CMPUT 291 - Fall 2023 - Mini Project 1

Design Doc

Team CPAJ - Camden Babin, Pranav Shanker, Adil Tapal, Jacob Winch

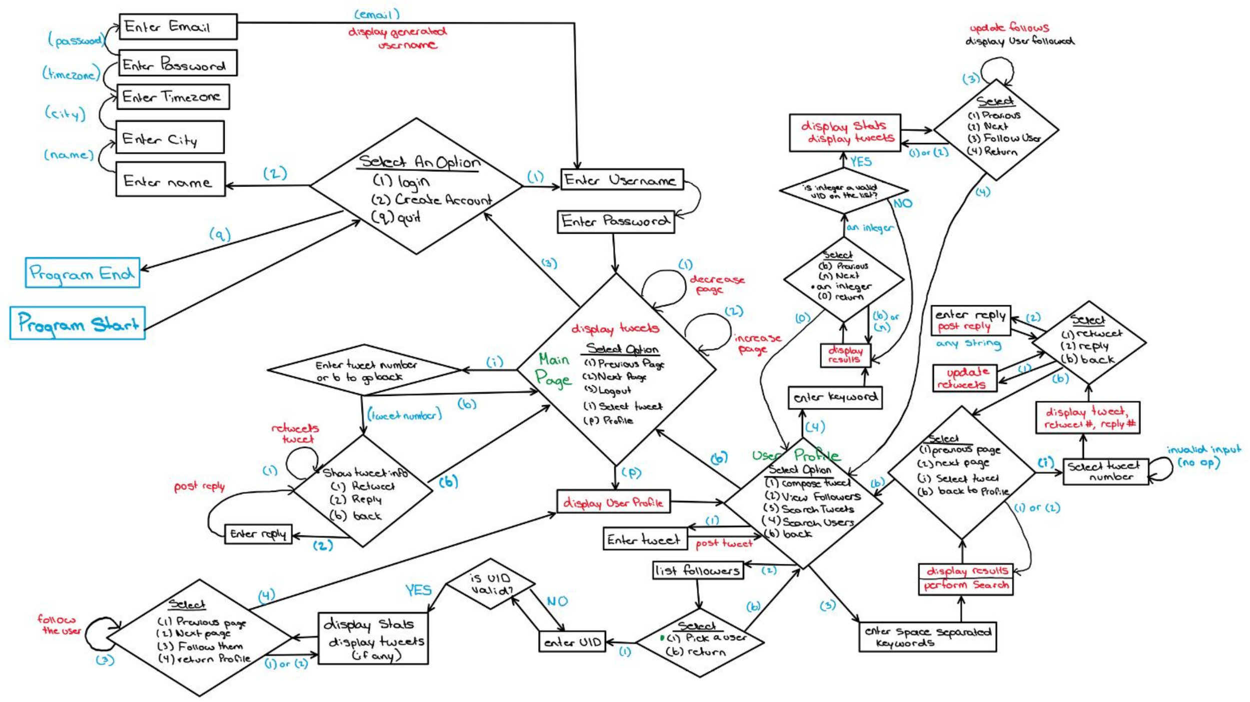
(a). General Overview of Our System with a Small User Guide

In this Project we used the functionality of SQLite database to mimic a twitter type application, we worked on this using a host programming language which allows us to create an enterprise database. Within this application, we worked on the following functionalities:
1) Login screen: the system allows us using queries to test the access into the application by allowing us to login with already created accounts or creating a new account.
2) After logging into the system, it allows us to interact with giving us quite a few system functionalities which include:

a) View feed page which shows us tweets and retweets from users we follow, ordered by date. At any point we only view 5 tweets with the option to load more if there are any.
b) Search for users allows us to search the database based on keywords such as mentions and hashtags.
c) The system also allows us to search for users based on keywords with results sorted by name and city, users can view the profile of the matched users.
d) The users can create new tweets or post replies to existing tweets, the system automatically updates the system on the backhand also updating the mentions and hashtag tables simultaneously.
e) List follower options list all the followers followed by the user.
f) Logout allows the user to safely logout of his account.
g) Creating a new account: the system allows to add new users, based on defined metrics of the system we error check all user input data. After successfully creating an account a user can start following other users and search for tweets.

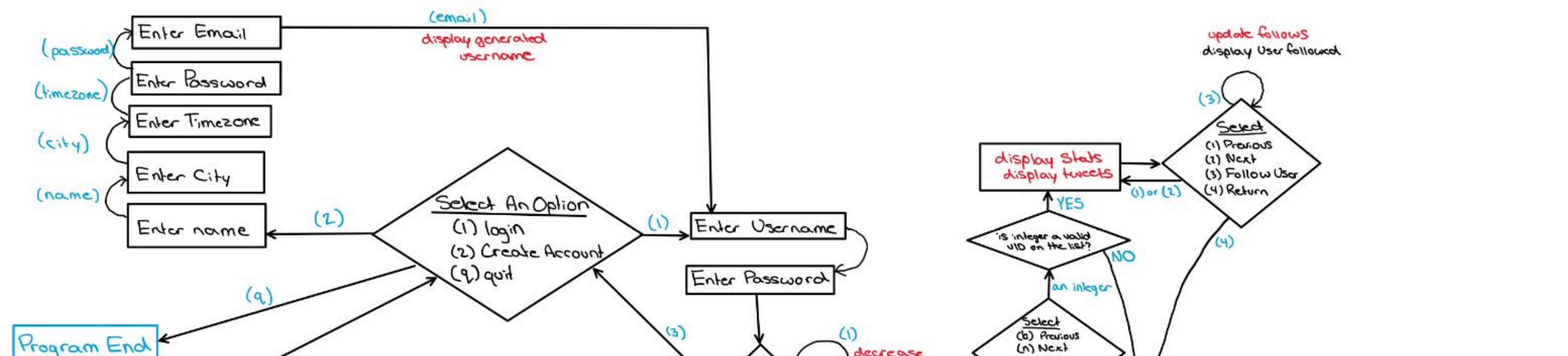
Figure 1 in the appendix shows the level of cohesion and coupling between the modules and classes used in our program. There are four distinct groups, the top left group represents the Display module, the bottom group represents the Functionality module, the top right group represents the Interface module and the UserSearch module on the right. The modules Display and Interface show cohesion. As these modules deal mostly with the text output and user display, many of the functions create a new display and call the next. The coupling between the Interface and the Display represent the function calls involved in signing in, which creates a display object from the Interface and allows the user to logout, returning from the display object and back to the interface. The Functionality class consists primarily of functions involving computations or database updates that need to be used in the display, which accounts for the high coupling between the Display and Functionality modules.

(b). A detailed Design of our Software



The following figure shows a detailed schema of our design project. Our code itself was divided into 5 of the following python classes:

1) Main.py: This includes our main function to start a connection with the database and program
2) Display.py: This includes our display class which loads our display with defined functions working along with functionality.py, interface.py and searchusers.py. It defines most of our command line user options and works with user input to run our program
3) Functionality.py: This includes the defined functionality options as per the requirements of the project including search tweets, compose tweet, list followers while also maintaining the output display specifications such as to display 5 tweets on the display at a given time.
4) Interface.py: This handles the login of a user into our system and launches our interface calling on display.py after a successful login. It includes our checking based on when logging in while also handling testing in the creation of a new account.
5) Searchusers.py: The UserSearch class is an extracted class for us to streamline our code, this works for the user search functionality allowing a user to search for other users, search tweets by keywords, allow the user to follow another user and show user stats.



This flow diagram can be used to trace user interactions with the program and maps out the full range of the system functionality from the program start to end. User input is shown in blue and system actions are shown in red. Diamonds represent a prompt the system gives the user. Example of a trace to search for a user, follow them, and exit thee program:
Program start > (1)>(username)>(password)>(p)>(4)>(emma)>(UID$_{\text{emma}}$)>(3)>(4)>(b)>(3)>(q) >Program End

(c). Our Testing Strategy

- In order to test our system we needed some data. Since we were not provided a test database we needed to create this test database. To create the test database we used the first database that was given to us in the previous assignment.
- Although the schema was similar there were key differences that we had to resolve. One of the key differences was that the users table in the test1.db did not include a pwd column that stores a user's password. In order to populate this column we just selected a password at random.
- Another difference was the primary key that was used for the tweets, mentions, and retweets tables. In the previous assignment, the primary key was writer and tdate but in this assignment the primary key was tid.
- So first we had to give each tweet a tid then match those tweets on writer and tdate to populate the primary key for the retweets and mentions tables. Once we had the test database we were able to test our program.
- There were several scenarios that we had to test for each specific functionality. Starting with the login page. Some scenarios that we had to test for the login page included, testing if a username is valid, testing if a password is valid, testing password case sensitivity, and testing to see if a user is a new or existing user.
- Some specific things that we had to test for the home page included showing tweets for a user with no followers and having multiple pages of tweets.
- Another thing that we had to consider was the order in which the tweets appeared. Some specific things that we had to test for the search for tweets functionality included testing for a match with or without a hashtag and the order in which the tweets had to appear.
- Some things that we had to test for the search for users functionality was making sure the order of users was correct, and that keywords matched. As well as when you follow a user it shows up in the database. This last part was something that we also had to test in the compose tweet functionality.
- For the list followers, we had to test that the user was following the users that were listed. As well as if the tweets were in the right order.
- The testing for list followers was similar to the search user functionality. We ran into a lot of bugs, too many to quantify. Some specific bugs that occurred were unique key constraint errors, type errors and many more.

(d). Our Group Work Break-Down Strategy

In order to keep coordinated our group created a discord server where we discussed how we would break down the project, meet to work on the project online, and a place to share our ideas. We split the project evenly by assigning an initial task such as working on the login page, or creating the database and a specific functionality to each member of the group. A detailed breakdown between partners is given below.

Camden Babin

Camden wrote base code for the login page with functionality for a user to login in and have the system output five tweets at a time with an option to scroll through them. Additionally, he worked on functionality for search tweets. From the main page displaying a user's tweets, he also created functionality to select a tweet and write a reply or retweet. For documentation, he generated graphs that describe the system and the user interactions.

Pranav Shanker

Pranav improved the starting user interface, as well as the login page which checks whether or not the correct password and username have been entered or not. He also implemented the functionality which made passwords non-visible during the time of writing. He created the logout page which would be executed once the user presses the specific command. Additionally, Pranav also assisted in implementing the list followers functionality as well as in creating the search tweets functionality. Pranav also aided in organizing the user interface and making it easier to use.

Adil Tapal

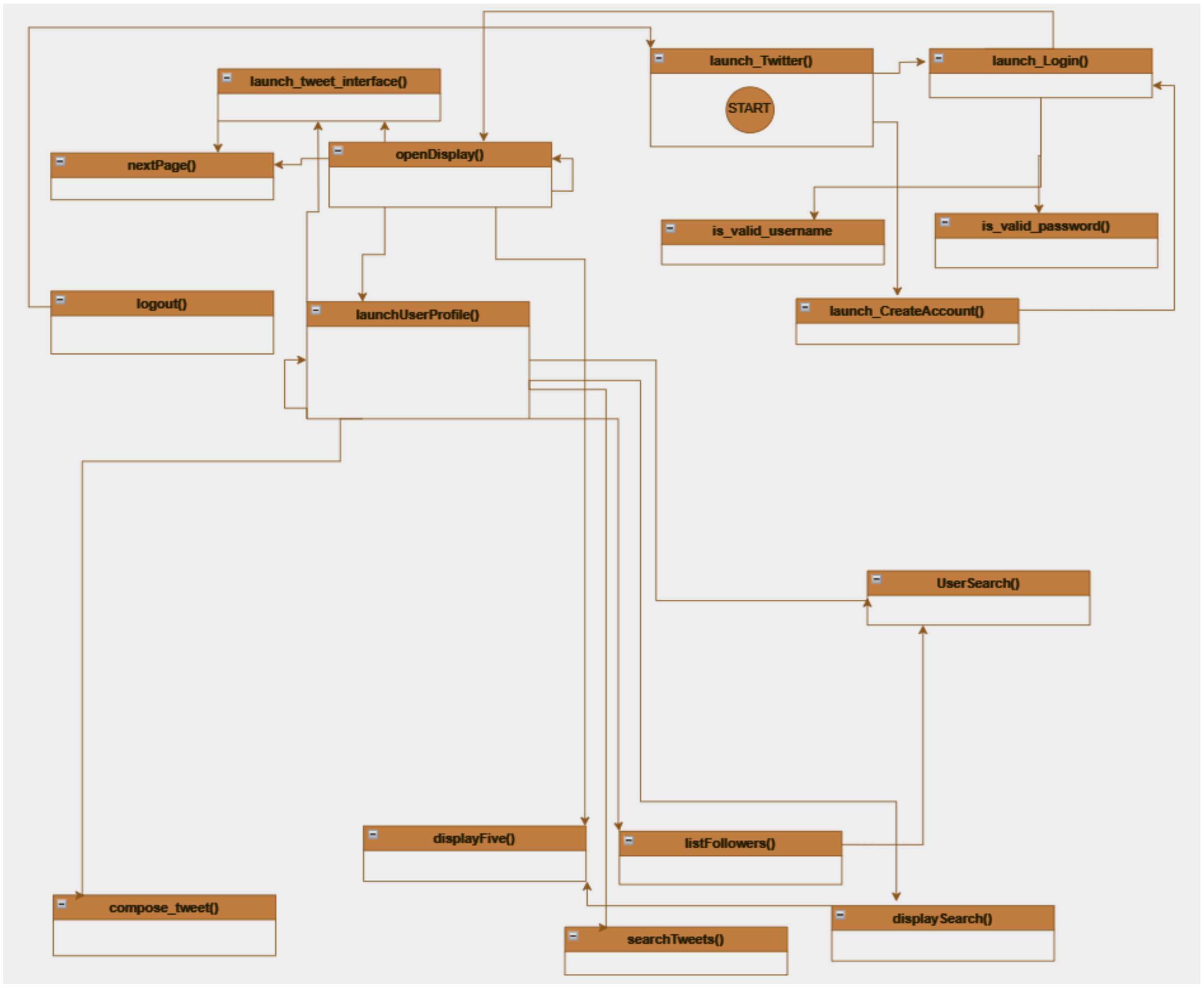
Adil created the compose tweet functionality along with working on the search user functionality, He also implemented and checked the mentions and hashtags table when composing a tweet or replying. He also worked on helper functions for search users, error checking and retweeting functionality. Adil also worked on the general overview and detailed design of the program. Approximately worked 12 hours on this altogether.

Jacob Winch

Jacob created the database that was used for testing by basing the database on the previous assignment. He also populated that database with new test cases. Jacob also implemented the entire search users functionality as well as helper functions that were used in the search for tweets, and list followers functionality. Additionally, Jacob aided in testing the program. In addition, Jacob wrote the “Our testing strategy” of the designDoc. All together Jacob spent around 10 hours working on the project.

Appendix:

<u>Figure 1</u>


launch_tweet_interface()
launch_Twitter()
START
launch_Login()
nextPage()
openDisplay()
is_valid_username
is_valid_password()
logout()
launchUserProfile()
launch_CreateAccount()
UserSearch()
displayFive()
listFollowers()
compose_tweet()
displaySearch()
searchTweets()