# retwitter

Postman Service Internship Assignment

Python Django

[Build Status](https://travis-ci.org/sid22/retwitter)

Re Twitter Assignment - complete with all basic, extended and extra credit functionalities.

Postman Collection of all the endpoints with valid request/responses

<https://documenter.getpostman.com/view/3851689/RzZ6HL3A>

The whole APP hosted on Google Cloud

<https://postman-221411.appspot.com>

## Table of contents

* [Installation](#installation)
  + [Setup instructions](#setup-instructions)
* [Basic functionality](#basic-functionality):
  + [User registration using unique username and a password](#signup)
  + [User login, with session maintainance](#login)
* [Extended Functionality](#extended-functionality):
  + [Follow](#follow), [Unfollow](#unfollow)
  + [Create](#create), [Read, delete tweet](#read-delete)
  + [Unit/Integration tests for all endpoints](#tests)
* [Extra Credit](#extra-credit)
  + [Like/unlike a tweet](#like-unlike)
  + [Retweet](#retweet)
  + [Replies](#replies) and [threading](#threads)

The project is a set of APIs made with the Django Framework in Python language. MongoDB database is used for storing information, MLab is used to provide Database-as-a-Service on hosted project. Redis in-memory data structure store is used as a cache store for session management, RedisLabs is used to provide it on hostel project.

|  |
| --- |
| ## Installation [(Back to top)](#table-of-contents) |
| To run the project locally, firstly the user must either: - Set a local environment variable "ENVIRONMENT" to "production" so that the local instance connects to MLab and RedisLabs - Or have a local instance of Mongod Server ( Port: 27017 ) and Redis Server ( 6379 ) running on standard ports. |

### Setup instructions

* Clone the repo https://github.com/sid22/retwitter.git
* Command: cd/retwitter
* Make a virtual environment python3 -m venv env and activate it source env/bin/activate
* Install the dependencies pip install -r requirements.txt
* Run it python manage.py runserver
* (Optional ) To run tests python manage.py test --verbosity 3

## Basic functionality

[(Back to top)](#table-of-contents)

### Signup

User registeration with unique username and password.

No bounds on username or password. Response has a token, which is then used further on to authenticate/authorize the user.

Route: user/signup

Method: POST

### Login

User login with unique username and password.

Response has a token, which is then used further on to authenticate/authorize the user.

Route: user/login

Method: POST

### Logout

User logout with server generated token.

Response has no token only acknowledgement of logout.

Route: user/logout

Method: GET/POST

## Extended functionality

[(Back to top)](#table-of-contents)

Generally all endpoints below require a valid token to be present in Request Header with key as 'Authorization' and token value can be obtained with login/signup.

If empty token, no header the following error response will come, status code 401.

{  
 "Error": "Token not valid, please login to get a valid token first"  
}

### Follow

The user idenitified in the token follows the user passed.

Route: user/follow

Headers: Authorization {{ Token }}

Request Body:

follow\_name: {{ username of user to follow }}

Method:/POST

Will throw error on empty username, if the username is already being followed.

### Unfollow

The user idenitified in the token unfollows the user passed

Route: user/unfollow

Headers: Authorization {{ Token }}

Request Body:

unfollow\_name: {{ username of user to unfollow }}

Method:/POST

Will throw error on empty username, if the username is already being not followed.

### Create

The user idenitified in the token creates a tweet with the text passed

Route: tweet/create

Headers: Authorization {{ Token }}

Request Body:

tweet\_text: {{ text for the tweet }}

Method:/POST

The text will be capped at 140 characters ( including spaces ), same for replies and threads.

Will throw error on empty tweet\_text also.

### Read Delete

The user idenitified in the token is used to check authorization for delete.

Route: tweet/{{tweet\_id}}

Headers: Authorization {{ Token }}

Methods: 1: GET The response will be the tweet info of tweet with the tweet\_id passed

1. DELETE The response will be an affirmation of the deleted tweet if the user is authorized to do it.

Will throw error if user not authorized.

## Tests

I have used primarily python's unittest module and Django's Test Client to test the handler funtions and endpoints respectively.

The coverage reports for the same have also been generated and are present in the /docs folder in the main repo.

The reports are hosted on Github pages and can be seen [Coverage Report](https://sid22.github.io/retwitter/)

The project has also been integrated with Travis CI system to automate testing and keep a build status.

Currently the coverage of tests is 88%.

Travis Builds can be seen [Travis Builds](https://travis-ci.org/sid22/retwitter)

To run all tests locally

python manage.py test --verbosity 3

To run tests for account componenet only

python manage.py test account/ --verbosity 3

To run tests for tweet componenet only

python manage.py test tweets/ --verbosity 3

To run all tests and fresh generate a coverage report ( in /docs folder )

python getcoverage.py

## Extra Credit

[(Back to top)](#table-of-contents)

### Like Unlike

The user idenitified in the token 'likes' the tweet whose id is passed. If the user has previously liked the same tweet, he will 'unlike' it. Hitting the endpoint again will again make the user 'like' the tweet.

Route: tweet/emotion/{{tweet\_id}}

Headers: Authorization {{ Token }}

Method:/POST

### Retweet

The user idenitified in the token 'retweets' the tweet whose id is passed. A retweet will have similar model as a tweet, but will also have certain flags like

is\_retweet: True

The original tweet which has been retweeted will also update with a retweet\_count increase, retweet\_users list will have id of user who has retweeted the tweet and list of id's of all retweets.

Deleting a retweet will correspondingly update the fields of original tweet also.

Route: tweet/retweet/{{tweet\_id}}

Headers: Authorization {{ Token }}

Method:/POST™

### Replies

The user idenitified in the token 'replies' to the tweet whose id is passed. A reply will be treated similar to a tweet object. It will the following flag.

is\_reply: True

The original tweet which has been replied to will also update with a replies\_count increase and the replies list of id's of all replies.

Deleting a retweet will correspondingly update the fields of original tweet also.

Route: tweet/reply/{{tweet\_id}}

Headers: Authorization {{ Token }}

Request Body:

reply\_text: {{ text for the tweet }}

Method:/POST

### Threads

The user idenitified in the token creates a thread of tweets. The tweets is a thread will have

"is\_threaded": True

Each 'thread' will also be given a thread id linking all tweets.

Route: tweet/thread/

Headers: Authorization {{ Token }}

Request Body:

thread\_count: {{ int value of number of tweets }},  
tweet\_text\_1: {{ text 1 }},  
--  
--  
tweet\_text\_i: {{ text i }}

Method:/POST

Response will have the thread\_id and list of all tweets in the thread.

To view a thread as one with all tweets:

Route: tweet/thread/{{ thread\_id }}

Headers: Authorization {{ Token }}

Method:/POST