Index Instigators Design Document - Mini Project 1 - CMPUT 291 - Fall 2023

a) General Overview & User Guide

This program acts similar to a social media platform, giving users the ability to create an account, follow other users, search for tweets and users, and more. Before logging in, the user must input the name of the database in which the initial data is stored; it is then that the user will be able to use the application. While using the application, the user should select one of the numbered options if prompted with a menu, or follow the instructions presented for their input at any other input prompt. A flow chart is attached as an appendix to the bottom of this document to help the user understand which menu takes them where and how to navigate the application.

First, the user will be prompted with a login screen where they can create an account (register) or log in if their account exists already. Afterwards, they will see a menu of options ranging from 1 to 7. When prompted for input, the user can input any of those numbers and will be taken to the appropriate next page. They should then follow the prompts on the page. The application will provide the user with a menu of options from which they will need to input one of the numbers displayed, or they will be prompted for their input such as for a search or for writing a tweet. Within each menu, there will be options to exit the specific functionality of the program which will bring the user back to the previous menu. When the user is finished using the program, they can logout (which will bring them back to the login screen), and then exit the program completely, thus closing the application.

b) Software Design

i) Login Screen

A login screen should be clean and readily accessible. It lists the basic categories and gives options for the users. The user is given three options which include login, register and exit the program. This is done through the login_reg_screen function. For the first option, the user must have the user ID and password available, and thus can log in with this information. This is handled by the login function. For registration, the register function will be called, and the user will have to input their information to create an account. If users choose to exit the program, then the connection with the database will close immediately instead of waiting for the system to arrange. Then we also clear the shell.

ii) Search for Tweets

Once search for tweets is chosen from the main menu, the user is prompted to enter keywords. If the user wants to search more than one keyword, they separate them using commas.These keywords are used for searching tweets, and 5 tweets are returned if they contain the keyword (contains means that sequence of characters exists in the tweet). The user is given the option to view more tweets, and if they choose yes, 5 more will be displayed. This cycle continues until the user is satisfied with the amount of tweets, or there are no more tweets.

After the user has finished the initial search, they will be presented with a menu that contains options 1 through 6. If they enter 1, they will either see 5 more tweets (or less if there are less than 5 tweets remaining) from their original search, or they will be told that there are no more tweets containing the keywords. If they enter 2, they will be prompted to enter the desired TID, and specifications of the tweet will be returned. If they enter 3, the search process restarts. If they enter 4, they are prompted for a TID to retweet. If they enter 5, they are prompted for a TID to reply to and then the replying text. Lastly, entering 6 will take them back to the main menu.

iii) Search for Users

The choice4 function handles most of this functionality. Within this function, the user can search for other users, and the display of said users is handled. There are multiple helper functions which each come together to ensure that the user is able to search for users, select a user, see more users, follow another user, and more. After searching for a keyword, the user can then see more users or select a specific user and view their information. The user is also able to go to the previous page (added from project specifications) for ease of use which is within the choice4 function.

The first helper function is search, which returns a list of users whose names or cities match the keyword. It avoids duplicates by ensuring that if the user's name and city includes the keyword, they are omitted from the result of the city search. Secondly, there is the select helper function which retrieves the necessary information about a selected user to then be printed in the user_info function. This function prints the user's information and comes with its own menu. In the menu, the user may follow the selected other user, which would then invoke the follow function (which inserts a follows relationship into the table if it doesn't already exist), or try to see more tweets. In any of the inner menus of this functionality, there is an option to exit, which will bring the user back to the previous menu.

iv) Compose a Tweet

The compose_tweet function will update the tweets table and call a helper function to update the hashtag in mentions and hashtags tables. First, it will find the user from the data who wrote the tweet. The goal is to count how many tweets exist already, then to add 1 to have a unique tid, or if no previous tweets exist, to set tid to 1. After composing a tweet, update_mention_hashtag is called to update the other tables.

For the update_mention_hashtag function, it will update the mention table and hashtag table by calling helper functions update_hashtag and update_mentions. First, it checks if there is a hashtag inside the given tweet. There exist four scenarios: (1) user enters hashtag as last word and presses enter, (2) user enters hashtag and then a space to continue the tweet, (3) user enters two hashtags consecutively without any space between them, or (4) the tweet has no hashtag. If a hashtag exists within the tweet, the function will call the helper functions mentioned above to update the hashtag and mention tables.

In regards to the helper functions, the update_hashtag function updates the hashtag table with the appropriate information, and the update_mentions function acts the same but instead updates the mentions table.

v) List Followers

The list_followers function will list all followers and provide more options so the user can select what to do next. It will display all followers and if there are no followers, it will give a message to explain as such. In regards to the option menu, it asks users whether they would like to choose any of their followers to see more information about them, and if so, which follower. A user can choose to return back without having to see more information, but if they selected one of their followers, a helper function will be called to display all of the followee's information including the number of tweets, the number of users being followed, the number of followers and up to three most recent tweets.

In addition, the select and user_info functions are called within the list_followers function; they work in the same way as when searching for users. These functions will allow the user to see a specific user's information, follow them, and see more tweets.

vi) Logout

Users can choose to log out from the program after a successful login or register. After logging out, the program will go back to the main menu. Users will be able to choose one of the three login options again.

c) Testing Strategy

fanbang: for individual parts like list all tweets or retweets, I first created my branch, and tested my function in the database. Then I chose to use the query_test function to test the query. After that, I copied more rows into the testing data to test 5 tweets on a page functionality. For the group part, I merged my code into the main branch and then I tested my function's feasibility.

mekha: For my individual work, I created my own branch, and as I made each function I tested it using a test database. As each successive function was made, I tested it and its interactions with the existing functions. Once I was satisfied with my code, I copied the main file onto my branch, and then added each function one by one, to make sure it worked properly after the transfer. Once all of my code was copied over, I tested the overall functionality of both my code and the rest of the groups, to make sure that there weren't any unwanted interactions that resulted in errors. After I was confident with my testing, I merged with the main branch and repeated the tests.

jkisilev: For my functions, I first downloaded the test databases and the main function to work with. This helped me ensure my code would be compatible with that of my group members. Upon doing so, I used print statements to see the changes I was making to the relevant tables (i.e. follows), and to see the results from the SQL select statements, so I would be able to properly understand and display the data necessary. To make sure the code would run, I considered multiple edge cases of what the user may do, and accounted for them (i.e. if there were no new tweet pages, or no previous tweet pages) in addition to adding some extra data to the database. For this, try and except statements were used to counteract the error that would arise. Once satisfied with my tests, I reviewed my functions with the group, uploaded them to my own branch, and then merged them with the main function.

bparkash: For my functionalities, I used the common test database that we had and I tested it multiple times with different values by changing the database with some complex values to check if I covered most of the cases and especially constraints. For instance no two rows in a hashtag can have the same term or no two rows in mention can have the same tid and term. That means users can enter multiple similar hashtags under the same tweet but the mention table gets uploaded only once. 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 I avoided global variables and implemented everything inside my functions with little changes inside the main function itself.

As a group, we worked together to test the whole program after we had all finished our respective functionalities. This included testing edge cases and debugging one another's code. Also, we tried to reduce the redundancy of the code by implementing one another's functions where we thought was necessary. Overall, this part of the testing phase allowed us to find errors that we did not see at first and subsequently work together to fix said errors.

d) Group Break-Down

Our group, the Index Instigators, was formed on October 23, 2023, and we had our first full group meeting on October 24, 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 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 followers (about 15 hours)
- fanbang: create main.py, login screen, create test database (about 16 hours)
- jkisilev: search for users, design document (about 14.5 hours)
- mekha: search for tweets, logout (about 14.5 hours)

The breakdown of our project timeline is as follows:

- October 23, 2023: group formation, create github repository, initial discussion
- October 24, 2023: first group meeting, assign tasks, create deadlines
- October 24 - November 3, 2023: work on functionalities of system
- November 3, 2023: individual functionalities due (main.py due on October 30, 2023)
- November 4, 2023: upload video or text description of tasks for the rest of the group to watch (in order to understand how each functionality works)
- November 4 - November 6, 2023: merge individual files and test project
- November 6, 2023: final in-person group meeting for mass testing and debugging
- November 6 - November 7, 2023: add designDoc to github repository and submit project

Appendix