Project Report

This project aims to design and implement a document store using MongoDB, providing a scalable and robust system for managing tweet data. The system facilitates a variety of operations such as data insertion, search and retrieval, along with user and tweet analysis.

In this document we will go through the following:
I) General Overview (& User Guide)
II) Details of Algorithms Used
III) Testing Strategy
IV) Source Code Quality

I) General Overview (& User Guide)

The application is a command-line tool designed to manage tweets and search for both tweets and users, using a MongoDB database, accessible via Python. It integrates MongoDB queries with Python functions to search tweets and users based on either keywords or top statistics.

User Guide:

i) Getting started

- Ensure Python, Pymongo, and MongoDB are installed on your machine
- Clone or download the project files
- Open the terminal and navigate to the directory
- Add the JSON file for the database to the Functions directory
- Run the command python3 load-json.py <JSON file name> <port number> to set the port and JSON file to be used
- To run the program, enter the command python3 main.py <port number>

ii) Functionalities

- To search for a tweet, enter the corresponding number to the function, then enter keywords to search for the tweets, separated by a space.
    - To search for tweets containing certain hashtag(s), make sure to include the "#" symbol before each hashtag term
    - To search for tweets containing certain words, do not include the "#" symbol
    - Keywords are case insensitive
    - To see all fields of a tweet, enter the corresponding number of the tweet
- To search for a user, enter the corresponding number to the function, then enter an identifying alphanumeric keyword. This keyword should be contained in the display name or location of the user you are searching for and is case insensitive.
    - To see all fields for a user, enter their username
- To list the top tweets of retweets, likes, or quotes, enter the corresponding number to the function, then enter the corresponding field and the number of tweets you wish to see
    - To see all fields of a tweet, enter the corresponding number

- To list the top users based on followers, enter the corresponding number and type the number of users you want to see
    - To see all information of a user, select the corresponding number
- To compose a tweet, enter the corresponding number and enter your tweet

iii) Logging off

- To exit the program, enter the corresponding number

II) Details of Algorithms Used

The software uses multiple algorithms to complete the necessary functions for searching and inserting:

- **load-json.py** - This file uses an algorithm to insert large amounts of data into the database at a time. It inserts batches of up to 10000 lines at a time to the database.
- **Searching** - The functionalities use an algorithm to search the database for matching tweets or users with keywords with a mongoDB query. It searches through every entry once to prevent duplication of results.
- **Getting top** - The functionalities use an algorithm to get the top n users or tweets using mongoDB querying. They get all the users or tweets, then sort descending by field, and afterwards return the results in order, to a limit of n. This prevents returning more tweets or users than exist in the database.
- **Seeing all fields** - This functionality queries the database for the matching user or tweet based on the unique id.

These algorithms run at most in *O(n)* time, or linearly, so our software would scale linearly with large databases.

III) Testing Strategy

We followed the descriptions of the various functionalities as explained in the assignment overview. We also thought of potential edge cases outside of what is prescribed and tested these. To test the potential test cases, we ran the program as a user and used MongoDB queries to compare.

IV) Source Code Quality

The code is well commented and well documented. It is made using a multi-file system for better readability and modularity.

Our group divided the various functionalities among ourselves along with the project report. We held a meeting to test the software and all contributed to testing.