxq) 4

[**2. Project Requirements**](#_s8tvbcpndfdn) **4**

[2.1 Functional](#_nut9bfa4x5r4) 4

[2.2 Usability](#_p6102w55pc12) 4

[2.2.1 User interface](#_cm0uglhdzq4d) 4

[2.2.2 Performance](#_obcm4tqhdre9) 5

[2.3 System](#_oc4t3wniflsg) 5

[2.3.1 Hardware](#_683g7pjlevav) 5

[2.3.2 Software](#_nc8gzajo58wn) 5

[2.3.3 Database](#_pi0zjke0ru85) 5

[2.4 Security](#_xzvt94xc9fhb) 5

[**3. Project Specification**](#_eksquogfbmju) **6**

[3.1 Focus](#_rg433nvb9yv) 6

[3.2 Domain](#_hmz5zpc4eyke) 6

[3.3 Area](#_y5trxio2ijz9) 6

[3.4 Libraries](#_jked61g3emcr) 6

[3.5 Development Environment](#_13rpc1v4zvqk) 6

[3.6 Framework](#_8uisdeskcnpl) 6

[3.7 Platform](#_mzvwrs9cvj6c) 6

[3.8 Genre](#_5y6p0zahmzml) 7

[**4. System – Design Perspective**](#_a5j4ixnmnpey) **7**

[4.1 Identify subsystems – design point of view](#_3umvy5vsdveo) 7

[4.2 Sub-System Communication](#_23df5mtsj783) 12

[4.3 Entity Relationship Model](#_qvg43d2qlux8) 15

[4.4 Overall operation - System Model](#_3y7zpqrcr1im)  17

[**5. System – Analysis Perspective**](#_lnwvi8acowhv) **17**

[5.1 Identify subsystems – analysis point of view](#_bxo700v8xz0t) 18

[5.2 System (Tables and Description)](#_bj18u0w7fth6) 19

[5.2.1 Data analysis](#_it8ed9kqjnr)  19

[5.2.2 Process models](#_at9axl9vgcdb)  31

[5.3 Algorithm Analysis](#_933hvher64ls) 31

[5.3.1 Big - O analysis of overall System and Sub-Systems](#_wt06g8nan9n8) 31

[**6. Project Scrum Report**](#_vlc1r9u9hvb9) **32**

[6.1 Product Backlog (Table / Diagram)](#_mcoqhh8a85vm) 32

[6.2 Sprint Backlog (Table / Diagram)](#_jj6jdkr9qw9u) 33

[6.3 Burndown Chart](#_vx7yl22u7qwn) 34

[**7. Subsystems**](#_dbbrecckf7l2) **36**

[7.1 Subsystem 1 – User Management -](#_lwdx0mmrvoty) Patti 36

[7.2 Subsystem 2 – Analysis Management -](#_qwg9ggqud3v0) Morgan 36

[7.3 Subsystem 3 – Query Management -](#_hqjuk6py8zx) Logan 37

[7.4 Subsystem 4 – Massmine Integration -](#_3ixyol1ashjz) Josh 37

[**8. Complete System**](#_hrqicvy4qa2o) **38**

# **1. Project Definition**

## 1.1 Why

Massmine ([www.massmine.org](https://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

2.1.1 Dockerized web application and associated database to:

* + 1. Store user information (username, password, Twitter authorization keys, first and last name if desired)
    2. Store collected Twitter data (tweet id, author, content, etc.)
    3. Create and update new users (change oauth keys, email, etc.)
    4. Allow users to gather Twitter data. The mass of data collected by a query is a ‘study’.
    5. Allow users to perform basic analysis on this Twitter data.

## 2.2 Usability

### 2.2.1 User interface

* + 1. Login interface: should allow the creation of accounts, as well as signing in.
    2. User account interface: Allow user to enter or change their Twitter authentication codes, as well as their email and personal information.
    3. 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.
    4. Query interface: Allow the user to create queries or ‘studies’.
    5. 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

* + 1. 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. Multiple uses should be able to use the web-app at once. Up to about 10 accounts expected.

## 2.3 System

### 2.3.1 Hardware

* + 1. Any computer or server capable of running a Docker container.

### 2.3.2 Software

* + 1. Web application from Django framework, programmed in Python and HTML.
    2. Additional modules in Python to communicate with Massmine..

### 2.3.3 Database

* + 1. SQLite, provided by Django framework. While SQLite has a theoretical database size limit at 140TB, it does store its database as a single file, which means that users may encounter size restrictions based on their file system. If more than about 4GB is desired, it may be necessary to either create a new container or use a different backend database.

## 2.4 Security

* 1. Passwords used to sign in to the account will be salted and hashed.
  2. 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.
  3. The user queries will be scrubbed for invalid input

# 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 Required installations in environment

* + Python3
  + Django 2.1.7
  + Pip 19.03
  + Numpy 1.16.2
  + Matplotlib 3.0.3
  + Plotly 3.7.1
  + Cufflinks 0.15
  + Pandas 0.24.2
  + Celery 4.3.0 (rhubarb)
  + Erlang OTP 20
  + Rabbitmq-server 3.6.10
  + Django-celery-results 1.0.4 (stable)
  + Django-encrypted-model-fields 0.5.8
  + Expect 5.45.4
  + Django-tables2 2.0.6
  + Enchant 2.0.0
  + Textblob 0.15.3
  + Massmine 1.1.0

## 3.5 Development Environment

* + Pip virtualenv
  + Django 2.1.7
  + Ubuntu 18.04 host(virtual machine)

## 3.6 Framework

* + Django 2.1.7, a high-level Python web framework. This framework is scalable and secure.
  + SQLite is our framework for the databases. It is the default database for Django and fills our limited needs as a proof of concept.
  + Server: We use the default testing server for Django, which does not support https.

## 3.7 Platform

* + Docker image meant for desktop access on a protected network. This platform allows for user usage on any operating system, although the server should be running a Linux distribution to run the Massmine tool. Users desiring to make this a public-facing web page will need to download from source and set up an https server for security reasons, and either an smtp server or a third-party smtp service for the forgot password functionality if desired.

## 3.8 Genre

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

# **4. System – Design Perspective**

## 4.1 Identify subsystems – design point of view

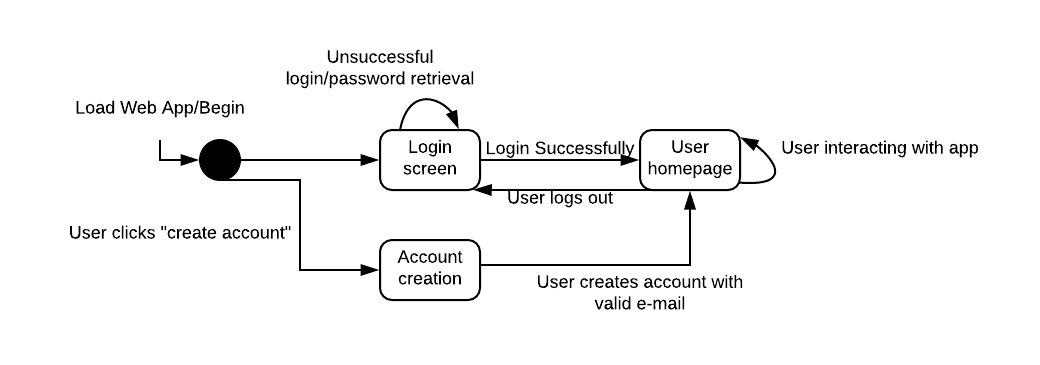
* + Subsystem 1: User Management - Patti
    - Home page
      * Gives options to register, log in, or go to the admin page (requires authentication)
    - User Profile:
      * Username
      * Email
      * Password
      * Consumer key
      * Consumer secret
      * Access token
      * Access token secret
      * First name (not required)
      * Last name (not required)
    - Registration:
      * Takes some user profile information at time of registration. Ensures password meets security standards
    - Edit Profile:
      * Allow user to change everything in their profile except their username.
    - Login/Logout
      * The user should be able to log out from any page
      * Login should only be available from the home page
    - Administration
      * Can edit, delete, create users
      * Can see all user information
  + Subsystem 2: Analysis Management: - Morgan
    - 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
    - User may choose one of these analyses on the Analysis page and feed it the study ID of the study to be sorted, so that the results are graphically displayed
  + Subsystem: Query Management - Patti
    - Accepts keyword input from user and a count of how many tweets to gather, and uses it to gather the most recent ‘x’ tweets where x is the count indicated by the user.
    - Stores individual tweets as an object in the database.
  + Subsystem: Massmine Integration - Patti
    - Group of tweets returned by a Massmine command are parsed and stored in the database as a ‘study’ for future analysis.
    - Formats command sent as query to Massmine and authenticates user credentials with Massmine
    - Studies are made viewable as a table without analysis on the Analysis page.
    - Provides functionality to choose from a list of studies created by only the currently logged in user for analysis

## 4.2 Sub-System Communication

* Controls by subsystem - Morgan + Patti

## Subsystem 1: User Management and Templates

* + - Django has built-in controls to prevent cross site scripting and cross site request forgery. - Morgan
    - The user may navigate from page to page with buttons on the top of the screen. - Morgan
    - Data entry, including username, password, email, and oauth keys, will be checked for sql injection, invalid characters, etc. - Patti
    - Account creation will require a unique username and password that fits length and complexity standards. - Morgan
    - Login will require the correct username and password combination and verified in database. - Patti
    - User passwords will be hashed and salted in the user database. - Patti
    - Authorization keys are encrypted at rest. - Patti
    - User credentials must be verified to access the edit profile page and change password page, as well as the admin pages. - Patti
    - Login diagram - Morgan



## Subsystem 2: Analysis Management - Morgan

* + - User credentials must be verified to access the analysis page.
    - This subsystem will communicate with the Massmine integration section.
    - 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 study model.
    - Only uses tweets from user’s own studies.
    - References the study database based on user input (study id), but will only accepts integers to avoid injection.

## Subsystem 3: Query - Patti

* + - User credentials must be verified to access this page.
    - User can select various qualifications on their search such as date and a word or phrase to search for.
    - Input words or phrases should be scrubbed for command injection.
  + Subsystem 4: Massmine Integration - Morgan
    - Users need oauth keys to be able to gather twitter data. In the Massmine communication if a key is incorrect the user can’t request a study.
    - Massmine itself is a command line tool, and will be running concurrently on the local machine.
* I/O by subsystem - Morgan + Patti
  + Inputs
    - Subsystem 1: User Management - Patti
      * Username from user
      * Password from user
      * Email from user
      * Consumer key from user (used in Twitter development authorization)
      * Consumer secret key from user (used in Twitter development authorization)
      * Access token from user (used in Twitter development authorization)
      * Access secret token from user(used in Twitter development authorization)
      * First name from user (not required)
      * Last name from user (not required)

## Subsystem 2: Analysis - Morgan

* + - * Analysis requirements from user (choose from a drop-down menu)
      * Study ID to distinguish which study will be used

## Subsystem 3: Query - Morgan

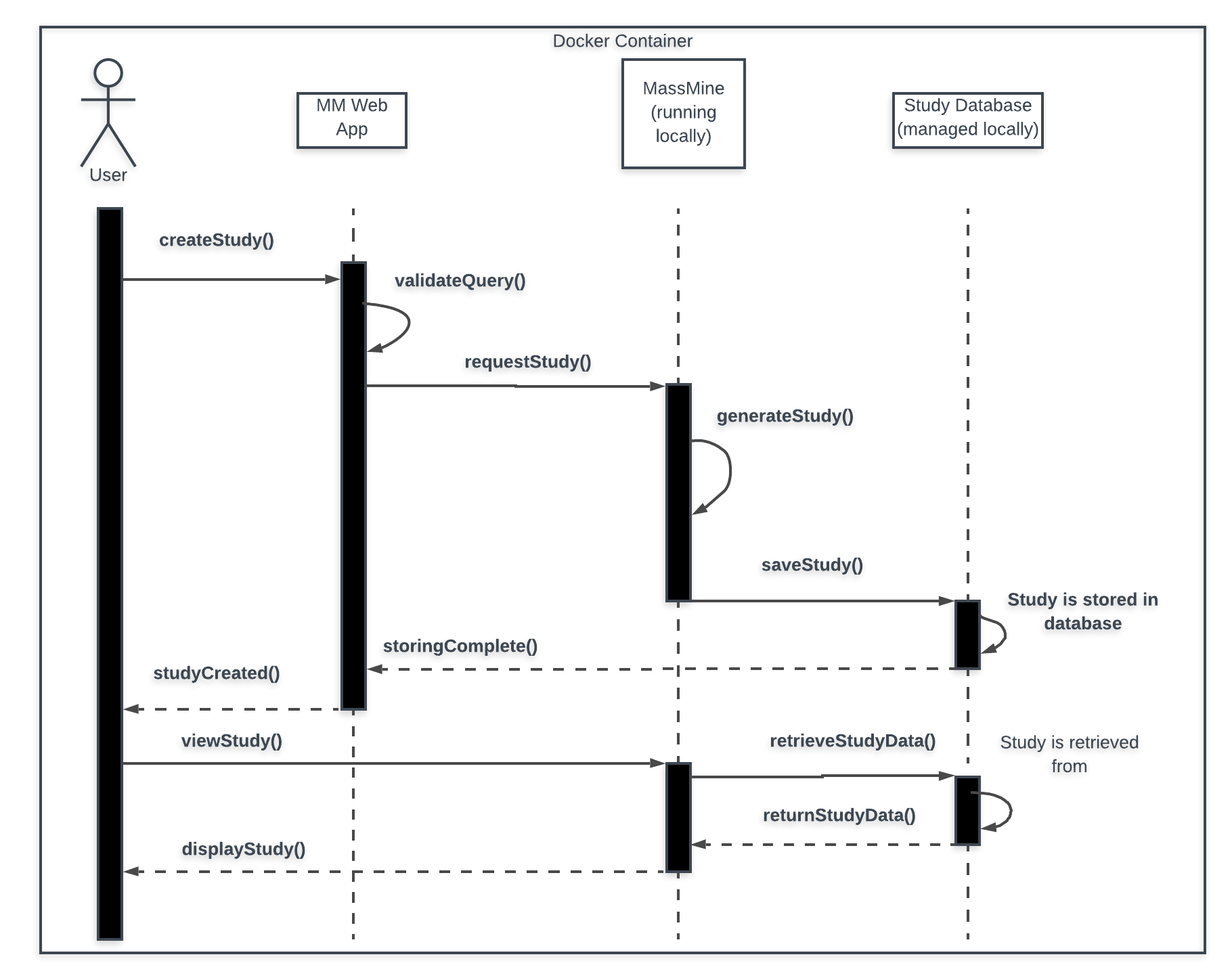
* + - * Study requirements from user (date, search terms, etc.)
    - Subsystem 4: Massmine communication - Morgan
      * User information (oauth keys)
      * User query
  + Outputs - Morgan
    - Subsystem 1: User Management
      * Valid/Invalid response for authentication checking

## Subsystem 2: Analysis - Morgan

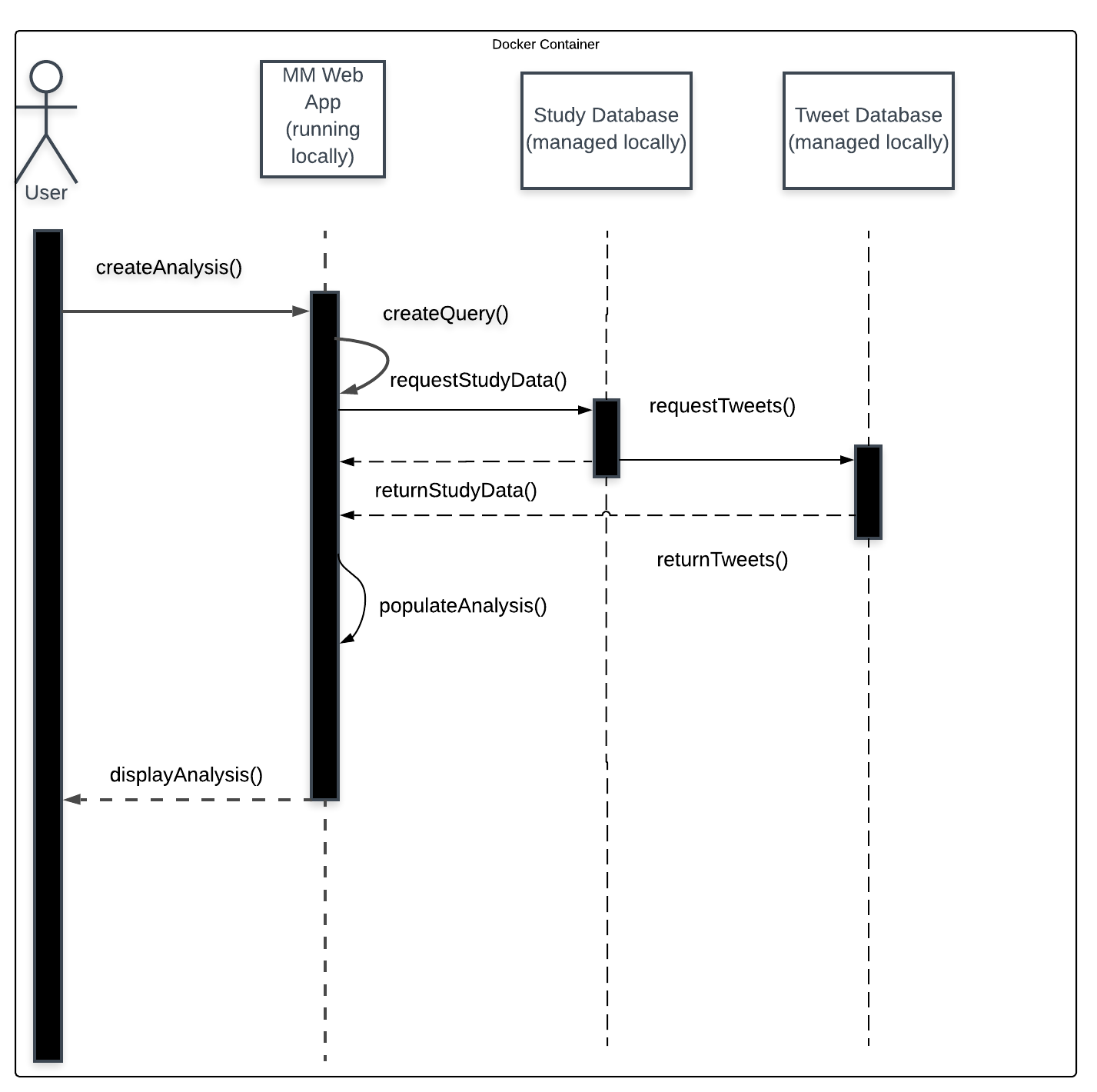
* + - * Gives options for analysis (tweets to select, conditions to set)
      * Analysis results to user as graphical display
      * Gives error message if the user is not logged in on analysis page.

## Subsystem 3: Query - Logan

* + - * Gives options for query (date, time, phrases)
      * Sends command line query to massmine
    - Subsystem 4: Massmine communication - Josh
      * Results from Massmine to tweet storage
      * Displays user’s studies
* Dataflow - Morgan
  + Data-flow for creating a study

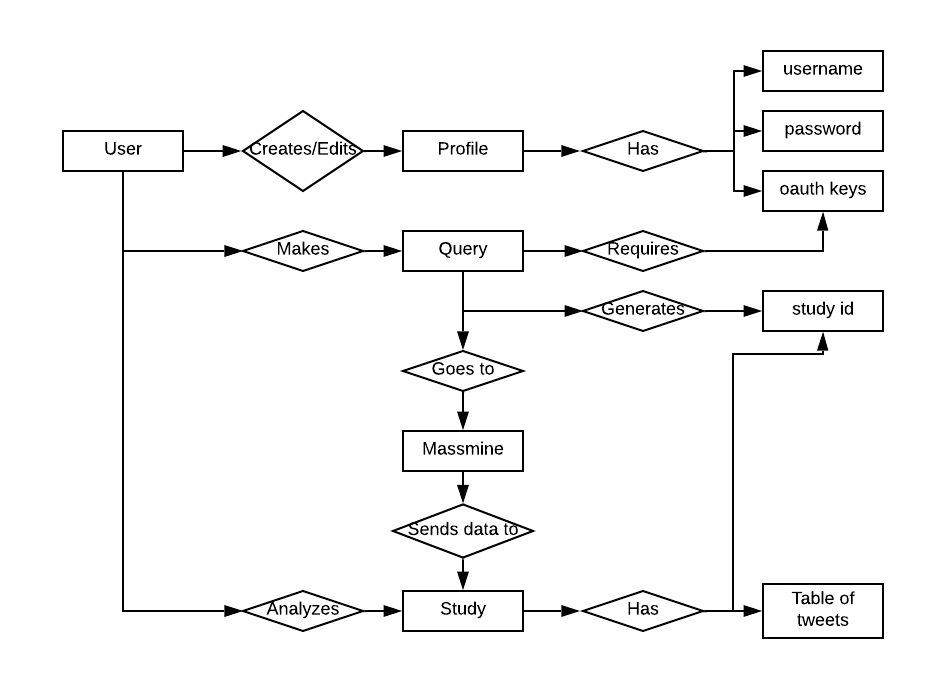


* Data-flow for creating an analysis - Morgan

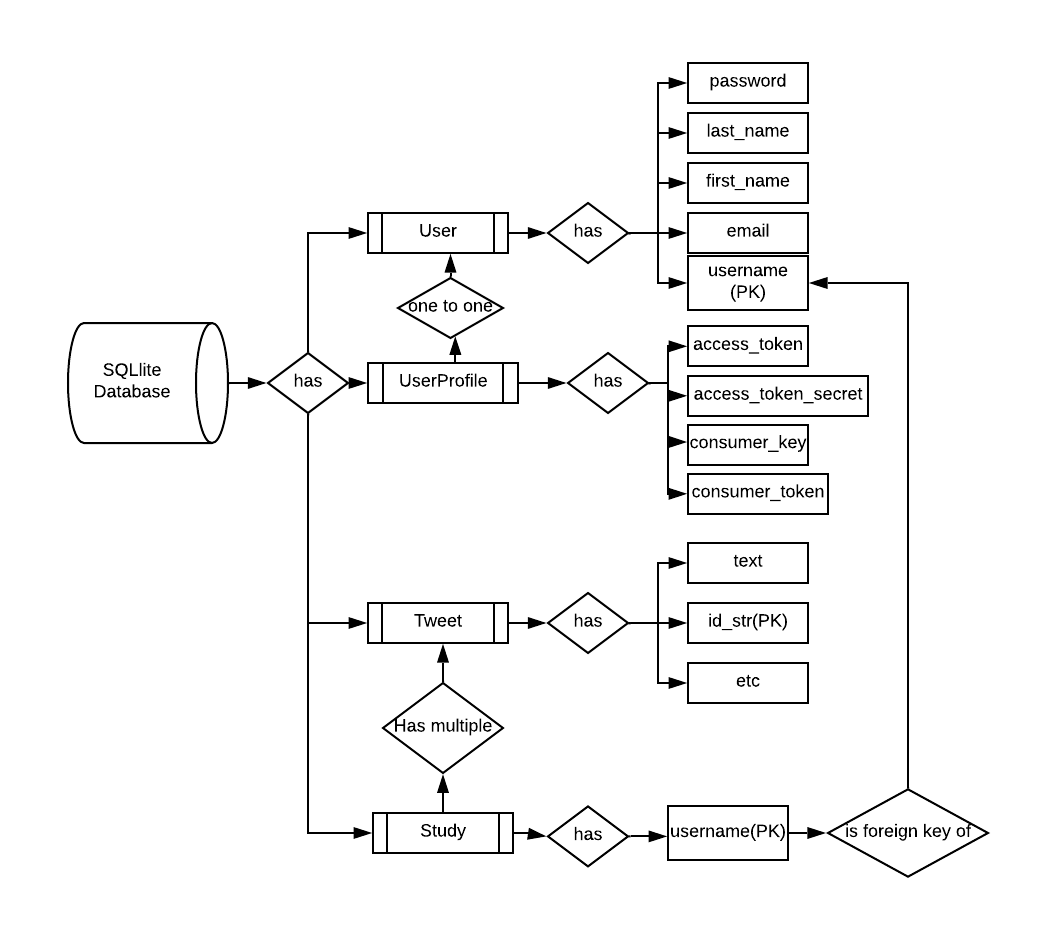


## 4.3 Entity Relationship Models

* System Wide ERM: - Patti

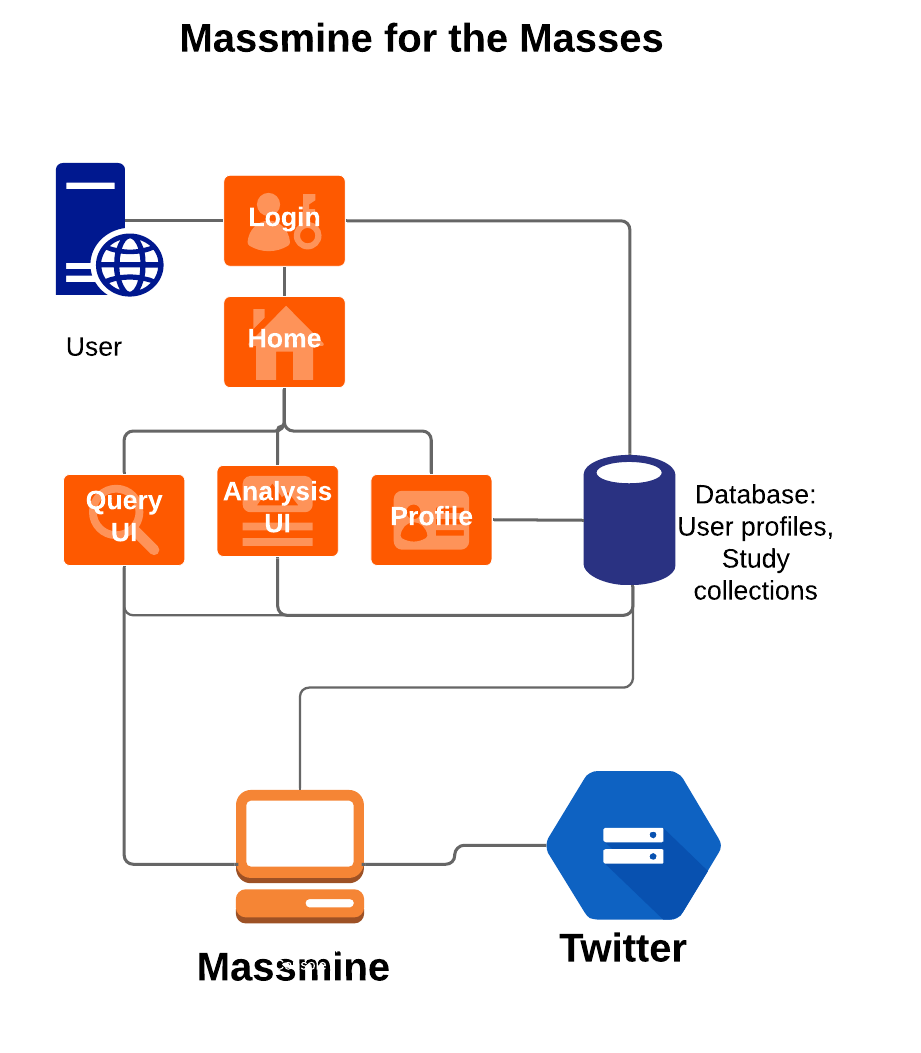


* Database ERM: - Patti



## 

## 4.4 Overall operation - System Model - Logan + Patti



# **5. System – Analysis Perspective**

## 5.1 Identify subsystems – analysis point of view

## Subsystem 1: User Management - Patti

* + Web GUI
    - Registration
    - Login
    - Logout
    - Edit Profile
    - Change Password
    - Admin
  + Data Storage:
    - User and User Profile model stored in local database.
    - Sensitive fields, including all four Twitter developer keys, are encrypted at rest.
  + Data Structure:
    - User model is provided by django.contrib.auth.
    - User Profile has a one-to-one relationship with the User model to add extended attributes.
  + Performance:
    - Anticipate up to 10 users per Docker container.

## Subsystem 2: Analysis - Patti + Morgan

* + 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 3: Query - Patti

* + Structure:
    - Accepts keywords from the user (individual words or a phrase) and a number ‘x’ of how many of the most recent tweets containing that keyword or phrase should be gathered

## Performance

* + - Must be able to perform asynchronous querying
* Subsystem 4: Massmine communication - Patti
  + Structure:
    - Host runs an instance of Massmine for each user query.
    - Each group of tweets (a ‘study’) has its own id linked to that group of tweets and the user who created the study.
    - These studies should be available for download.
  + Performance:
    - The host should support at up to 10 instances of Massmine asynchronously (one per expected user).

## 5.2 System (Tables and Description)

### 5.2.1 Data analysis - Patti

* + - Data dictionary:
    - User model: taken from <https://docs.djangoproject.com/en/2.1/ref/contrib/auth/>

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Type** |
| username | Name identifying a user | CharField, max length 150. May contain alphanumeric, @, +, . and - characters. Primary key. |
| password | Hashed and salted password to confirm user login | CharField. |
| first\_name | Optional detail if users wish to personalize their accounts. | CharField. 30 characters or fewer. |
| last\_name | Optional detail if users wish to personalize their accounts. | CharField. 150 characters or fewer. |
| email | Optional detail if users wish to personalize their accounts. In future releases could be used to provide notifications and recover password support. | CharField, formatted as email address. |
| is\_staff | Designates whether this user can access the admin site. Admin viewable only. | Boolean. |
| is\_active | Designates whether user is active. May use instead of deleting user. Admin viewable only. | Boolean. |
| is\_superuser | Designates that this user has all permissions. Admin viewable only. | Boolean. |
| last\_login | Datetime of the user’s last login. Admin viewable only. | DateTime |
| date\_join | Datetime of when account was created. Admin viewable only. | DateTime |

* + - User Profile

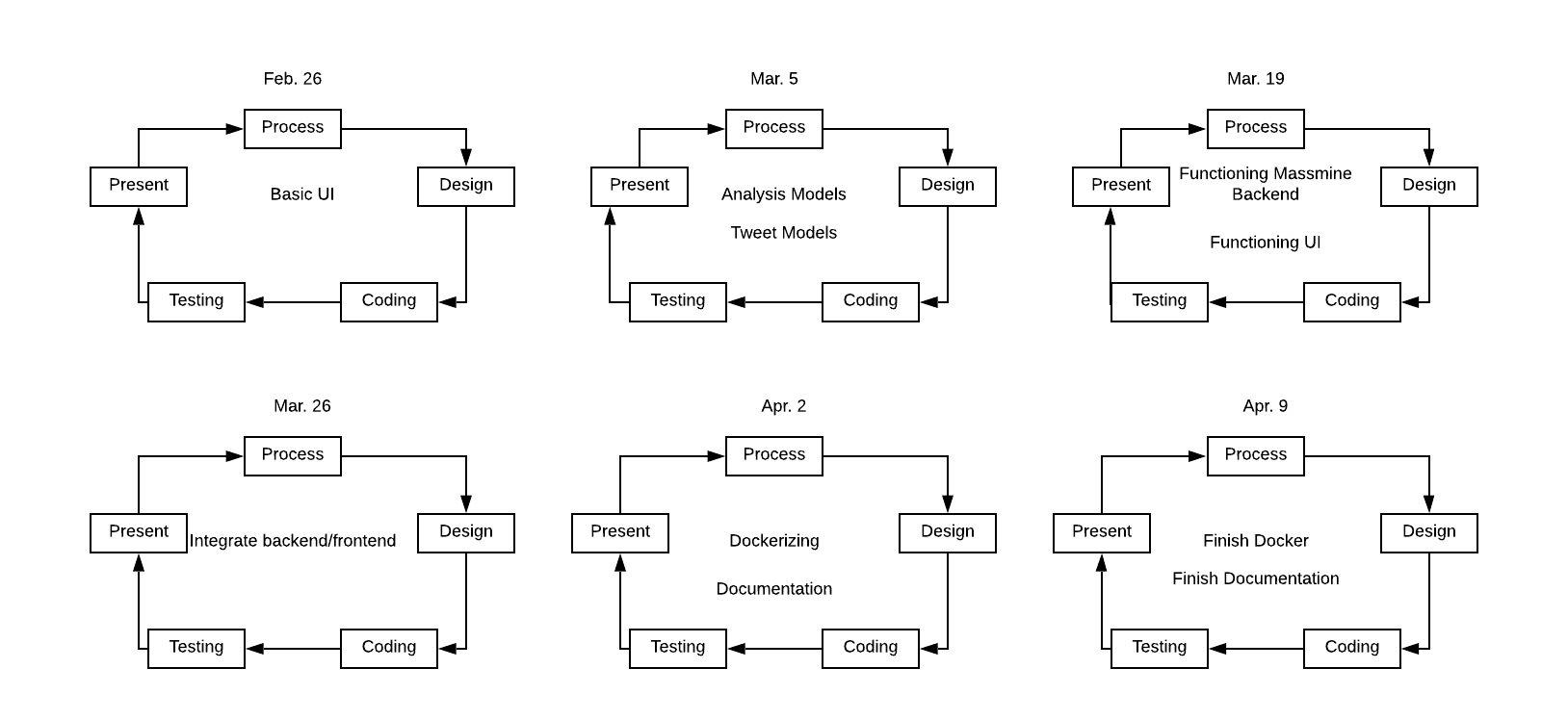
|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Type** |
| User | One-to-One link to user model | Primary Key |
| consumer\_key | Twitter authorization key. | EncryptedCharField, max\_length 50 |
| consumer\_secret | Twitter authorization key | EncryptedCharField, max\_length 50 |
| access\_token | Twitter authorization key | EncryptedCharField, max\_length 50 |
| access\_token\_secret | Twitter authorization key | EncryptedCharField, max\_length 50 |

* + - Tweet:
      * + Attributes found in the official Twitter tweet model: <https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object.html> )

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Type** |
| created\_at | UTC time when this Tweet was created. Example:  "created\_at":"Wed Aug 27 13:08:45 +0000 2008" | CharField, max length 100 |
| 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". Twitter calls it ‘id\_str’ | Primary key, CharField, max length 100 |
| text | The actual UTF-8 text of the status update. See [twitter-text](https://github.com/twitter/twitter-text/blob/master/rb/lib/twitter-text/regex.rb) for details on what characters are currently considered valid. Example:  "text":"Tweet Button, Follow Button, and Web Intents" | TextField, max length 280 |
| device | 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". Twitter calls this field ‘source’ | CharField, max length 100 |
| 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 | BooleanField, default false |
| retweet\_count | Number of times the tweet has been retweeted | IntegerField, default 0 |
| 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" | CharField, max length 100 |
| 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" | CharField, max length 100 |
| 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" | CharField, max length 100 |
| user\_id\_str | Taken from user object of Tweet. | CharField, max length 100 |
| name | Taken from user object of Tweet. | CharField, max length 100 |
| screen\_name | Taken from user object of Tweet. | CharField, max length 100 |
| url | Taken from user object of Tweet. | CharField, max length 100 |
| description | Taken from user object of Tweet. | CharField, max length 100 |
| verified | Taken from user object of Tweet. | BooleanField, default false |
| followers\_count | Taken from user object of Tweet. | IntegerField, default 0 |
| friends\_count | Taken from user object of Tweet. | IntegerField, default 0 |
| listed\_count | Taken from user object of Tweet. | IntegerField, default 0 |
| favorites\_count | Taken from user object of Tweet. | IntegerField, default 0 |
| num\_tweets | Taken from user object of Tweet. | IntegerField, default 0 |
| user\_created\_at | Taken from user object of Tweet. | DateTimeField, default null |
| utc\_offset | Taken from user object of Tweet. | CharField, max length 100 |
| time\_zone | Taken from user object of Tweet. | DateTimeField, default null |
| geo\_enabled | Taken from user object of Tweet. | BooleanField, default false |
| hashtags | Parsed from text of tweet | CharField, max |
| lang | *Nullable.* When present, indicates a [BCP 47](http://tools.ietf.org/html/bcp47) language identifier corresponding to the machine-detected language of the Tweet text, or und if no language could be detected. See more documentation [HERE](http://support.gnip.com/apis/powertrack2.0/rules.html#Operators). Example:  "lang": "en" | CharField, max length 100 |

### 5.2.2 Process models - Patti

* System Process Model (Agile):



## 5.3 Algorithm Analysis

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

* User Management - Patti
  + Updating/adding user: O(log n) - Looks for username primary key.
  + Checking authentication: O(log n) - Looks for username primary key and checks hashed password
* Analysis Management - Morgan

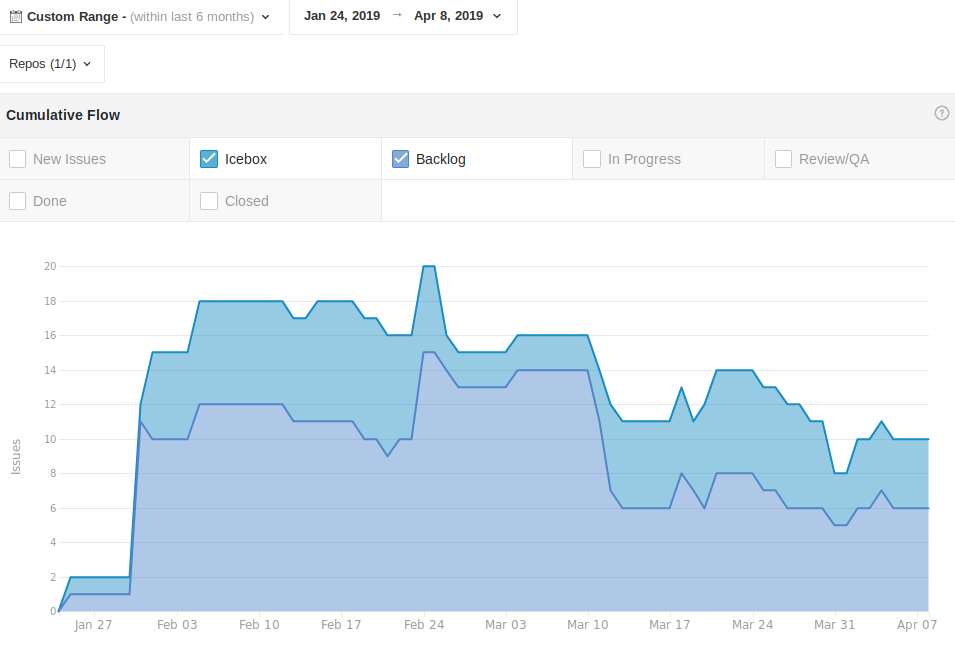
## O(n) - Displaying the analysis will require parsing through a study and displaying said study.

* + O(log(n)) - Retrieving tweets from the integrated django database
* Query Management - Patti
  + O(1) - formats user command and sends it to the massmine subsystem
* Massmine Integration - Morgan
  + 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

# **6. Project Scrum Report - Patti**

## 6.1 Product Backlog

* Overview:
  + Jan. 24-Apr. 8



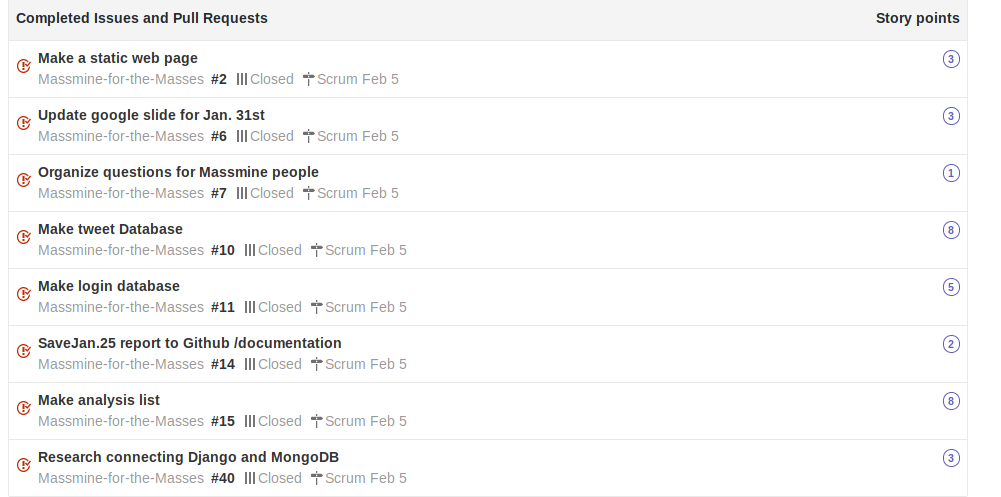
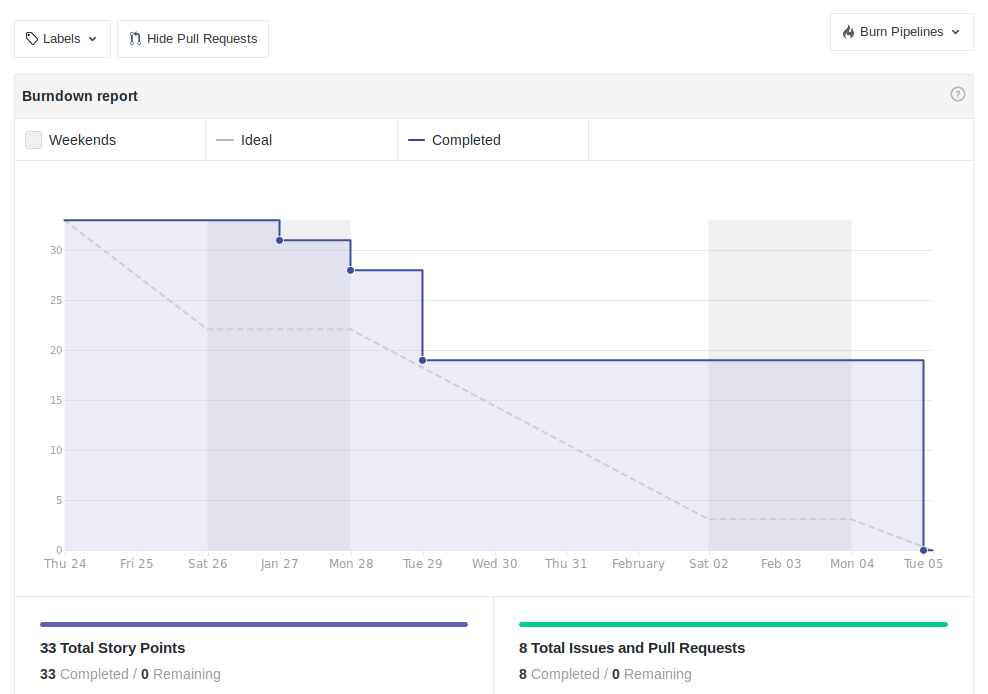
* Initial Scrum (Feb. 5):
  + Make tweet database
  + Make login database
  + Make analysis list
  + Research connecting django and MongoDB
  + \*research\* Parse massmine results into tweet database
  + Organize questions for massmine people
  + Make a static web page
  + Save Jan. 25 report to Github, documentation
  + System analysis documentation (part 5)
  + System design documentation (part 4)
  + Update Presentation Feb. 14
  + Update presentation for Feb. 07
* Sprint 1 (Feb. 12)
  + Add oauth attribute to web page
  + Create user page
  + Make user entry
  + Login page crypto/validation
  + Link login database to login page
  + User management pages
  + Add password confirmation to registration
  + Change password
  + Analysis UI
  + Make analysis database
  + Create Analysis Dictionary
  + Research using separate ports for Massmine
  + Set up logging
  + Practice Presentation 4/26
  + Make presentation 2-26
  + Figure out chicken cluck build
  + Create proper requirements file to be used with docker
  + Make web application all https
  + Research production server
* Sprint 2 (Feb. 19)
  + Set up logging
  + Practice Presentation 4/26
  + Make presentation 2-26
  + Figure out chicken cluck build
  + Create proper requirements file to be used with docker (Yamel)
  + Fix presentation
  + Add password confirmation to registration
  + Change password
  + Link login database to login page
  + Analysis UI
  + Query UI
  + Fix user in webapp
  + Make web application all https
  + Research production server
  + Enforce encryption measures on client side
  + Enforce encryption on server side
  + Enforce encryption/hashing of username, passwords, oauth
* Sprint 3 (Feb. 26)
  + Query UI
  + Massmine query script
  + Implement Plotly for Analysis
  + Add success page operations
  + Fix admin page
  + Add way to change oauth/email
  + Fix font sizes on profile pages
  + Enforce encryption measures on client side
  + Enforce encryption on server side
  + Enforce encryption/hashing of username, passwords, oauth
  + Password reset basic or password hints
  + Figure out celery implementation
  + Encrypt oauth at rest
  + Klout scores
  + Implement logging
  + Create error pages for erroneous login
  + Add template to existing error pages
* Sprint 4 (Mar. 12)
  + Fix Scrum report
  + Create study object in a model.py
  + Create Tweet object in a model.py
  + Query UI
  + Have graph pull from django database
  + Fix documentation graphs
  + Link Query UI to massmine
  + Password reset basic or password hints
  + Figure out celery implementation
  + Encrypt oauth at rest
  + Klout scores
  + Implement logging
  + Show analysis graph in UI
  + Have graph either open a new tab or integrate well with web app
  + Have a unique response for each item in drop down of analysis type
  + Create error pages for erroneous login
  + Add template to existing error pages
  + Give admin a profile so he/she can see that page
  + Fix registration bug
  + Fix documentation graphs
  + Link analysis UI to analysis database
* Sprint 5 (Mar. 19)
  + Link Query UI to massmine
  + Create Tweet Object in a model.py
  + Create Study object in a modelpy
  + Have graph pull from django database
  + Have query page grab user oauth
  + Parse Json and store tweet in database
  + Make bash script installation
  + Encrypt Oauth at rest
  + Make page to view studies
  + Add parsing extractor to query views.py
  + Test downloading/running docker image
  + Figure out celery implementation
  + Subsystem documentation - Logan
  + Subsystem documentation - Patti
  + Subsystem documentation - Morgan
  + Subsystem documentation - Josh
  + Klout scores
  + Implement logging
  + Dockerize it
  + Have massmine pipe json object to django
  + Update backlog screenshots in doc for Mar. 24
  + Update installation instructions
  + Install Jsan in environment
  + Remove requirements.txt file
  + Add pandas to installation instructions
  + Async research
  + Make celery async example
  + Parse Json and store tweet in database
  + Add parsing extractor to query views.py
  + Update presentation for mar 21
* Sprint 6 (Mar. 26)
  + Link Query UI to massmine
  + Create Tweet Object in a model.py
  + Create Study object in a modelpy
  + Have graph pull from django database
  + Have query page grab user oauth
  + Parse Json and store tweet in database
  + Make bash script installation
  + Encrypt Oauth at rest
  + Make page to view studies
  + Add parsing extractor to query views.py
  + Test downloading/running docker image
  + Figure out celery implementation
  + Subsystem documentation - Logan
  + Subsystem documentation - Patti
  + Subsystem documentation - Morgan
  + Subsystem documentation - Josh
  + Klout scores
  + Implement logging
  + Dockerize it
* Sprint 7 (Apr. 02)
  + Subsystem documentation - Morgan
  + Subsystem documentation - Patti
  + Subsystem documentation - Josh
  + Subsystem documentation - Logan
  + Test downloading/running docker image
  + Link view studies to analysis
  + Freq\_words analysis
  + Verify user credentials with massmine
  + Query page implement asynchronous
  + Have query page grab user oauth
  + Klout scores
  + Implement logging
  + Make bash script installation
  + Massmine instances different test

## 6.2 Sprint Backlog (Charts in burndown report section)

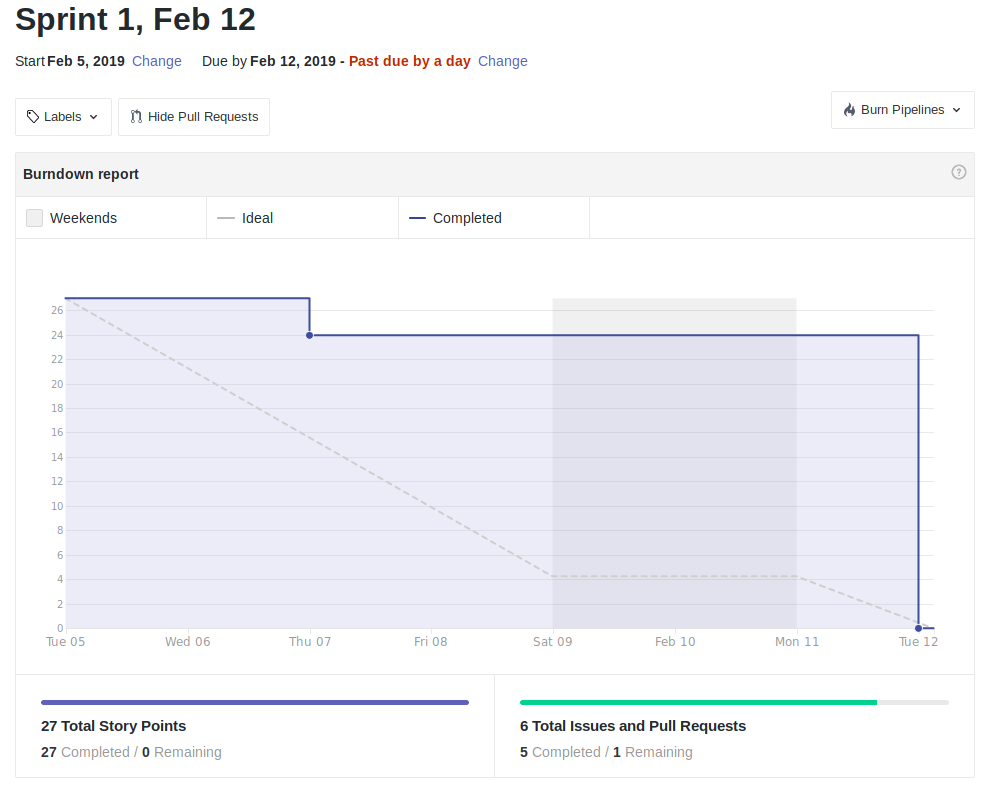
* Initial Scrum (Feb. 5):
  + Make tweet database (8, Logan) (made irrelevant by project restructuring)
  + Research connecting django and MongoDB (3, Josh, Patti) (made irrelevant by project restructuring)
  + Make analysis list (8, Morgan)
  + Organize questions for massmine people (1, Logan)
  + Update google slide for Jan. 31st (3, Morgan)
  + Make login database (5, Patti) (Made irrelevant by project restructuring)
  + Make a static web page (3, Josh)
  + Save Jan. 25 report to Github, documentation (2, Patti)
* Sprint 1 (Feb. 12)
  + \*research\* Parse massmine results into tweet database (5, Logan)
  + System analysis documentation (part 5) (8, all members)
  + System design documentation (part 4) (8, all members)
  + Update Presentation Feb. 14 (3, Morgan)
  + Update presentation for Feb. 07 (3, Patti)
* Sprint 2 (Feb. 19)
  + Make production server (https) (1, no member, made irrelevant by project restructuring)
  + Create Analysis Dictionary (5, Morgan)
  + Add oauth attribute to web page (3, Patti)
  + Create user page (5, Patti and Josh)
  + Make user entry (2, Josh)
  + Login page crypto/validation (3, Patti and Josh)
  + Make analysis database (8, Morgan)
  + User management pages (no estimate, mistakenly created)
  + Research using separate ports for Massmine(8, Josh)
* Sprint 3 (Feb. 26)
  + Set up logging (2, Josh)
  + Practice Presentation 4/26 (2, all members)
  + Make presentation 2-26 (2, Patti and Morgan)
  + Figure out chicken cluck build (21, Josh)
  + Create proper requirements file to be used with docker (Yamel) (1, Morgan and Josh)
  + Fix presentation (2, no assigned member)
  + Add password confirmation to registration (3, Patti)
  + Change password (5, Patti)
  + Link login database to login page (3, Patti, made irrelevant by project restructuring)
  + Analysis UI (8, Morgan)
  + Fix user in webapp (8, Patti)
  + Make web application all https (1, no assigned member, made unnecessary by project restructuring)
  + Research production server (1, Josh, made unnecessary by project restructuring)
* Sprint 4 (Mar. 12)
  + Massmine query script (3, Josh)
  + Implement Plotly for Analysis (8, Morgan)
  + Add success page operations (no estimate, Patti)
  + Fix admin page (8, Patti)
  + Add way to change oauth/email (5, Patti)
  + Fix font sizes on profile pages (1, Patti)
  + Enforce encryption measures on client side (no estimate, made unnecessary by project restructuring)
  + Enforce encryption on server side (no estimate, made unnecessary by project restructuring)
  + Enforce encryption/hashing of username, passwords, oauth (no estimate, created by mistake)
* Sprint 5 (Mar. 19)
  + Create study object in a model.py (no estimate, no assigned member)
  + Create Tweet object in a model.py (no estimate, Logan)
  + Query UI (13, Logan)
  + Show analysis graph in UI (no estimate, Morgan)
  + Have graph pull from django database (3, Morgan)
  + Have Graph either open a new tab or integrate well with web app (3, Morgan)
  + Have a unique response for each item in drop down of analysis type (3, Morgan)
  + Link Query UI to massmine (8, Logan, Josh)
  + Create error pages for erroneous login (1, Patti)
  + Add template to existing error pages (1, Patti)
  + Give admin a profile so he/she can see that page (2, Patti)
  + Regression test (1, Patti)
  + Add massmine installation instructions (2, Patti)
  + Make example plotly page (3, Patti)
  + Fix registration bug (13, Patti)
  + Fix documentation graphs (3, Patti)
  + Fix Scrum report (3, Patti)
  + Link analysis UI to analysis database (no estimate, made irrelevant by project restructuring)
* Sprint 6 (Mar. 26)
  + Link Query UI to massmine (8, Josh andLogan)
  + Create Tweet Object in a model.py (3, Logan)
  + Have graph pull from django database (3, Morgan)
  + Have massmine pipe json object to django (3, Morgan)
  + Update backlog screenshots in doc for Mar. 24 (1, Patti)
  + Have query page grab user oauth (2, Logan)
  + Update installation instructions (1, Patti)
  + Install Jsan in environment (3, Patti)
  + Remove requirements.txt file (1, Patti)
  + Add pandas to installation instructions (1, Patti)
  + Async research (2, Patti)
  + Make celery async example (2, Patti)
  + Parse Json and store tweet in database (no estimate, Logan)
  + Add parsing extractor to query views.py (5, all members)
  + Update presentation for mar 21 (2, no assigned member)
* Sprint 7 (Apr. 02)
  + Parse Json and store tweet in database (5, Logan)
  + Add parsing extractor to query views.py (5, all members)
  + Create Study object in a modelpy (5, Josh)
  + Create Study object in a modelpy (5, Josh)
  + Presentation for Mar. 28 (1, all members)
  + Fix scrum reports in documentation (1, Patti)
  + Make page to view studies (5, no assigned member)
  + Encrypt oauth at rest (5, Patti)
  + Have graph pull from django database (3, Morgan)
  + Test downloading/running docker image (2, no assigned member)
  + Dockerize it (5, Josh)
* Sprint 8 (Apr. 09)
  + Subsystem documentation - Morgan (5, Morgan)
  + Subsystem documentation - Patti (5, Patti)
  + Subsystem documentation - Josh (5, Josh)
  + Subsystem documentation - Logan (5, Logan)
  + Test downloading/running docker image (2, no assigned member)
  + Link view studies to analysis (5, Josh)
  + Freq\_words analysis (5, Morgan)
  + Take in all 4 massmine credentials instead of one (1, Patti)
  + Query page implement asynchronous (8, no assigned member)
  + Have query page grab user oauth (2, Logan)
  + Credential verification (2, Patti)
  + Presentation for Apr. 4 (1, all members)
  + Credential setup script (1, Patti)
  + Fix main documentation (2, Patti)

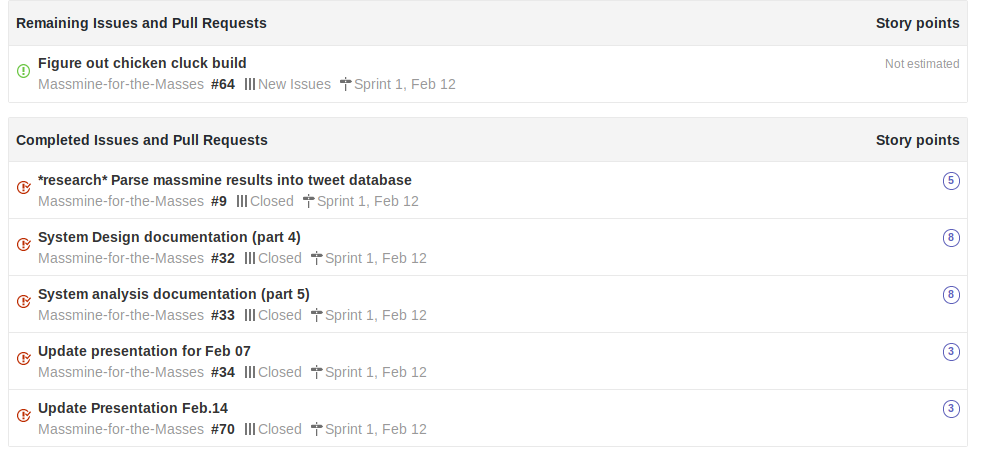
## 6.3 Burndown Chart and Backlog Items Per Sprint

* For Scrum Feb 5:

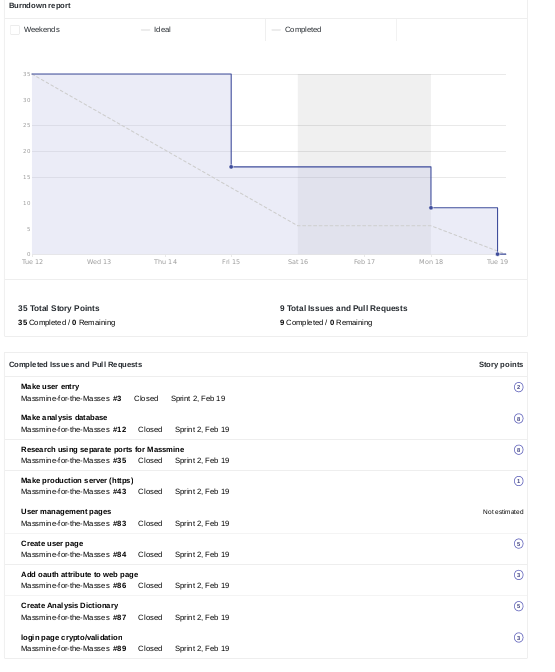


* For Scrum Feb 12:

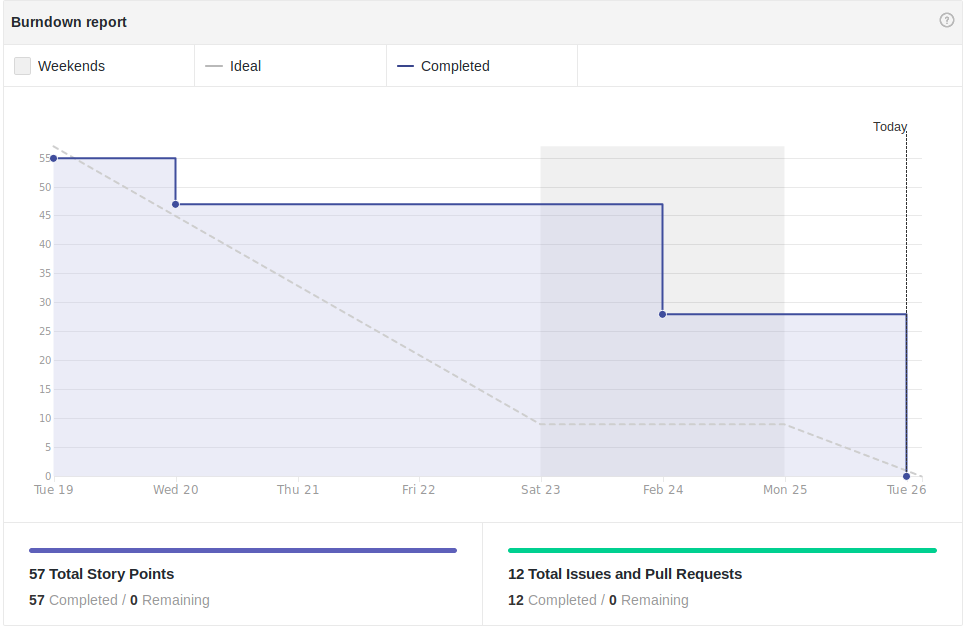


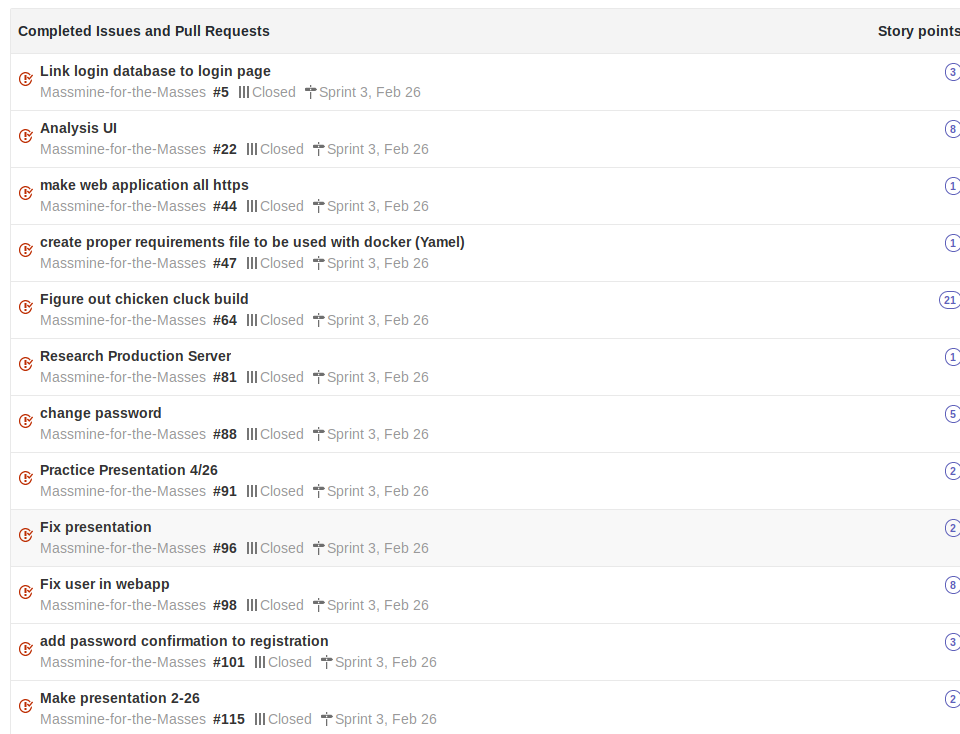


* For scrum Feb. 19:

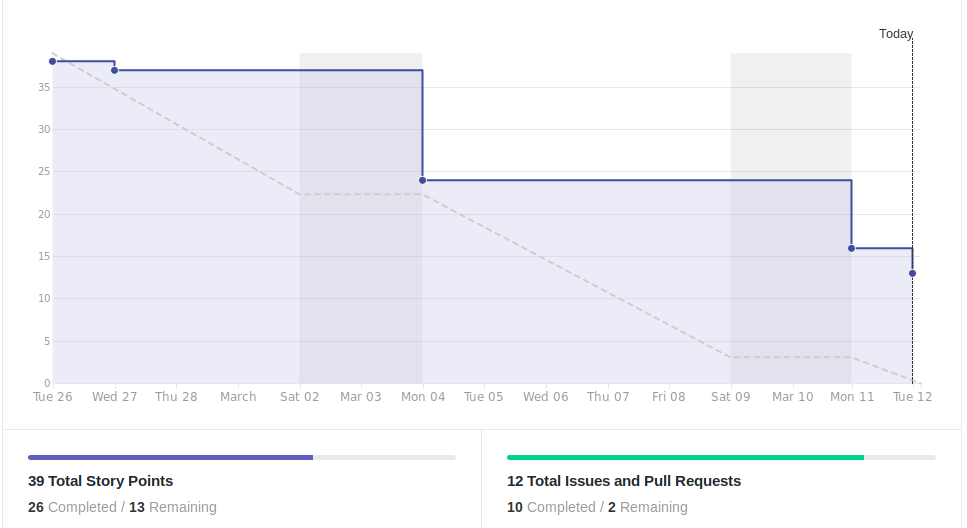


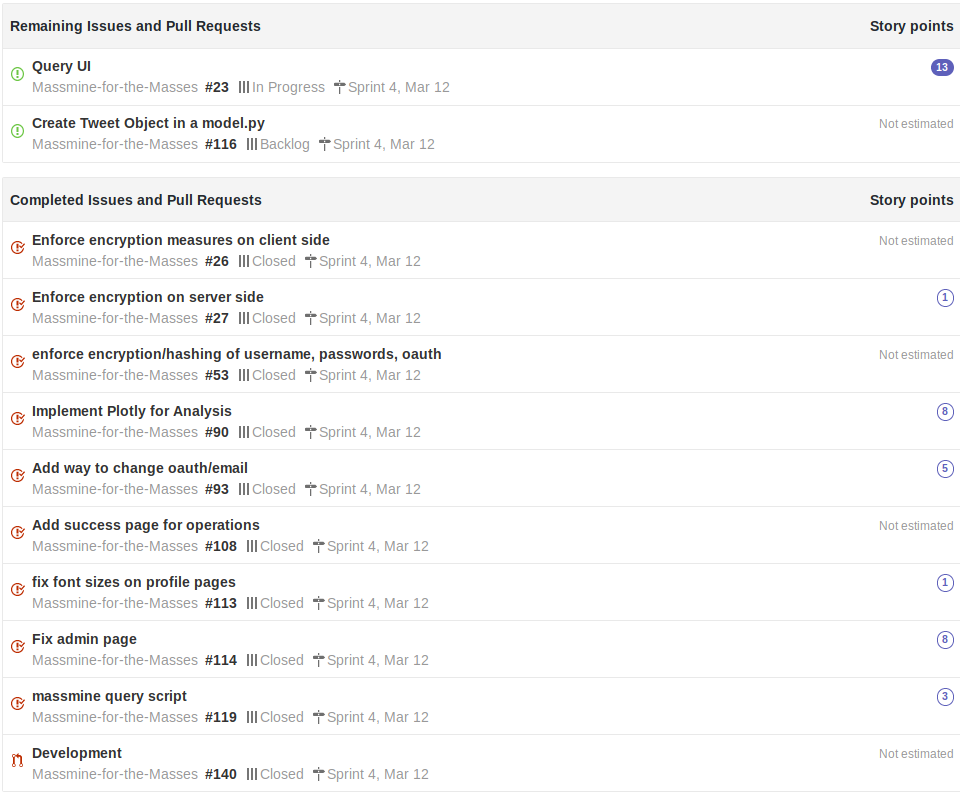
* For scrum Feb. 26:

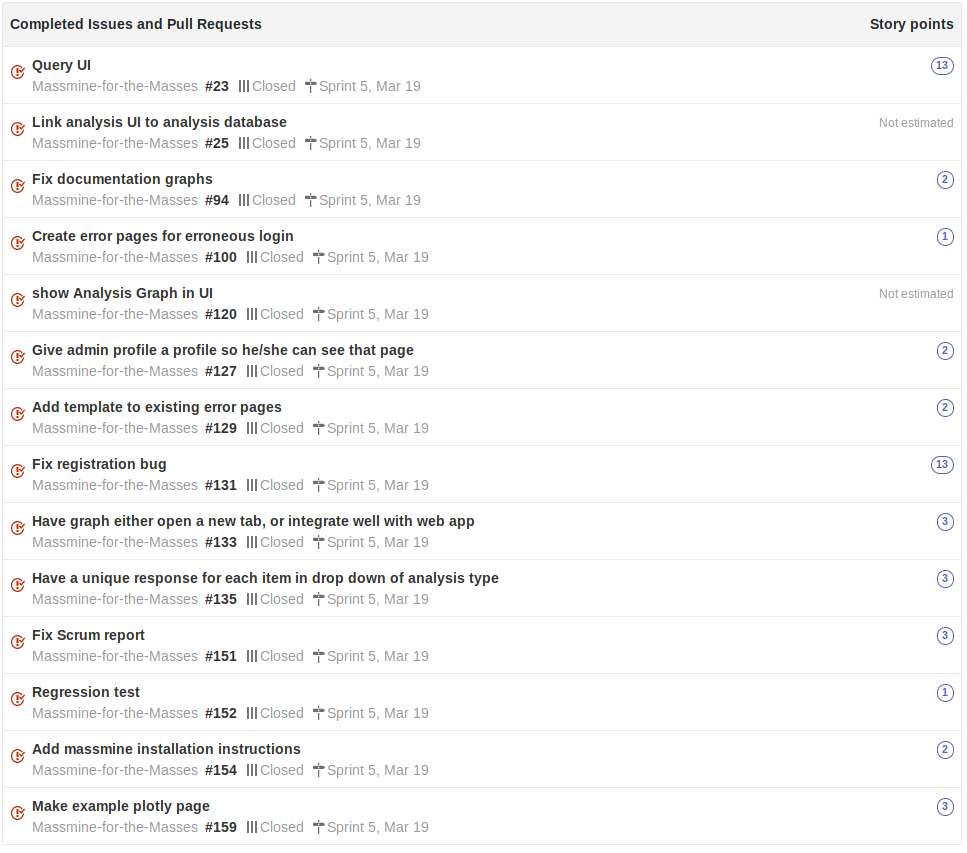
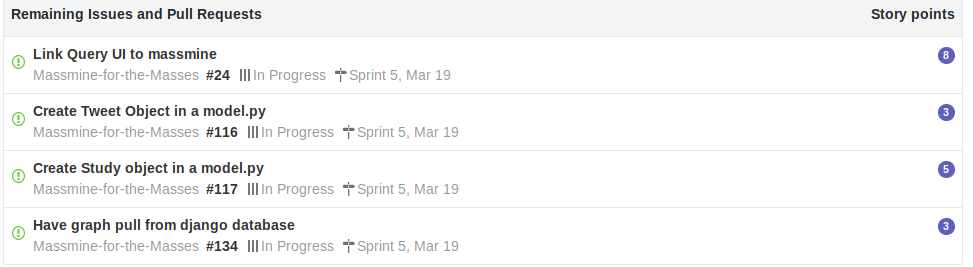
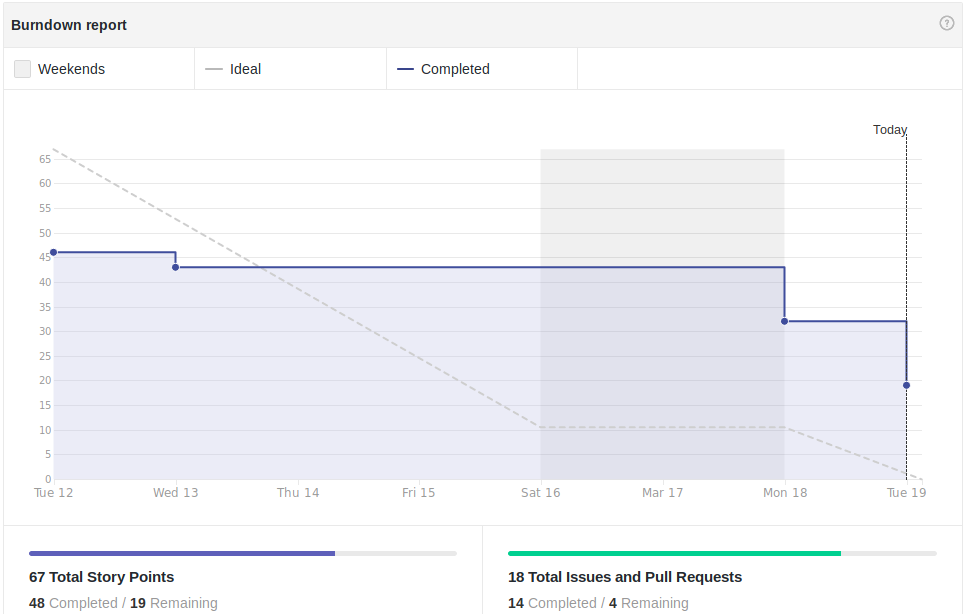


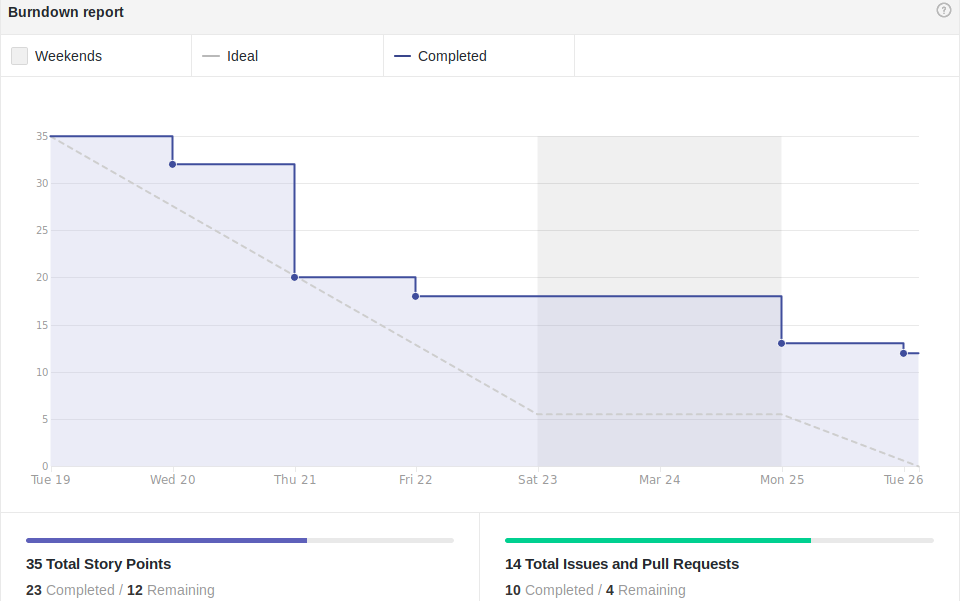


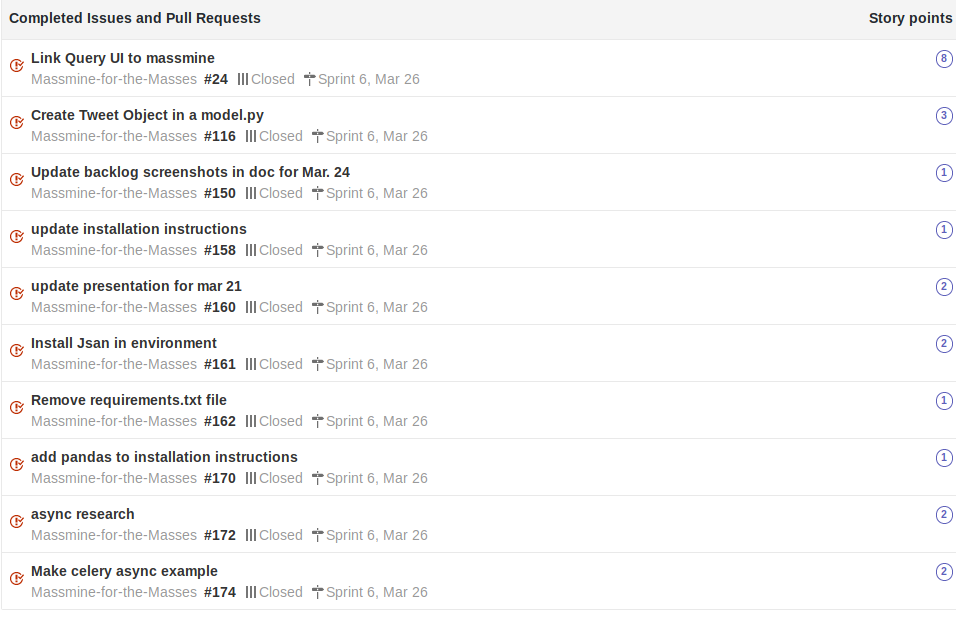
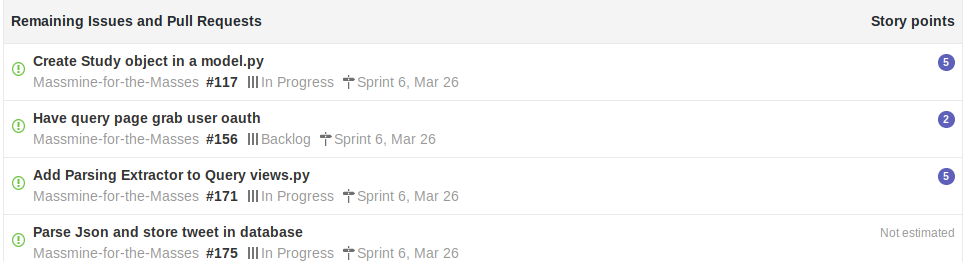
* For scrum Mar. 12



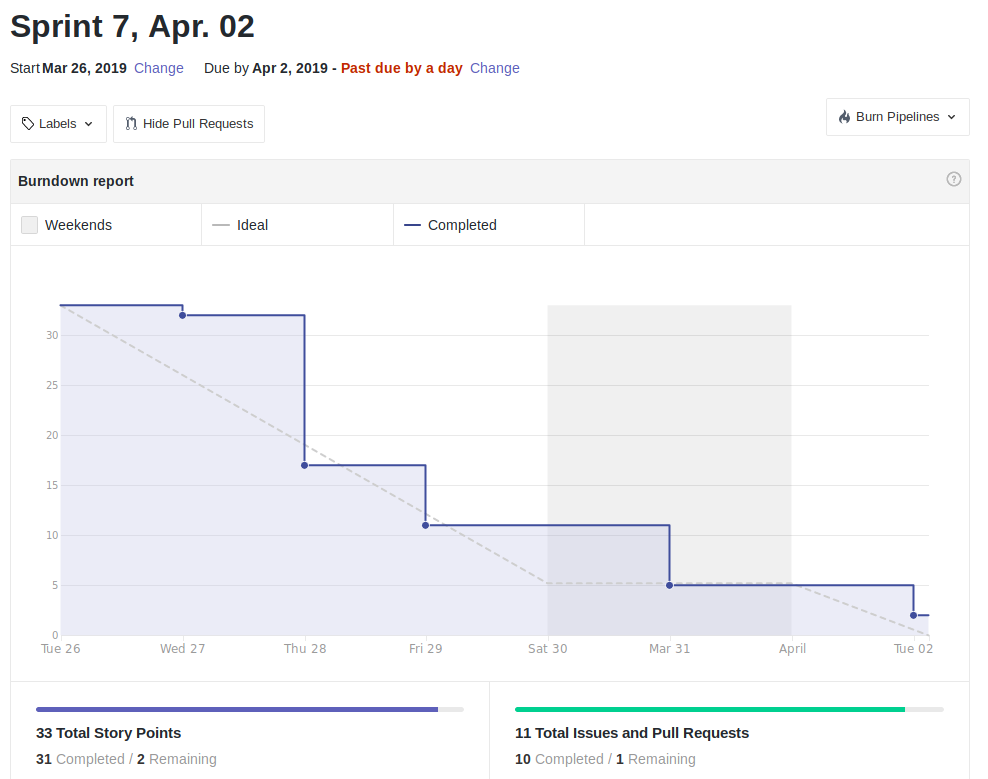
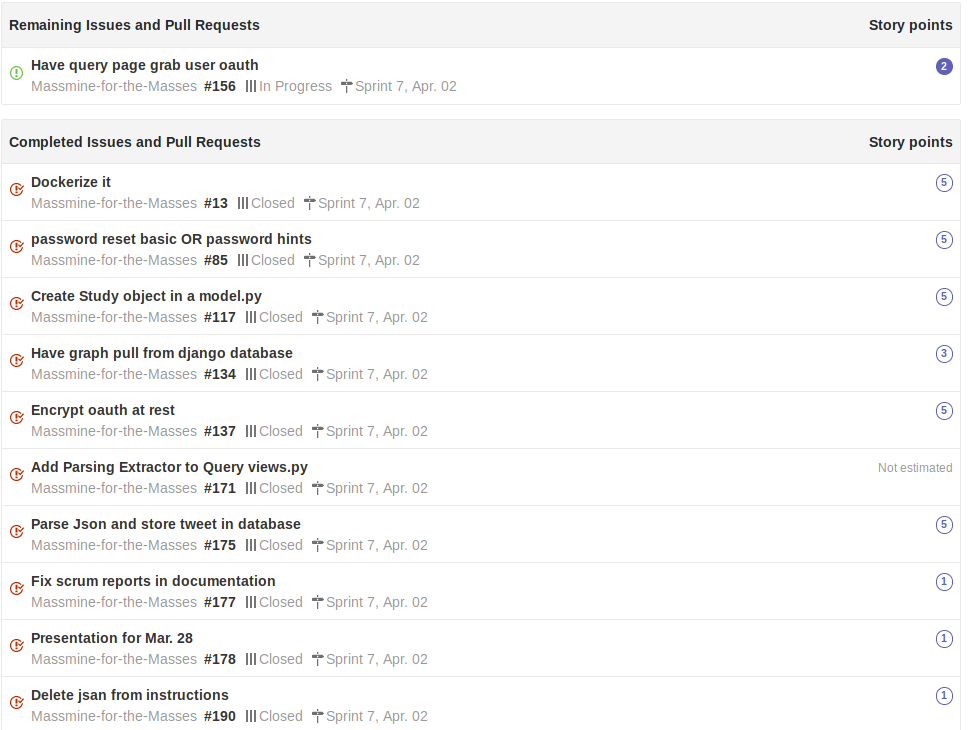


* For scrum. Mar. 19: 
* For Mar. 26:





* For Apr. 02:

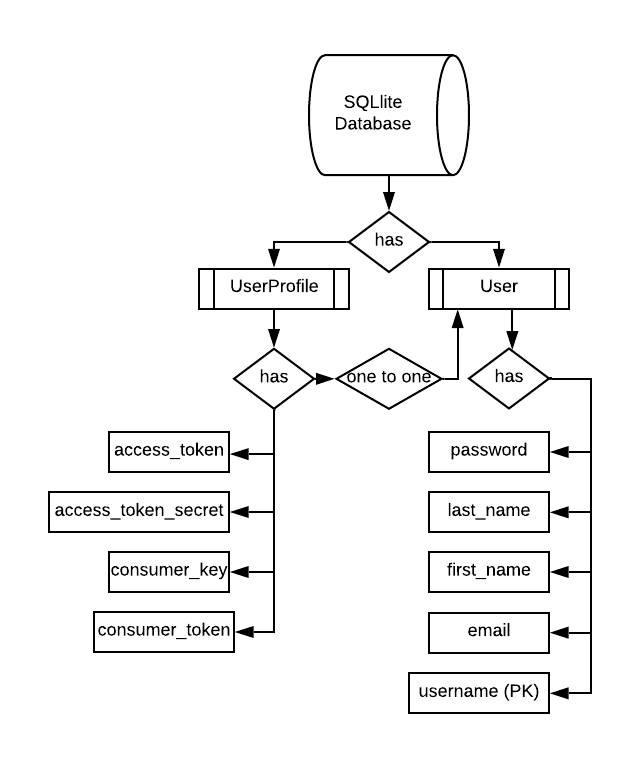
* For Apr. 09:

# **7. Subsystems**

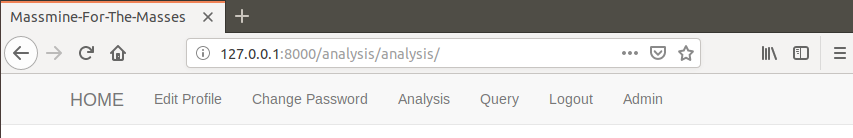
## 

## **7.1 Subsystem 1** – User Management - *Patti*

* Initial design
  + Illustration
    - Database design:



* + - UI design:
      * Using a white and grey bootstrap css: <https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css>
      * Each page is accessible from a navbar at the top of the screen. This navbar works equally well fullscreen or in a smaller window.



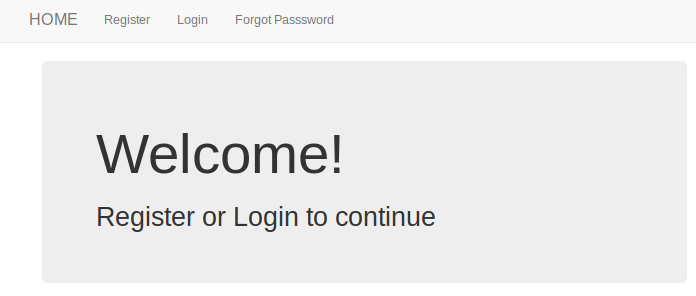
* + Use Cases for User Interface
    - Registration Use Case scenario:
      * Jane would like to create an account on the Massmine-for-the-Masses website. She goes to the desktop where the application is running, and clicks ‘Register’ on the navbar at the top of the webpage. She enters a username (‘Jane’) and email, her password twice, and all four Twitter authorization keys. She leaves the first name and last name fields blank. Then she clicks ‘Submit’ at the bottom of the page, and is returned to the home page. Then she clicks ‘Login’ at the top of the navbar, and enters her username and password on the login screen. This time when she hits submit, she is returned to the Home screen but the message ‘Hello Jane’ appears on the screen and she can see new navbar links, such as Edit Profile, Change Password, Analysis, Query, Logout, and Admin.
    - Change Information Use Case scenario:
      * Jane realizes that she put the wrong numbers in for her Twitter consumer key. She clicks ‘Edit Profile’ on the navbar at the top of the screen, and the screen displays her email address, first name, last name, and all four Twitter authorization keys. She clicks on the consumer key text field, and edits the field to contain the correct information. Then she clicks ‘Submit’ at the bottom of the page, and is returned to the home screen.
    - Change Password Use Case scenario:
      * Jane realizes that ‘janepassword’ is not a very secure password, and wants to change it. She clicks ‘Change Password’ on the navbar at the top of the screen, and the screen prompts her for her old password, a new password, and a confirmation of the new password. It also displays instructions that the password can’t be too similar to her other personal information, must contain at least 8 characters, can’t be a commonly used password, and can’t be entirely numeric. Jane enters ‘janepassword’ as the old password and ‘password1’ as the new password and confirmation, and is given an error page which tells her the new password does not meet the requirements. She clicks on ‘Change Password’ again, and this time enters ‘janepassword’ as the old password and ‘thisisapasswordforjane’ as the new password and password confirmation. This time, she is returned successfully to the home screen.
    - Deleting A User Use Case scenario
      * Bob, an admin in the department, is told that Jane has graduated three years ago and is no longer using the account she made. He logs in to the webpage with his basic user account and clicks on ‘Admin’ on the navbar at the top of the page. He is prompted for an admin username and password, which he enters. He clicks on ‘Users’ under the ‘Authentication and Authorization’ tab, then clicks on the ‘Jane’ username. He scrolls to the bottom of the page, and clicks the large red button labelled ‘Delete’. The page prompts for confirmation, and he clicks ‘Yes, I’m sure’. He is then returned to the administration page, where a notification at the top of the screen says ‘The user ‘Jane’ was deleted successfully. Bob can see that Jane is no longer in the list of users, so he clicks ‘Log Out’ at the top of the page and goes to get coffee.
  + Choices made
    - We chose to add the user’s authorization keys to a second ‘profile’ model that has a primary key to the default user model, rather than extending the user model directly. This allows us to take advantage of some of Django’s built in features for authentication, such as changing a password, without having to recreate too much functionality.
* Data Dictionary
  + User model: taken from <https://docs.djangoproject.com/en/2.1/ref/contrib/auth/>

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Type** |
| username | Name identifying a user | CharField, max length 150. May contain alphanumeric, @, +, . and - characters. Primary key. |
| password | Hashed and salted password to confirm user login | CharField. |
| first\_name | Optional detail if users wish to personalize their accounts. | CharField. 30 characters or fewer. |
| last\_name | Optional detail if users wish to personalize their accounts. | CharField. 150 characters or fewer. |
| email | Optional detail if users wish to personalize their accounts. In future releases could be used to provide notifications and recover password support. | CharField, formatted as email address. |
| is\_staff | Designates whether this user can access the admin site. Admin viewable only. | Boolean. |
| is\_active | Designates whether user is active. May use instead of deleting user. Admin viewable only. | Boolean. |
| is\_superuser | Designates that this user has all permissions. Admin viewable only. | Boolean. |
| last\_login | Datetime of the user’s last login. Admin viewable only. | DateTime |
| date\_join | Datetime of when account was created. Admin viewable only. | DateTime |

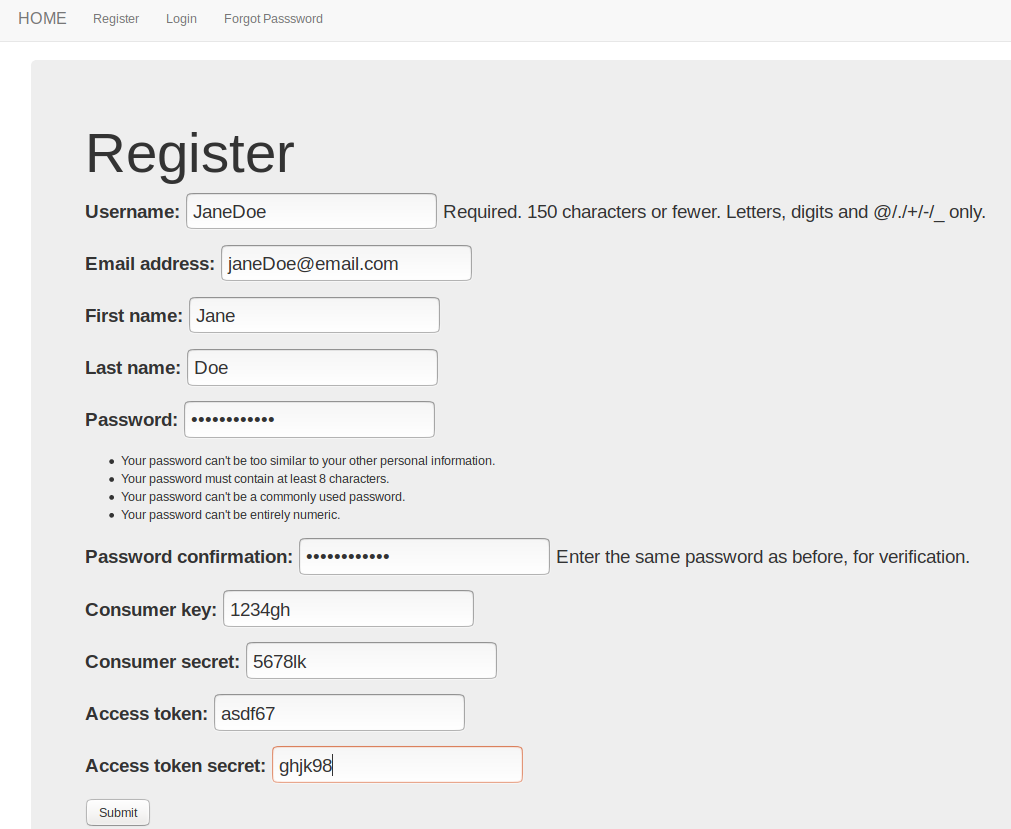
* + User Profile: Created for this project

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Type** |
| User | One-to-One link to user model | Primary Key |
| consumer\_key | Twitter authorization key. | EncryptedCharField, max\_length 50 |
| consumer\_secret | Twitter authorization key | EncryptedCharField, max\_length 50 |
| access\_token | Twitter authorization key | EncryptedCharField, max\_length 50 |
| access\_token\_secret | Twitter authorization key | EncryptedCharField, max\_length 50 |

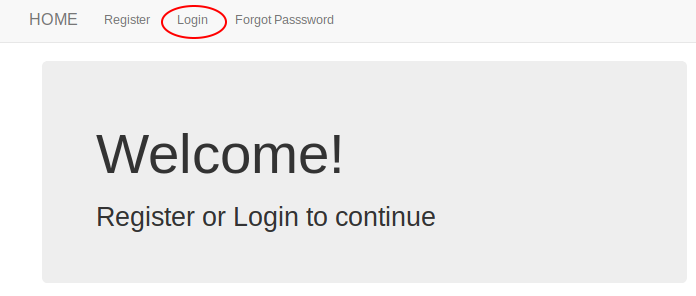
* Refinement
  + No significant changes were made to the user or profile models from the initial design.
* Scrum Backlog
  + Initial Scrum (Feb. 5):
    - Set up Github and issue tracking (1)
    - Make login database (5) (Made irrelevant after project restructuring)
    - Research connecting django and MongoDB (3) (Completed with Josh, made irrelevant by project restructuring)
    - Save Jan. 25 report to Github, documentation (2)
  + Sprint 1 (Feb. 12):
    - System Design Documentation (part 5) (8) (Completed by all members)
    - System Analysis Documentation (part 4) (8) Completed by all members)
    - Update presentation for Feb. 07 (2)
  + Sprint 2 (Feb. 19):
    - User Management pages (no estimate, mistakenly created)
    - Create user page (5) (Completed with Josh)
    - Add oauth attribute to web page (3)
    - Login page crypto/validation(2) (Completed with Josh)
  + Sprint 3 (Feb. 26):
    - Practice Presentation 4/26 (2) (Completed by all members)
    - Make presentation 2-26 (2) (Completed with Morgan)
    - Add password confirmation to registration (3)
    - Change password (5)
    - Link login database to login page (3) (Made irrelevant after project restructuring)
    - Fix user in webapp (8)
  + Sprint 4 (Mar. 12):
    - Fix admin page (8)
    - Fix font sizes on profile pages (1)
    - Add way to change oauth/email (5)
    - Enforce encryption measures client side (no estimate, made irrelevant by project restructuring)
    - Enforce encryption/hashing on username, passwords, oauth (no estimate, mistakenly created)
    - Enforce encryption on server side (no estimate, made irrelevant by project restructuring)
    - Add success page for operations (no estimate, was determined to be unnecessary)
  + Sprint 5 (Mar. 19):
    - Create error pages for erroneous login (1)
    - Add template to existing error pages (1)
    - Give admin a profile so he/she can see that page (2)
    - Regression test (1)
    - Add massmine installation instructions (2)
    - Make example plotly page (3)
    - Fix registration bug (13
    - Fix documentation graphs (3)
    - Fix scrum report (3)
  + Sprint 6 (Mar. 26):
    - Update backlog screenshots in doc (1)
    - Update installation instructions (1)
    - Install Jsan in environment (3)
    - Remove requirements.txt file (1)
    - Add pandas to installation instructions (1)
    - Async research (2)
    - Make celery async example (2)
    - Add parsing extractor to query views.py (5, all members)
    - Update presentation for Mar. 21 (2, all members)
  + Sprint 7 (Apr. 2)
    - Fix scrum reports in documentation (1)
    - Encrypt oauth at rest (5)
    - Presentation for Mar. 28 (1, all members)
  + Sprint 8 (Apr. 9)
    - Subsystem documentation - Patti (5)
    - Take in all 4 Massmine credentials instead of 1 (1)
    - Credential setup script (2)
    - Credential verification (2)
    - Fix main documentation (2)
    - Presentation for Apr. 4 (1, all members)
* Coding
  + Approach
    - The approach taken to this project was object-oriented and based around the Django structure, using model objects to describe data structures, forms to describe how they would be used by other parts of the project, and views to describe their precise interactions.
  + Language
    - Python files provide the basic functionality for Django, with additional HTML files forming the basic templates for the web pages.
* User training
  + Basic Use:
    - How to Register
      * When you navigate to the Massmine-for-the-Masses webpage without being logged in, you should be greeted with a home screen that looks like this:



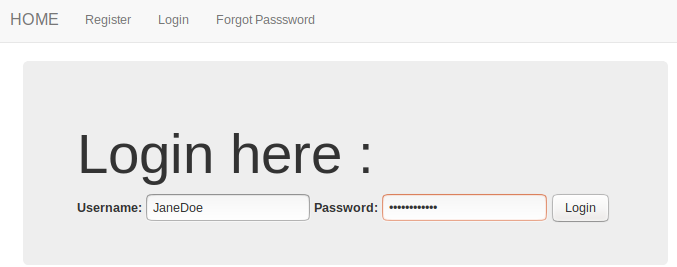
* + - * Click on the button that says ‘Register’ at the top of the screen, and enter your information when prompted. The consumer key, consumer secret, access token, and access token secret, are the four keys from your Twitter developer account found at <https://developer.twitter.com/en/apps> . If you don’t know what your keys are, you can register with some numbers in place instead, but you will not be able to run a massmine query until you have entered valid credentials for these four fields.



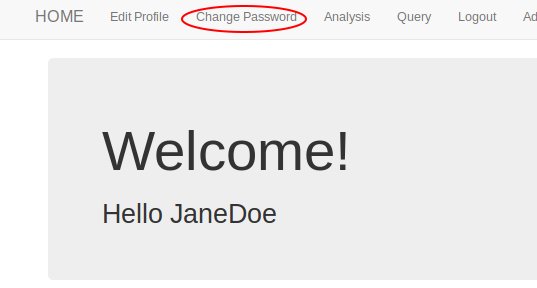
* + - * Once you have entered your information, click ‘Submit’ at the bottom of the page. If your password is incorrect, you will have to enter your password and password confirmation again. If you have bad characters in other fields (such as % or &) you will also have to replace the information in those fields, and will be prompted to do so at the top of the screen.
      * Upon successful registration, you will be redirected to the home page, and you will be able to log in with your new password and username.
    - How to Login
      * From the home screen, click on ‘Login’ at the top of the page.



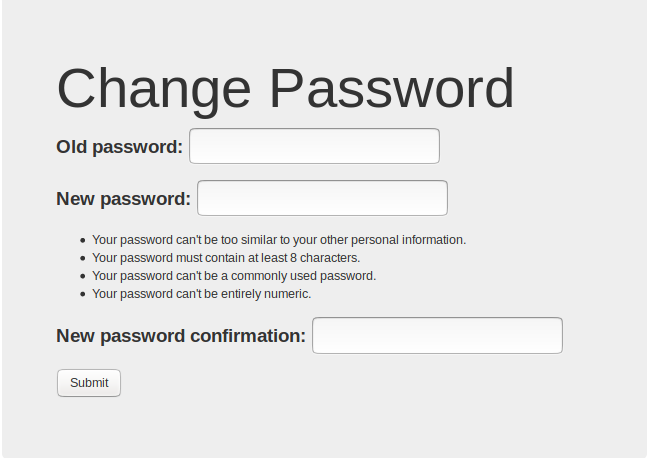
* + - * On this page, enter your username and password.



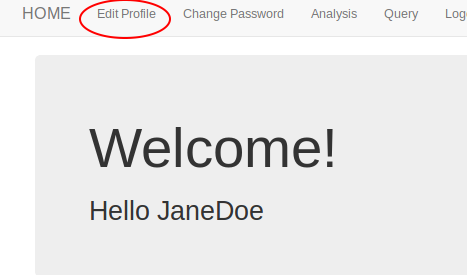
* + - * If you enter incorrect information, you will be directed to an error page. On a successful registration, you will be redirected to the Home page.
    - How to Change Password
      * Once you are logged in, click on the ‘Change Password’ button on the navbar.



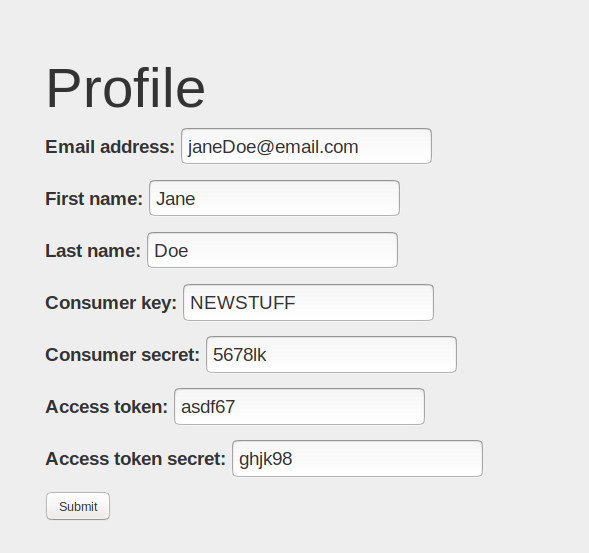
* + - * On this page, you will be prompted for your old password, a new password and the new password again to verify it. If your new passwords don’t match, or do not fulfill the security requirements, you will be directed to an error page.



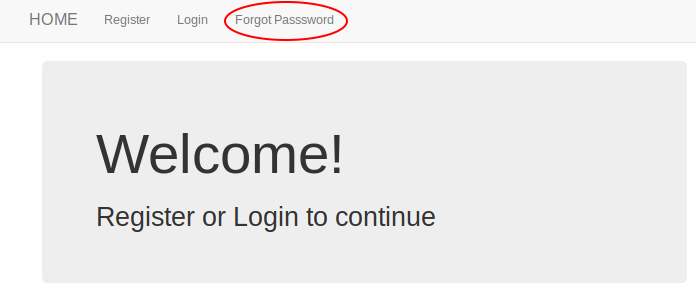
* + - * If you enter the correct old password and your new password matches its verification and meets requirements, you will be redirected to the home page.
    - How to Edit Profile Information
      * To get to the edit profile page, click on ‘Edit Profile’ on the navbar at the top of the screen.



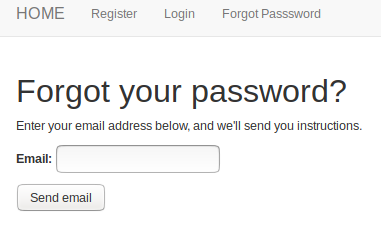
* + - * What information is already in the system will be displayed in the text fields. To edit an item, simply alter the contents of that field and hit submit.



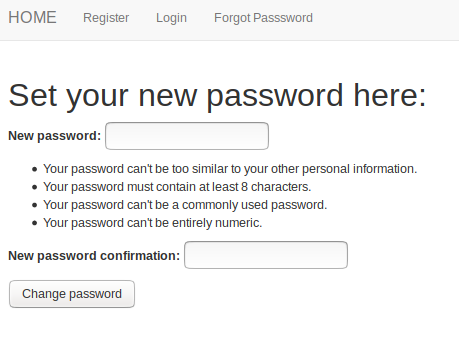
* + - * If you leave a required field blank or have illegal characters, you will be prompted to fix that field. Otherwise, you will be successfully redirected to the Home page.
    - How to use Forgot Password
      * If you are not logged in and do not remember your password, click on the ‘Forgot Password’ button at the top of your screen.



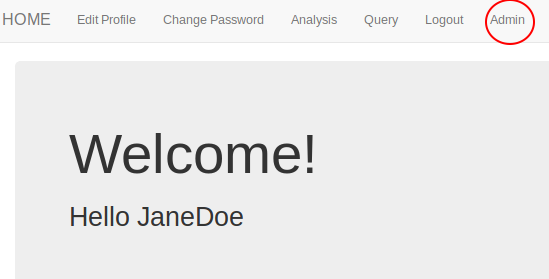
* + - * On the next screen, enter your email address and if that email address exists in the database, you will be sent an email to reset your password. (Note to administrators: If you are using the proof-of-concept version on github, you may be storing these password reset requests as a file in the folder webapp/sent\_emails, not emailing them. Please check the Advanced Use section of this guide for instructions on setting up the Forgot Password functionality.

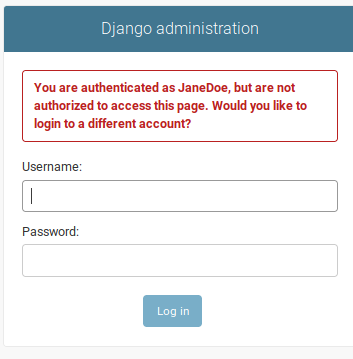


* + - * Once you receive the email, copy and paste the given link into your browser. You will be presented with a page that prompts you for a new password.

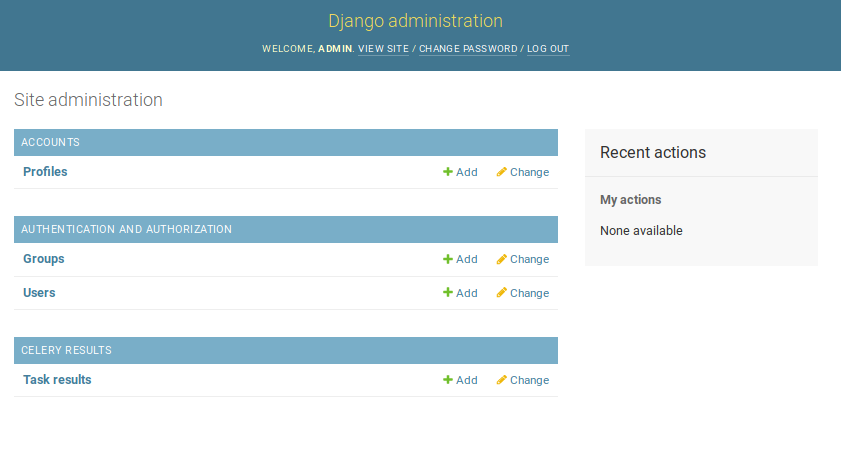


* + - * If your password does not meet the listed requirements, you will be prompted to change it. On a successful password change, you will be prompted to log in with your new password
  + Administrator Use:
    - How to log in as Administrator
      * In order to access the Admin page, you will need to log into the regular site first. You may use your administrator credentials or regular credentials to do this if you wish. If you use your regular credentials, you will be prompted to enter admin credentials after clicking on the ‘Admin’ tab.

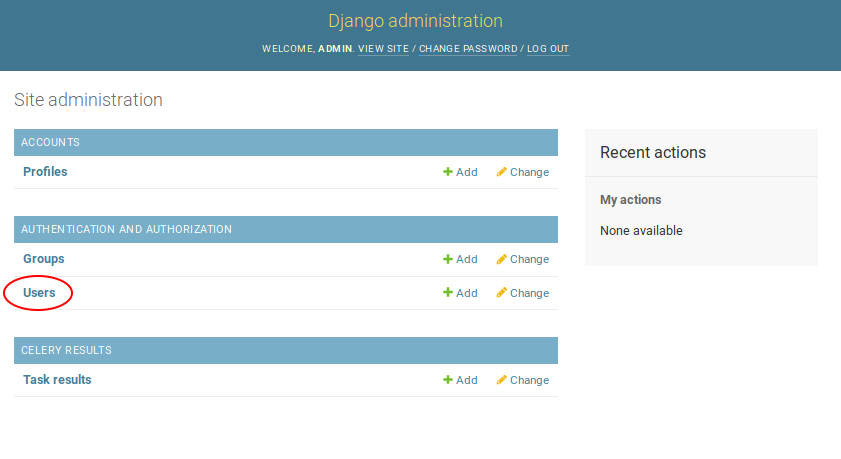


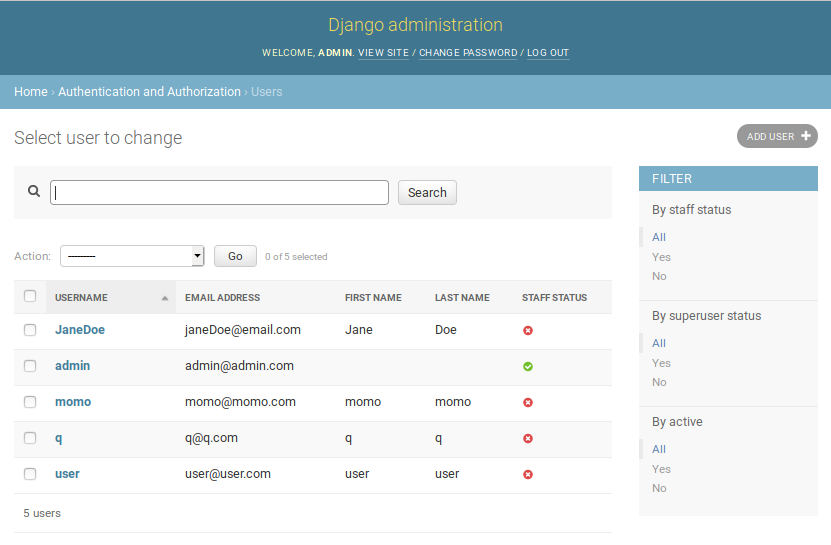


* + - * After you log in with admin credentials, you will be on the administrator site.

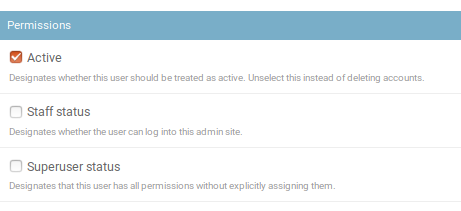


* + - * To return to the main site, click ‘View Site’ at the top of the screen.
    - How to make Administrative Users
      * It may be helpful to have more than one administrator account. To make an existing user account and administrator, login to the administrator site and click on ‘Users’.

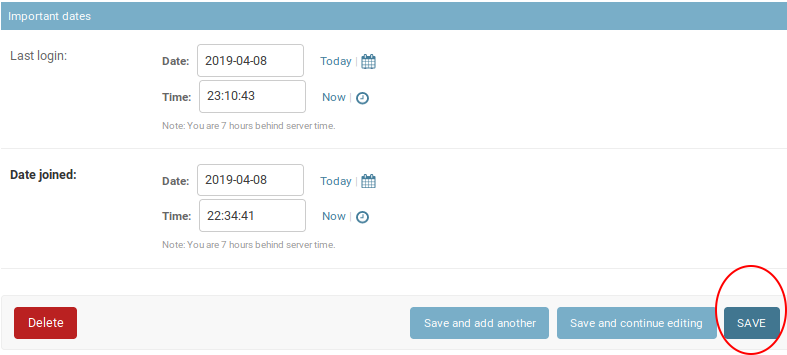




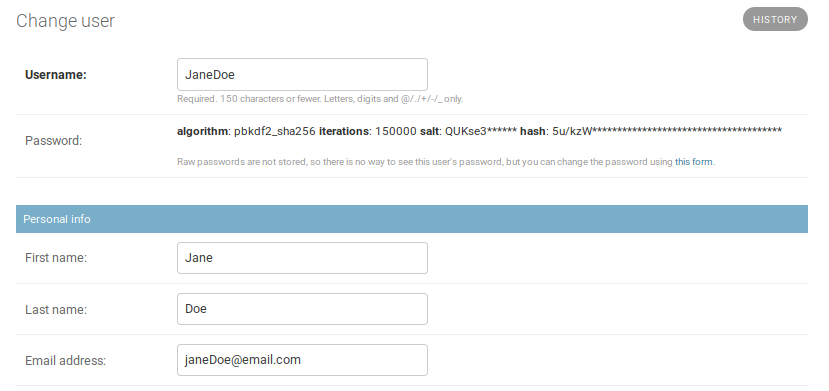
* + - * Next, click on the user you wish to make administrator. Scroll down to permissions, where you can choose ‘staff’ (recommended) or ‘superuser’ (this is complete access to everything).



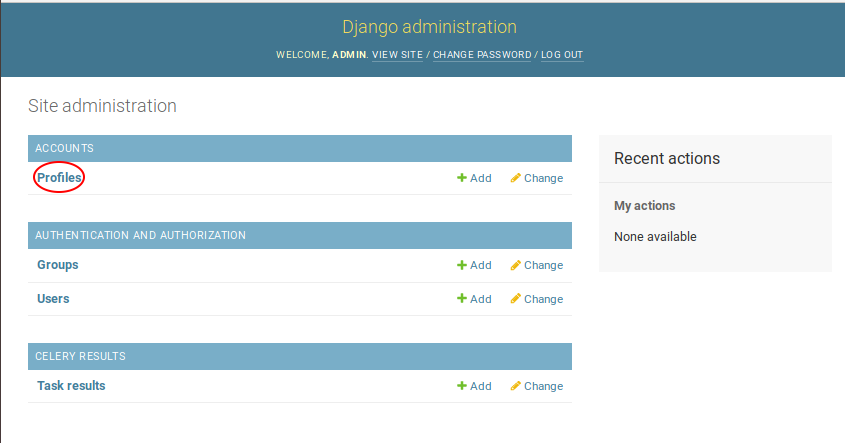
* + - * Once you have selected the desired status, (Do not unselect Active unless you want to make the user inaccessible), just scroll to the bottom of the page and click ‘Save’.



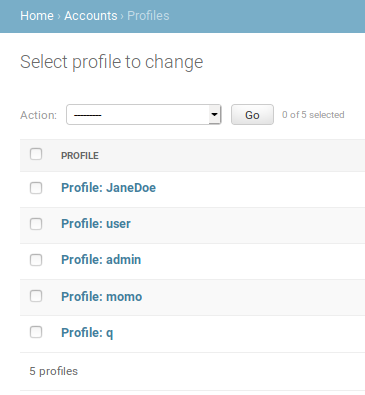
* + - * Congratulations! You have a new administrator.
    - How to delete or edit users
      * To edit or delete a user, log in to the administrator site as in the instructions above, then click on ‘Users’ under the Accounts and Authorization section. Then, click on the user you wish to edit or delete.
      * If you wish to edit the user, you can change the user’s first name, last name, and email address on that page. A link next the password allows you to reset that as well. DO NOT RESET THE USERNAME. It is editable, but changing it here will not change the user ID on associated tweet studies and may have unexpected and undesired effects.



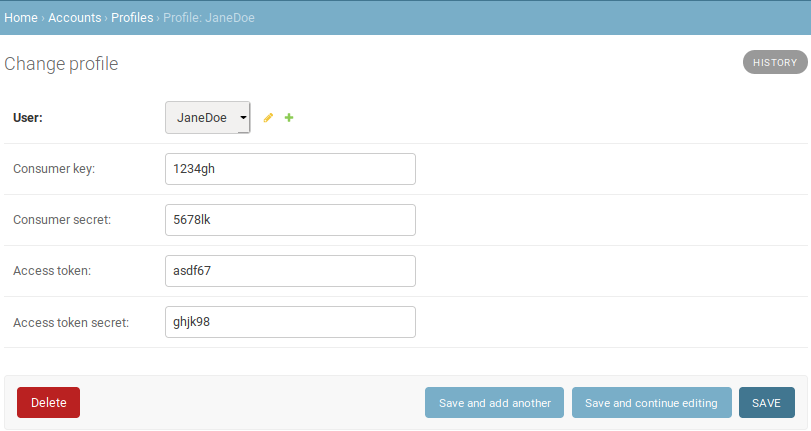
* + - * Once you have made your desired changes, scroll to the bottom of the page. On the left, click ‘Delete’ to remove the user. On the right, click ‘Save’ to save edited information.
    - How to delete or edit user profiles
      * To edit a user’s profile, login to the administrator page as instructed above, and click on ‘Profiles’ under the Accounts section.



* + - * On the next page, click on the user you wish to edit.



* + - * These profiles contain the user’s Twitter authentication credentials, which you can edit or delete as appropriate.



* + Advanced Use:
    - How to Compile Project From Source
    - Setting up Your Own Server
    - Setting up the Forgot Password Functionality
    - Creating Administrative Users from the Command Line
* Testing
  + Initial testing was manually done on a per-item basis, using regression testing on each successive sprint. Additional testing was provided on an as-needed basis by other team members.

## **7.2 Subsystem 2** – Analysis - *Morgan*

* 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** – Query - *Logan*

* 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** – Massmine Integration - *Josh*

* 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: <https://github.com/JoshCMoore/Massmine-for-the-Masses>
* Evaluation by client and instructor
* Team Member Descriptions

## 