CMPUT 291 Project 1

1. Overview: Our social media clone application is built using Python and SQLite allows users to create accounts, log in and interact with each other through viewing other's tweets and retweets. The application allows users to search for tweets or users using keywords. Users can also follow each other and view other users' profile details. Users can also write new tweets or reply to existing tweets by composing new tweet text.

2. User guide: When users run the program, they are asked to first enter the database file name into the terminal to indicate which database the application will run on. Then, the user will be directed to the home screen where tweets and retweets from the users' followees are displayed 5 at a time. The user will then be given options to see more tweets and retweets from users they follow, which then display 5 more tweets or retweets on the window; or select a tweet to see more details, which the user can enter the tweet ID to see number of replies, retweets, tweet text and whom the tweet replies to; or to see more commands, which then takes the user to the next window; user can also opt for the last option to logout and go back to login screen. If users choose "More command" on the last window, they will be directed to the next window, where users are presented with multiple commands that allow operations on the database such as "**Searching for tweets**", "**Searching for users**", "**Compose a tweet**", "**List followers"**. Last but not least, users are allowed to exit the window to go back to the home screen. If a user chooses "**Searching for tweets**", they can enter a keyword to see more details of tweets matching the keyword, users can also choose to retweet or compose a reply to it. If users opt for "**Searching for users**", they will be able to enter a keyword and the application will return a list of users that have their names or cities matching the keyword. Users are allowed to enter a user's ID to see more details about the user such as number of followers, number of retweets, most recent tweets, etc. Users can also choose to follow another user if desired. If users choose the option to "**Compose a tweet**", users are allowed to write the tweet text into the interface, with any words after the '#' symbol will be saved into hashtags tables. At last, if users desire to choose "List followers", they are able to view the users that follow them and choose to see more user details, with opportunities to follow them.

3. Detailed design:

- We decided to break the application structure down into multiple files and functions that correspond to certain tasks which the application needs to perform to run efficiently. Some of those files and functions are standalone, while others are dependent on functions in a different file. We separate the components of the application into 3 categories stored in 3 different folders: SQLs that store all queries to be performed on the database, Controllers for files and functions that accept and perform certain user commands , Models that stores a class definition Users that store all necessary information of the current user using the app, and Interfaces for storing text for the user interfaces that users use to indicate what tasks they want the application to perform.
- First, we built a main.py file that initially sets up a connection and cursor with the specified database name and location from user input. The program then attempts to open the database file specified if it exists and prompts an error message in case of failure to open it. If the database connection is established, the application redirects user to login screen within a while loop to keep the application running

- Then the user is asked to input a specific command to choose whether to login, signup, or exit the application. If the user chose the signup option, the function signup would be called inside login_screen_actions.py that prompts the user to enter all valid credentials and generate a unique user ID to be added to the database through an INSERT query execution. If the user chooses the login option, they are prompted to enter their unique user ID and password for the account and the application will perform an SQL query to verify if the credentials provided match any entries in users table in the database. If the login is successful, the program will fetch user data from the database and store every necessary information inside a User object and the login function will return the ser object and an exit_program flag to False to main.py to indicate that a user is logged in and to be directed to home screen
- In the home_screen_actions.py, we decided to display the tweets and retweets from user's followees 5 at a time for each in print_tweets_home() and print_retweets_home() functions, we set up our query to fetch 5 tweets and retweets at a time using limit and offset at the end of each query, with offset value incrementing by 5 or the amount we want to see more every time. Users will then be presented with options to see more tweets or retweets, which the system will fetch the result of the same query call a before but offset value incremented by 5. Users can also choose the options to see more tweets details, which will direct them to tweet_info_interaction function in search_tweet_actions.py; or to see more commands, which the application will direct them to main_screen function in main_screen_actions.py. If the user chooses to exit and go back, the application will redirect them back to the login screen in login_screen_actions.py again.
- In main_screen_actions.py, the user is allowed to pick one of the 4 main command: Search for tweets will take them to search_tweets() function in search_tweets_actions.py, Search for users will take them to search_for_users() function in seach_users_actions.py, Compose a tweet will direct them to compose_tweet() function in compose_tweets.py, List followers will take them to list_followers() function in list_followers_actions.py; lastly, user can choose to go back which the application redirect them to te home_screen() function in home_screen_actions.py.
- In search_tweets_actions.py, we first ask for user input of keywords. We then process the keywords given, if the keywords do not start with '#', we find all tweet texts contains the keywords and if they do, we find the tweets that mention the hashtag in the mentions table. We display the tweets that match the keywords 5 at a time, we then display a list of next commands allowing users to choose to see more tweets, which the application fetch the next 5 tweets matching the keywords; choosing tweet to see more details, user can enter the unique tweet id in function tweet_info_iteraction() that displays all tweet information including retweets and replies count, users are also able to reply and retweet to the tweet in retweet_tweet() function and compose_tweet() function in compose_tweets.py. Users can either choose to go back to the search menu to keep searching for other tweets or go back to the window of the main list of commands.
- In search_users_actions.py, we ask for user input for keywords in search_for_users, the application will execute an SQL query that fetches all user that have their name or countries name matching the keyword, users will then be able to choose to see more users, which the

application will return the next 5 users matching the keyword in name or countries names; or see more user details in function see_user_details(), which user will first enter a user ID into the input then the application will return all information regarding the user such as number of tweets, followers, followees and 3 most recent tweets, from then, they can choose to follow the user being shown or see more recent tweets. Users are able to go back from this window to the list of main commands in main_screen_actions.py.

- In compose_tweets.py, we are able to write new tweets in compose_tweet() function, we note that the parameter reply_id is defaulted to None when we are not replying to any tweets. After users finish putting in the tweet text, the application processes the text and search for hashtags to put new hashtags in the hashtags table and insert into the mentions table with new tweet ID and all hashtags terms in the tweet text. After composing the tweets, users will be returned with a new tweet ID of the tweet they just created. All hashtags are processed and inserted into hashtags and mentions table in the hashtag_creation() and create_mentions() function
- In list_followers_actions,py, we first display a list of all followers of the current user. User will be able to choose to see more details about a user by entering their unique user ID which will display the user's number of tweets, followers, followees and their most recent tweets. The current user can either choose to see more recent tweets from the displayed user or follow them. The current can also opt to go back to the main window of commands in main_screen_actions.py.

4. Testing strategy

- Initially, we experimented with various AI tools to generate synthetic data and build a database for our application. Eventually, we settled on Chat-GPT. We initiated the AI engine to establish the database structure, which we then populated with authentic data across all tables. This thorough dataset enabled us to comprehensively test all functions of our application. We initially designed the functions to access different databases. However, During the development/testing phase, we limited the access exclusively to the test database. Subsequently, our focus shifted to fine-tuning the interplay of various functions, ensuring their seamless integration and intended functionality through a process of trial and error.
- Our main testing strategy is to test each functionality as soon as we finish building them to ensure they are working as expected due to some functionalities we build after may be dependent on the functionality we built before. Furthermore, as we try to refactor our code, we make small unit tests for every function to ensure the integration process of everyone's work does not break the program. These overall help maximizing the coverage of our testing process and minimizing the potential for bugs and unexpected program behaviors
- We also decide to populate the database with test data as we test the functionalities of our code, which increases the coverage of data across all tables and helps us keep track of expected results when we test the application functionalities.

5. Group work breakdown

- Thasanka is in charge of the composition of tweets and showing followers of a User. He is responsible for the code in compose_tweets.py and list_followers_actions.py. He also worked on the SQL queries and text commands that were used to accomplish these two tasks. He was also in the process of creating a GUI which was almost finished, although he decided to not use it for the final product due to the potential of several bugs. He spent approximately 15 hours on his work
- An is responsible for setting up the general structure of the application, including organizing files and refactoring code in suitable modules and functions to improve modularity and reusability. He is also responsible for building the welcome screen and login functions of the application. He also finished the "Searching for tweets" functionality of the application, including writing suitable queries, setting up proper interfaces' messages, building and organizing functions in the search_tweets_actions.py file. Lastly, he is responsible for integrating the work from all team members together in one complete project into a working application that satisfies all specifications. Hespent approximately 17 hours on his work.
- Josh is responsible for setting up the "Search for users" function of the application. This includes writing SQL queries to collect users information in the database and presenting it in readable format in the interfaces. He is in charge of the functions in the search_users_actions.py file, as well as some interfaces messages in the user_commands.py and user_search_commands.py in the Interfaces folder. He is also responsible for refactoring and organizing the file structure for his code to make it more readable and modular. He spent roughly 15.5 hours doing his work.
- Our method of communication primarily consisted of Discord chats, text messages, and version control via GitHub. We established 3 different branches other than main to work on our individual implementations without any commit conflicts, though we decided to remove the branches after the project completion to prevent any potential of confusion.

6. Assumptions and potential issues

AS-1: The user is assumed to remember their UserID and password.
AS-2: There is no constraint on the length of the tweet, the user is assumed to compose a tweet of appropriate length.
AS-3: Users cannot follow themselves
AS-4: For tweets, it is assumed that hashtags are always separated with a space from other words and is always non-empty (i.e "#" is not acceptable)
AS-5: Users cannot compose and add an empty tweet
AS-5: Application returns every tweet when user doesn't specify the keyword
PI-1: If the user has forgotten their UserID or password, there is no current way for them to retrieve it.
PI-3: Same user can create as many accounts as they want with identical information such as emails, name, password, etc.