# TweetBot Guide

## Requirements

This project is done using pipenv where all the required packages are locked (in Pipfile.lock) and by running a simple command, the required packages will be installed. Note that the pipenv environment is not the same as your machine's environment. Packages installed in your machine may not be in the pipenv and vice versa. Having said that, the packages needed are:

- Python (v3 and above)
- Django
- Twython

Development are done on local machine. If there is a wish to publish the app onto Heroku, The packages needed are:

- Heroku
- Git
- Gunicorn
- Whitenoise
-

Read more / Links:

- Python (development language)
- Django (python front-end framework)
- Twython (twitter api package that works with python)
- Heroku (Deployment Platform)

*This guide assumes you are proficient with git and will skip the git part.

## File Structure

Root (only edit Pipfile and Profile when needed)
db.sqlite3 – the database file
Pipfile, Pipfile.lock – the package manager file which tells pipenv what packages to install
manage.py – Django's code entry
Procfile – configuration for Heroku's deployment

TweetBotWeb (Django's root folder)
settings.py – Django's settings file
urls.py – Django's router configuration file
tweetbot.py – the tweetbot code from the first project which is edited to modularize
tweetbot.log – the log file logged from tweetbot.py
wsgi.py – configuration for Django which is NOT to be edited
legacy/ – this zip folder contains the first tweetbot version written

tweetmodel folder (most of the database structure)
admin.py – write the database that is to be shown in admin
models.py – the model structure of the Note and History database
urls.py – router configuration for tweetmodel

views.py – the views to be rendered for tweetmodel

botapp folder (most of the logic)
admin.py – write the database that is to be shown in admin
models.py – the model structure of the TwitterProfile, Schedule, and Topic database
urls.py – router configuration for botapp
views.py – the views to be rendered for botapp

templates / static – the two folders containing the html, css, and js used for the project

## Database

tweetmodel.model.Note
note – the note to be tweeted out
link – include the link if needed
topic – the note's topic
count – the number of times the note has been tweeted

tweetmodel.model.History (not editable from the front-end)
note_id – the note tweeted out
topic – the note's topic
timestamp – when the note was tweeted

botapp.model.TwitterProfile (not editable from the front-end for security reason)
consumer_token – the twitter app's key
consumer_secret – the twitter app's secret
access_token – the user's twitter token
access_secret – the user's twitter secret

botapp.model.Topic
title – the topic's name / tag
rank – the rank of the topic to identify which note to tweet.

botapp.model.Schedule (which days are scheduled for a tweet)

## Tweetbot.py (api) – Lists the two methods needed for front-end

run()
Goes through the database to find which note to tweet randomly with weight on the topic considered. The note is then tweeted out using the twitter account provided.
Returns a message upon completion.

tweet_note(note_id)
Tweets out the note of choice based on the id given and then updates the history and note database.
Returns a message upon completion.

## Starting up

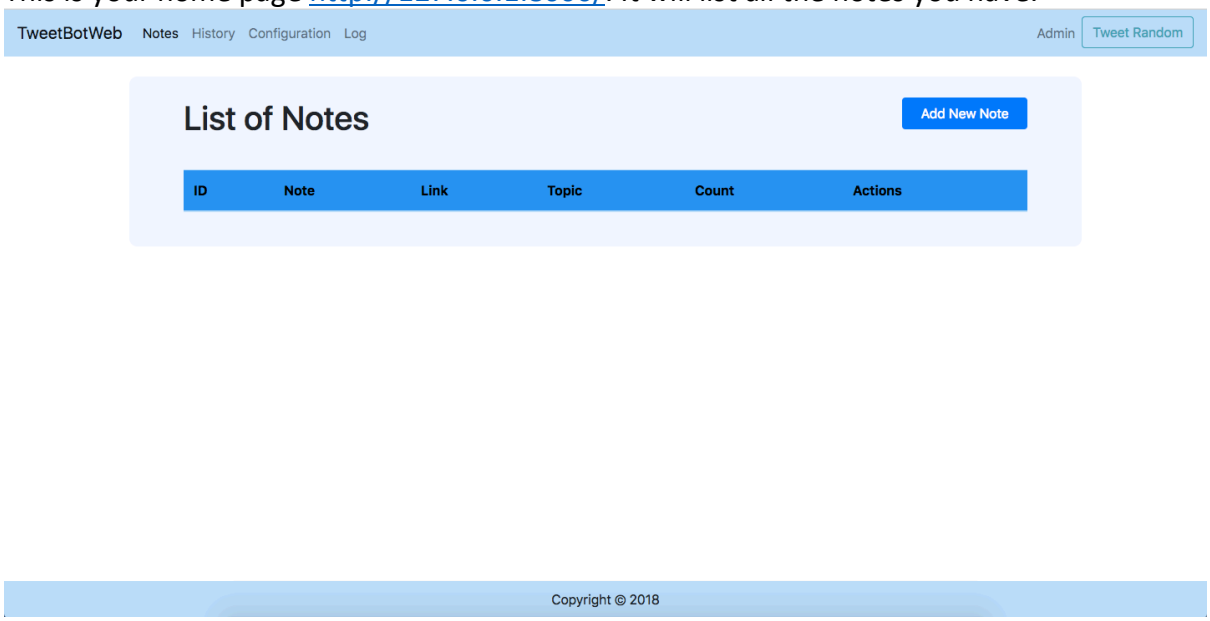
Starting up on local machine for OS X. Commands may differ for windows but the action should be similar.

1. cd TweetBot
2. pipenv shell
3. pipenv install
4. python3 manage.py makemigrations
5. python3 manage.py migrate
6. python3 manage.py createsuperuser
   enter your desired username, email, and password
7. python3 manage.py runserver
8. Go to your browser and open your localhost http://127.0.0.1:8000/

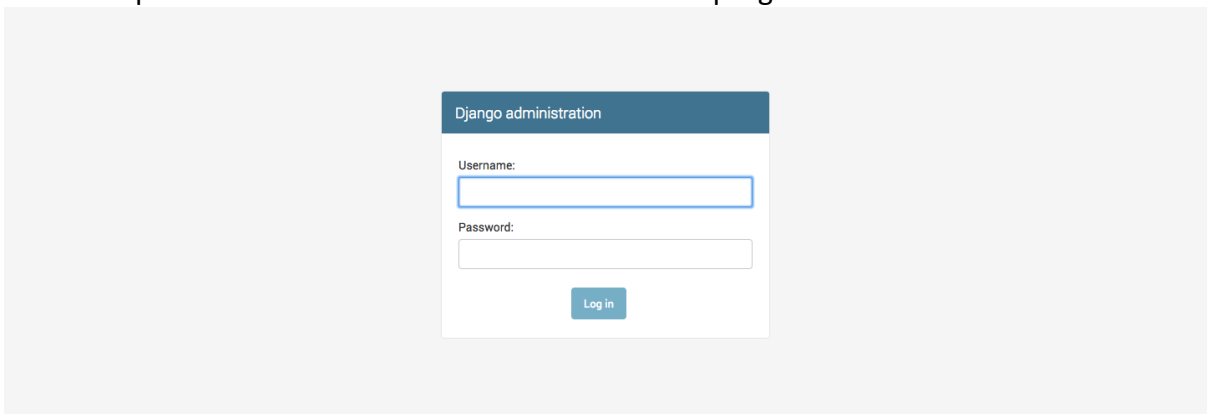
## Pages

Let's set up some stuff first before we begin.

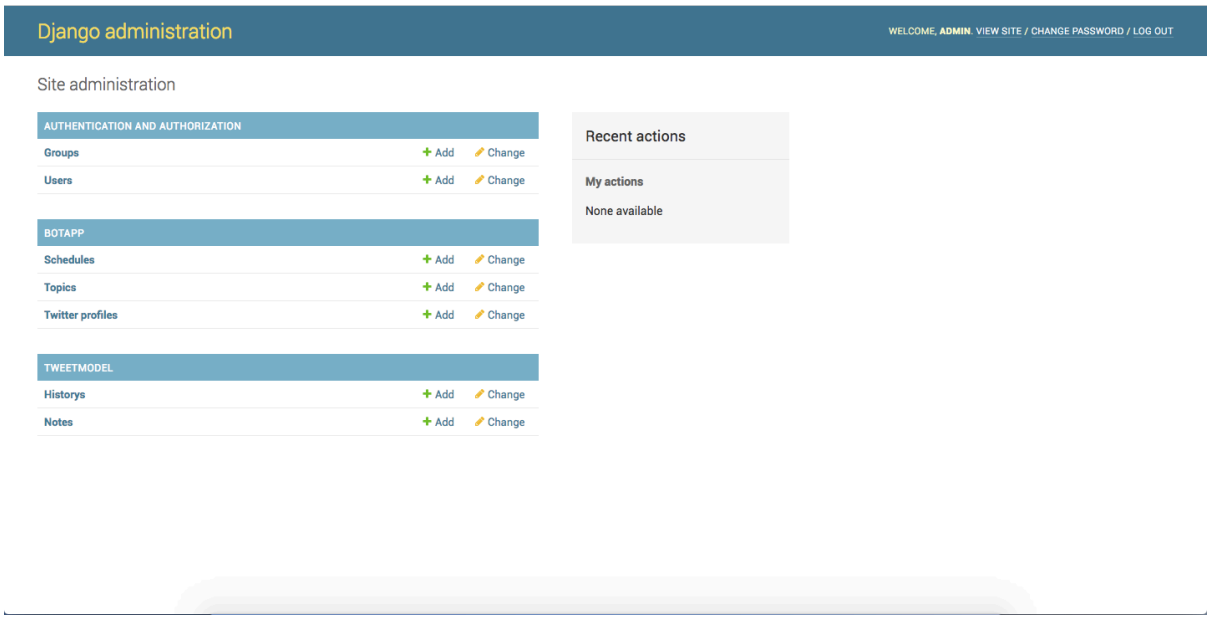
This is your home page http://127.0.0.1:8000/. It will list all the notes you have.



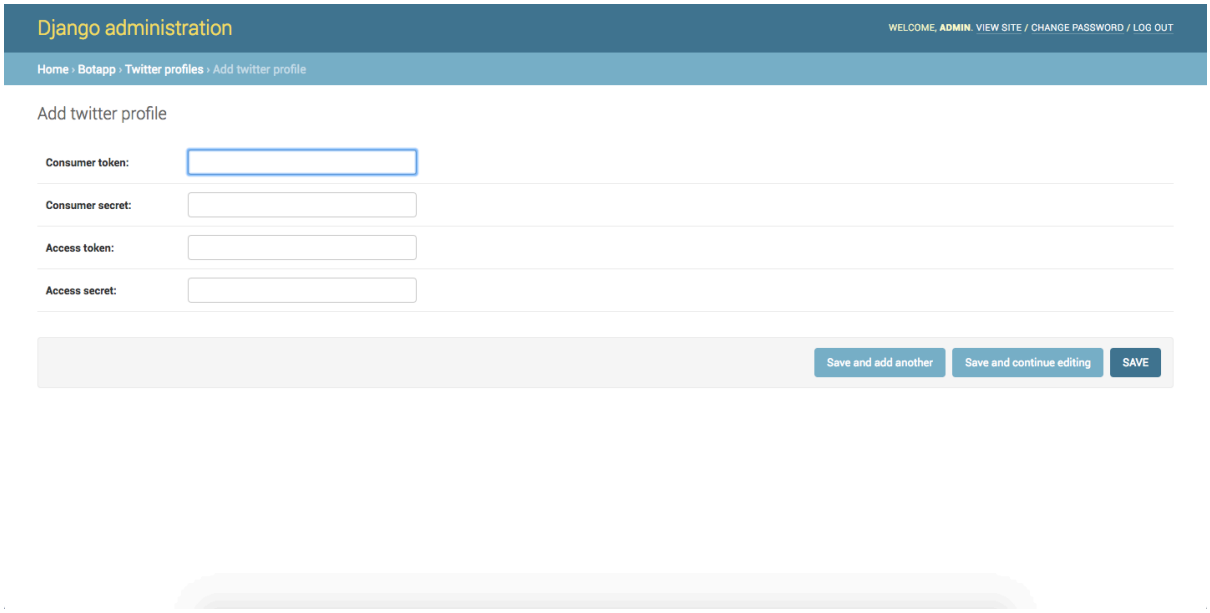

Let's set up some admin stuff. Click on Admin at the top-right corner.




You will reach the Django Administration page where you can modify most of the database here. Sign in to the page with the superuser account's credential you created.



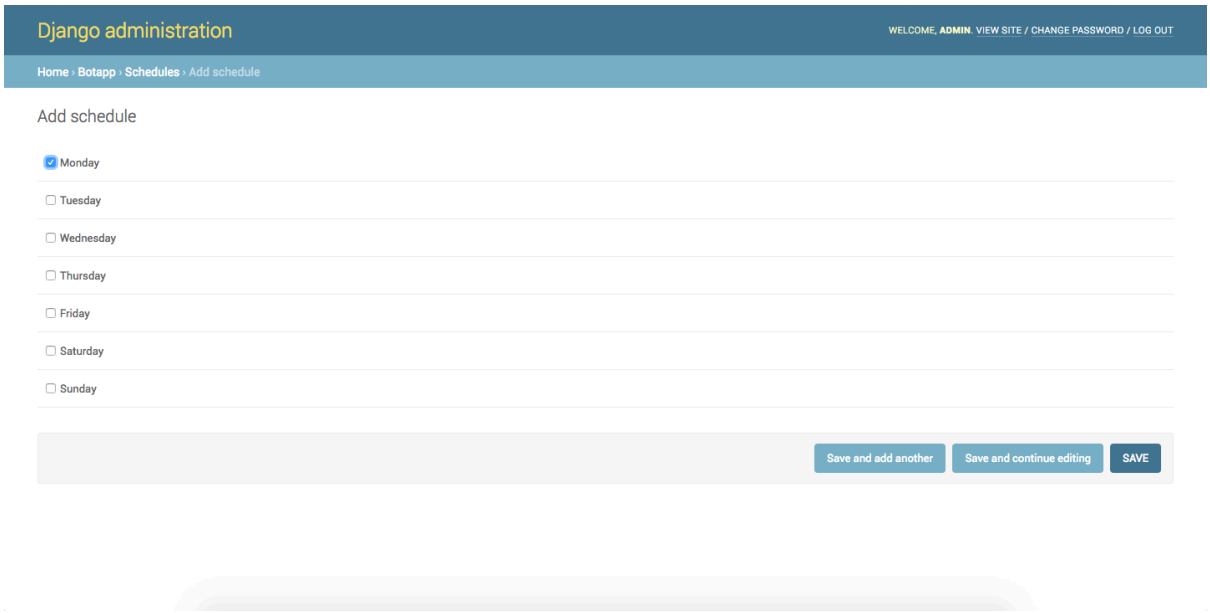

This is the site administration page. Go ahead and click on the “Add” beside Twitter Profile under the botapp table.




Insert the Twitter botapp’s key and secret into Consumer Token and Consumer Secret respectively.
Then, insert your Twitter account’s authentication Token and secret into Access Token and Access Secret respectively.
To find out how to obtain the key tokens and secrets, read this - https://developer.twitter.com/en/docs/basics/authentication/guides/access-tokens

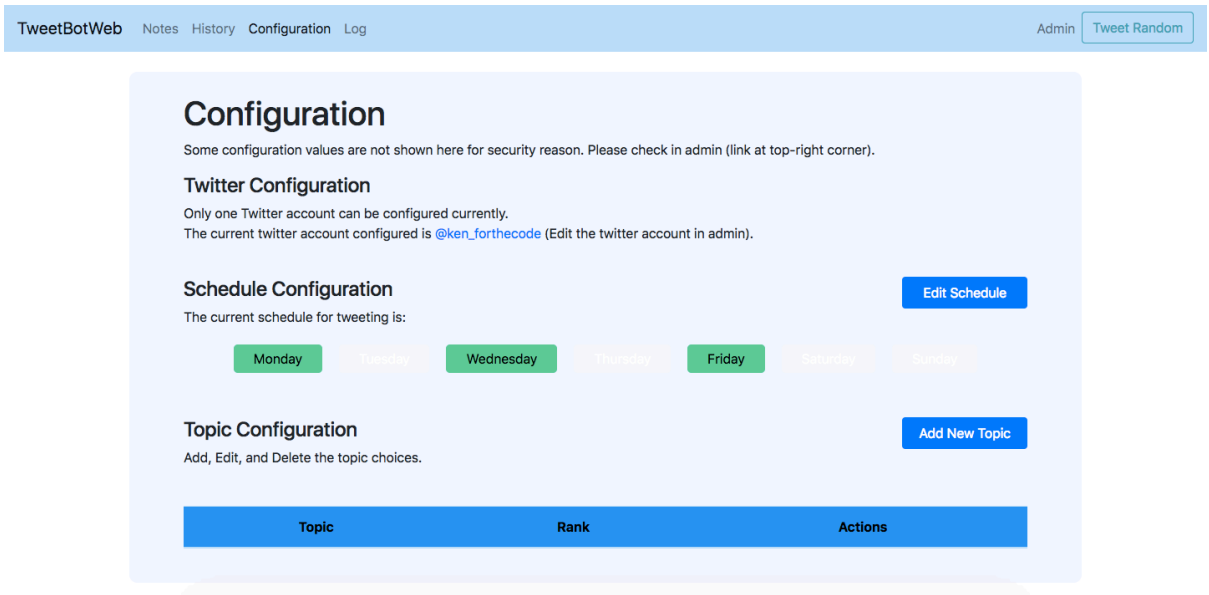
Save this profile.

Go back to the admin homepage and add a Schedule.



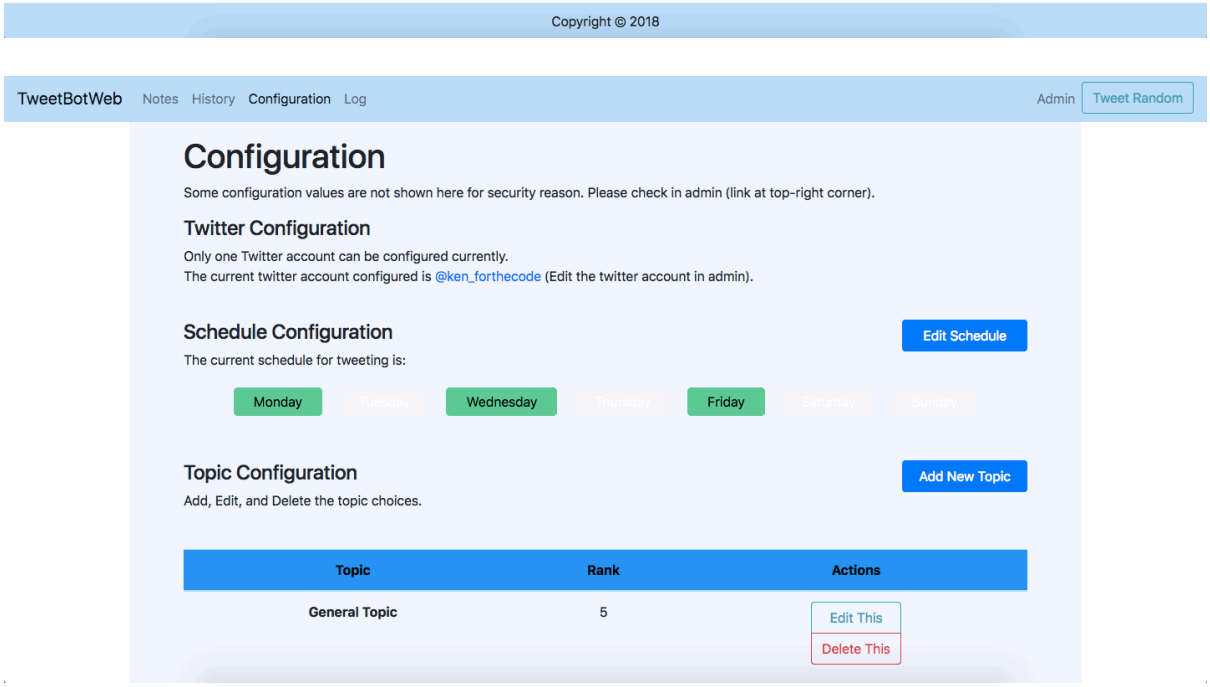

Decide on the days you want the tweetbot to tweet a note and save.

Now, you can go back to the app's page http://127.0.0.1:8000/
You can proceed to go to the app's configuration page http://127.0.0.1:8000/configuration




You will now see your twitter handle, and the highlighted days you have chosen.

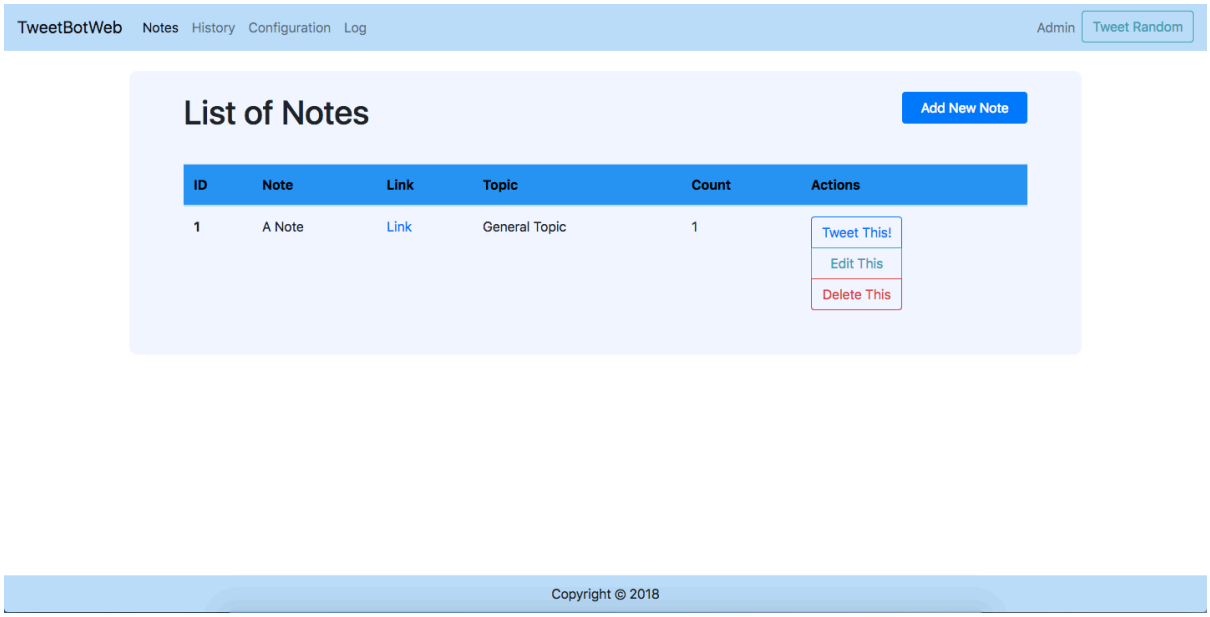
Now, let's add a topic and its rank. This is to let the tweetbot decide which notes in the database should be given more weightage when it chooses a not to randomly tweet.

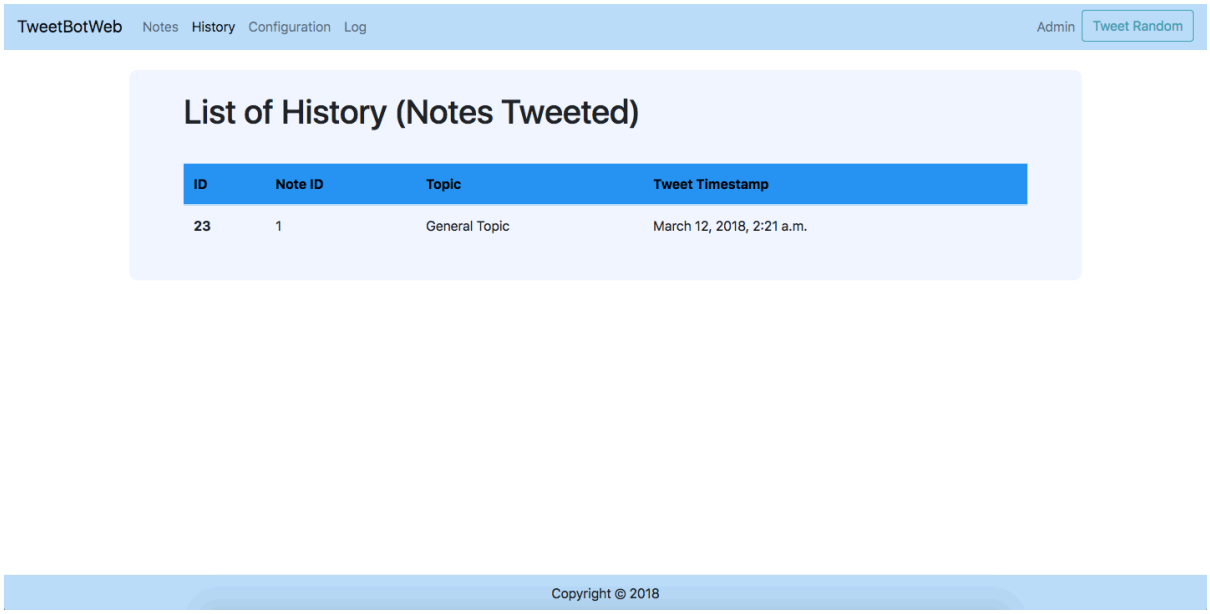

Now that we are done with the configurations, let's check out the other pages.

This is the notes page (http://127.0.0.1:8000/notes) where you will find the notes in the database.

You may Add New Notes, Edit a Note, Delete a Note and even Tweet a Note from this page.

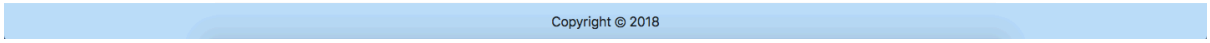



In the history page (http://127.0.0.1:8000/history), you can view all the tweets that was sent out.

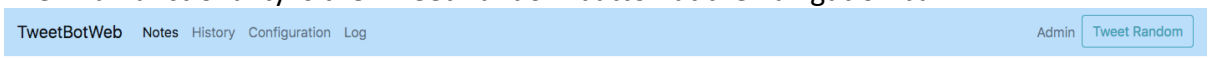



In the logs page (http://127.0.0.1:8000/log), you can view the tweetbot's logs to check if a tweet is sent out or not.

You may download the existing logs or delete the logs to check for ease of debugging.


The final functionality is the Tweet Random button at the navigation bar.




Tweet Random will find a note at random weighted by the rank of the topic to tweet out. It also checks if today's the schedule to tweet before tweeting and will return a "Not scheduled day" message if it isn't.

## Deployment

To deploy to Heroku, you will need a Heroku account. Create one at the website. Then install the heroku-cli into your local machine (not the environment). Quit the server if you are still running it. Then, follow the following instructions while still in the pipenv.

1. You need to commit your changes into git.
2. heroku login
   Login using the credentials you have created.
3. If this is a new deployment, create a heroku app.
   heroku create
4. You will be given an app url and its git similar to this – https://still-thicket-28653.herokuapp.com/ | https://git.heroku.com/still-thicket-28653.git
5. Hook the Heroku app within the git (change the app name as appropriate)
   heroku git:remote -a still-thicket-28653
6. Like a git branch, push it.
   git push heroku master
7. Set the web traffic scale to 1 (heroku's free service)
   heroku ps:scale web=1
8. Verify the web has been deployed
   heroku open

You web app has been deployed! Woohoo. Currently, there is a live version sitting at https://still-thicket-28653.herokuapp.com/

## Enhancement (for future)

Some thoughts have been considered to enhance the usability of the web app.

1. Firstly, the bot is still not a smart bot as it requires a human to press the button to tweet the note. A scheduler was thought to be sufficient to help with scheduling the tweet job. One possible scheduler is APScheduler. However, the implementation is halted because of the technicalities involved which required further thought. For example, we can start the scheduler but we can't stop the scheduler. The implementation of how to schedule from the front-end is also not very clear.
2. Currently, the way that a user can tweet is to provide their user authentication token and secret to the Django database. It can actually be further improved by using Twython's oauth method. However, some concerns to consider is how to route the authentication in Django. Also, as we allow multiple Twitter users, the database structure of the Notes, History, Topic as well as Schedule needs to be re-drawn so that only respective Twitter user can access its own database models.