Index Instigators Design Document - Mini Project 2 - CMPUT 291 - Fall 2023

a) General Overview & User Guide

This program acts similar to a social media platform, giving users the ability to search for tweets and users, see top tweets, compose tweets, and more. At the beginning, the user is presented with a main menu from which they can select a certain option to perform further operations. Each of these options is detailed in the software design. The user may choose any of the options given, and they will enter the menus of the respective functionalities each time they select an option from the main menu. A user can select to search for tweets, bringing them to the search for tweets menu; they can select to search for users, bringing them to the search for users menu; they can list top tweets or users, bringing them to the respective menu; they can also compose a tweet, inserting an entry into the database. At the end, the user has to manually exit the program by following the provided instructions. A flow chart detailing how the user can navigate the program is provided in an appendix at the end of this report.

b) Software Design

i) Phase 1

In this part, our program will run the load_json.py with .json file and port number as input. We also use the Pymongo module to connect to a Mongo client. We create a database “291db” and a collection of “tweets”. The major part of this part is to load all information into the database that we just created. This program allows you to handle large amounts of data. For a 1.35GB JSON file, it will only take about 45 seconds to load everything into the database. The technique that we use is to insert the data with a small batch, our case uses 1000 as the batch size. That means the data will insert 1000 every loop so that it will not run out of memory while inserting the data.

ii) Search for Tweets

This functionality runs exclusively through the search_tweets function. The user will be prompted to enter one or more keywords which will then be used in MongoDB queries to search the database for the tweets that match with a case-insensitive search. The query will return the matching tweets, and the function will display them. After seeing all the tweets, the user can then see more information about a tweet, search for another tweet, or exit the search. If selecting a tweet, all of the information about the tweet will be printed for the user to see.

iii) Search for Users

The choice2 function acts as the main pipeline for the searching for users. There are also two helper functions. The search_users function searches the database for users whose location or display name matches the user-entered keyword. It does a case-insensitive search using the MongoDB query and aggregate function. The second helper function, select, returns all of the information on a specific user if they are selected by the current user. Through this, the choice2 function guides the user to either select a user or exit user search. After exiting the search, one can then re-enter and search for another user.

iv) List Top Tweets

This functionality allows the user to list the top n tweets according to three categories, retweetCount, likeCount or quoteCount. The results will be displayed in an order of decreasing order of the selected

field. For each matching tweet, we got the tweet_id, data, content and username of the person who posted it. Also, the users are able to select a tweet to see all tweet information.

v) List Top Users

The list_top_users function works to print the users with the most followers. It calls two helper functions, print_top_users and select. The print_top_users function takes the formatted result from the original function and prints it for the user to see. Then, the user is given the option to select a user, which is done by the select function. It takes the user id and displays all of the information on the user.

vi) Compose a Tweet

To compose a tweet, the compose_tweet function was used. This function inserts a new tweet with the given parameters in the assignment brief. The user inputs their text, and the tweet is inserted with the user being 291user, the date being the current date, and the text being the user input. All other fields are set to null by extracting the format of another tweet, and then using set_tweet to set other fields to null. This function loops through the items in the tweet dictionary and sets the appropriate fields to null.

c) Testing Strategy

fanbang: for phase 1, the major part is to load all tweets corresponding to the given JSON file within the given time. For each step, we use the MongoDB extension inside Vscode. From the three given data files. We first test the load_json with the smallest data file like 10.json which only contains 10 tweets. In this case, we can go inside the MongoDB extension to check the database. Once we see “291db” as our database name and “tweets” as our collection name. We are also able to use the extension to go inside our database to see each object with all fields. For the functionality testing, I choose to do query testing first, which will make sure my query provides the correct result then I will do the unit testing.

mekha: For my individual work, I broke up my code into different steps, and after I implemented each step, I ran tests using the 10 tweet database. Once each step was working as expected, I moved onto the next one and repeated the process. Once I was satisfied with my code, I merged it to the main branch and helped my group with overall testing and debugging.

jkisilev: For my functions, I downloaded the skeleton main.py file and some of the provided test databases. Similarly to the previous project, I tested my individual functions by themselves to find any local errors, and then tried testing everything together. There was some trouble with writing the MongoDB query, and I was able to get support from my group members in figuring out the errors in the queries. Afterwards, I tested my functions again to ensure they worked as expected.

bparkash: For my functionalities, I used the common test database that we are given and I tested it multiple times with different values by changing the databases. For instance, for composing tweets, I try composing the same multiple tweets and all exist in the database with unique _id. That means users can enter multiple tweets and there will exist multiple tweets. Once it was all tested and worked as expected then merge to the main file on github but before that i made sure my algorithm didn't interrupt with other members defined functions. In order to ensure this, I avoided global variables and implemented everything inside my functions with little changes inside the main function itself.

As a group, we met and tested all the functionalities in order to debug everything at once. This allowed us to test the entire system and find errors we did not find originally.

d) Group Break-Down

Our group, the Index Instigators, was formed during reading week, and we had our first full group meeting on November 20, 2023. It was then when we broke down the functionalities and tasks that the project demanded, and set deadlines to keep ourselves accountable to the rest of the group so we could finish the project in a timely manner. Each person had about two tasks to complete, and we split up the tasks by assessing the demand of each item, and pairing those with the least amount of time needed with the most amount of time needed, so each person would have a similar amount of time spent on the project. The breakdown of tasks is as follows:

- bparkash: compose a tweet, list top users (about 10 hours)
- fanbang: create main.py, phase 1, list top tweets (about 11 hours)
- jkisilev: search for users, design document (about 9 hours)
- mekha: search for tweets, flow chart (about 9 hours)

The breakdown of our project timeline is as follows:

- November 15, 2023: initial discussions over Discord, group formation over reading week
- November 20, 2023: group meeting, assign tasks, create deadlines, create github repository
- November 20 - November 27, 2023: work on functionalities of system
- November 27, 2023: final in-person group meeting for mass testing and debugging
- November 27 - November 28, 2023: add designDoc to github repository and submit project

Appendix