a) To run the program, you can enter python menu.py <database name> into the command line. The program starts with logged_in being false and user_input being 0. It then enters a while loop. At the start of the while loop it asks the user for input (1 = log in, 2 = register, 3 = exit). If the user enters 3, the

disconnect function runs and it exits the while loop, ending the program. If the user enters 1, the log in function is run. If the user enters 2, the register function is run. From there, all further options are listed and described.

For example, selecting a tweet 1 through 5 will display that tweet's information. 6 is used to search for tweets, 7 to search for users, 8 to compose a tweet, ect.

b) register() = Initializes a user count based on the number of existing users in the database then prompts the user to enter their name, password, email, city, and timezone. Then check if the entered password is not already

in the database. Next it checks if the entered email is valid by looking for '@' and '.' and then ensures the entered timezone is within the valid range (-12.00 to 12.00). Finally it inserts the user's information into the

database and commits the changes to the database. this function also makes use of status codes (0 for success or 1 for failure).

login(usr,pwd) = Input (usr: The username entered by the user and pwd: The password entered by the user.). First verifys if a given usr exists in the database. If the usr exists, retrieves the stored password associated

with that usr then compares the provided password (pwd) with the stored password. Finally, returns the appropriate values based on the results of these checks(1 = username doesn't exist in database, 2 = password is

incorrect, 0 = success). If the username exists and the provided password matches the stored password, the function prints "Login successful".

while loop = the while loop handles the running of the application when it ends the application ends

show_tweet_info(usr, iteration) = Returns tweet/retweet information of users being followed by a user. Includes the tid, tweet writer's usr, tdate, and tweet text if it was a tweet as well as replyto and retweet date if it was

a retweet. Only returns 5 tweets/retweets ordered by date. If the iteration increases, the subsequent 5 tweets/retweets will be shown.

search_for_tweets(keywords, iteration) = Returns tweets that contain a keyword in the text. Will display the tid, tweet writer's usr, tdate and tweet text.

search_usr(usrs, curr_user, page = 1) = Searches for a user and provides a list of matching users ordered by name and then by city. The function also prompts the user to view the profile of the searched user.

compose(usr, text) = Compose a tweet and store it in the database.

reply(usr, text, reply_to) = Create a reply to a tweet.

retweet(usr, tid) = Retweets a tweet.

list_followers(curr_user) = Lists the followers of the current user.

c) login stuff = my testing strategy was to check for common mistakes (incorrect username or password, timezone out of range, invalid email ect.). I mainly tested by running the program and checking various inputs to

confirm that they worked.

login and register test case 1: usr = 1, pwd = the, name = first, email = thefirst@gmail.com, city = edmonton, timezone = 4. it worked no errors were raised

register test 2: usr = 2, pwd = the, name = test. Password Taken error

register test 3: usr = 2, pwd = the2, name = test, email = notemail. Invalid email error

register test 4: usr = 2, pwd = the2, name = test, email = email@email.com, city = edmonton, timezone = -13. Timezone must be between -12.00 and 12.00. error

login test 2: usr = 1000, pwd = the. User not found error

login test 3: usr = 1, pwd = not the. Incorrect Password error

For testing search_for_tweets, I populated the database with tweets and ran the code to see if the desired results were returned. I followed the same strategy for testing show_tweet_info.

d) Our first hour long meeting was focused on breaking the tasks down into specific functions and then assigning each function to a group member.

Chris took care of the search_for_tweets and show_tweet_information functions. In total 2 or 3 hours were spent on these tasks. He also populated the database with several users, tweets, retweets, followers, ect. for testing purposes. This

took up another hour or so. Chris contributed to writing the designDoc as well, which took about an hour, and worked on fixing errors in the final code, which took closer to 3 or 4 hours of time. Brandon took care of display(),

is_valid_email(email), register(), login(usr,pwd) and other login features and also contributed to writing the designDoc, In total around 4 hours were spent working on the login functions and around 2 or 3 hours were spent on the designDoc.

Shubh took care of saerch_user(usrs, curr_user, page = 1), compose(usr, text), reply(usr, text, reply_to), retweet(usr, tid). A total 3-4 hours were spent on these functions and around it took 2-3 hours fixing the functions.

Jaskeerat took care of the showDetails() function, showNumberRetweets, showNumberReplies and ListFollowers function. showDetails takes the tweetID as a parameter and gives out the details of the specific tweet. showNumberRetweets and showNumberReplies take tweetID as parameter and give the number of retweets and number of replies as output. ListFollowers function lists all the followers of the user and gives the option of selecting the users and viewing a few of their details like name, city, etc. It took 3-4 hours for making and debugging these functions. Jaskeerat also made the menu and integrated all of the functions together and helped with non working functions which took an additional 3-4 hours.

Our second meeting was also about an hour, and was focused on how the completed functions would be assembled. We coordinated and discussed progress using a group chat throughout the project as well.

Our third meeting was spent cleaning up a few last minute errors.