# CMPUT 291 Mini Project 1

## Team: 291-prj1-teamlt

## **User Guide**:

Before launching the program the user must first type the following command while in the project directory: python3 -m venv venv, source venv/bin/activate, pip install -r requirements.txt. This launches the virtual python environment that has InquirerPy and rich installed as Tweeter is dependent on them. From this point enter python Tweeter.py <database> to launch Tweeter on the provided database.

## **General Overview:**

This program emulates key features of Twitter, encapsulating the following functionalities:

After launching the virtual environment the user types python Tweeter.py <database> to launch the program and enter the start screen.

In all screens the user navigates through the selections using arrow keys and when prompted to enter text may submit their text by pressing Enter. In order to exit the program the user should first logout then select Exit from the start screen.

1. Sign Up
   Users can create an account by providing their name, email, city, timezone, and password. A user ID is generated for subsequent logins.

2. Login
   Users log in with their user ID and password. Upon verification in the database, access to the program's full features is granted.

3. Follow Feed
   Post-login, users are greeted with the follow feed, displaying recent tweets and retweets from followed accounts and providing access to the main menu.

4. Function Menu
   This central hub links to all available features within the program.

5. Tweet Search
   Users can search for tweets using keywords. Relevant tweets appear in batches of five per page, with options to reply or retweet. Additional tweet details, such as the count of replies and retweets, are available upon selection.

6. User Search
   A keyword search to find other users by name or city. Matches are shown five per page, with options to view detailed information, follow, or see more tweets from the selected user.

7. Compose a Tweet
   Users can create tweets, which are directly posted or recorded as replies if originating from the reply function.

8. List Followers
   Displays the user's followers. If there are none, the user is notified accordingly. For each follower listed, detailed information is accessible, with options to follow back or explore their tweet history.

9. Log Out
   This function securely logs the user out of the program.

## Detailed Overview

Responsibilities & Interface:

start_screen()
- Prompts the user to login, sign up, or exit the program.

login()
- Requests user ID and password from the user.

sign_up()
- Collects name, email, city, timezone, and password for account creation.

quit()
- Closes the program.

follow_feed()
- Shows tweets and retweets from followed users, five at a time.

search_for_users()
- Asks for a keyword and displays users matching that keyword in name or city, five at a time.

search_for_tweets()
- Requests keywords, parses them for hashtags and text, and presents matching tweets in sets of five.

compose_tweet()
- Prompts for tweet text and adds it to the tweets table. It redirects to the function menu or previous screen accordingly.

list_followers()
- Enumerates users following the logged-in user.

logout()
- Resets user_id to None and navigates back to start_screen().

Structure & Relationships:
- The start screen directs to login, sign up, and logout functions.
- Signing up or logging in leads back to the start screen or to the follow feed.
- The follow feed connects to the function menu, which in turn links to user and tweet searches, tweet composition, follower listing, and logout.
- Searching tweets or users may lead to tweet composition.
- Logging out always returns to the start screen.

Sign Up
- Offers three choices from the start screen: login, sign up, or quit. Choosing sign up prompts for personal details and redirects to the start screen after account creation.

Log In

- After inputting valid credentials, the user is taken to the follow feed.

Follow Feed
- Displays a paginated list of tweets from followed users with navigation options. Absence of tweets defaults to the function menu.

Function Menu
- Presents six options: Follow Feed, Search for Tweets, Search for Users, Compose a Tweet, List Followers, and Log Out.

Search for Tweet
- Engages the user to enter keywords for tweet searches, offering paginated results with interactive options for replying, retweeting, or returning to the function menu.

Search for Users
- Prompts for a keyword to search other users, sorting matches by name and city with pagination. User selection provides detailed information and interaction options including follow and tweet viewing.

Compose a Tweet
- Asks for tweet content, then posts it. Context-sensitive, it may return to the function menu or previous screen.

List Followers
- If the user has followers, they're listed with options to follow back or view their tweets. Otherwise, a notification of 'no followers' is displayed.

Log Out
- Logs the user out and confirms with a message, returning to the start screen.

## Group Work Strategy

The group designated functionalities over a discord call and each member participated in regular meetings to discuss progress and concerns. Once all members completed their assigned functionalities we merged the branches and combined our individual work into a single program. After the files were merged Levi implemented the InquirerPy display for each functionality and made simplifications where possible.

| | Levi | Ivan | Tyler | Ky |
|---|---|---|---|---|
| Assigned Functionality | Login Screen | Compose a Tweet, List Followers | Search for Users, Logout | Search for Tweets |
| Time Spent in Hours | 15 | 10 | 10 | 10 |
| Progress Made | Complete display and functionality | Complete functionality | Complete functionality | Partial display and Complete functionality |
| Methods of Coordination | Discord, Github Issues | | | |

## Test Strategy

Our general testing strategy involved diabolic testing of the user interface and user input entry, scenario testing based on the demo session rubric. Due to the use of InquirerPy coverage testing would require extensive usage of mock objects in a non-standard library. Originally we were going to use ChatGPT to generate the test cases but it became apparent that it struggled with understanding how to mock the sqlite3 and InquirerPy methods.

We tested the following scenarios:
1. Sign up as a new user
2. Sign in with a valid user id and password
   - Attempt to sign in with an invalid user id or password
3. Search for tweets
   - Search using one keyword
   - Search using multiple keywords
   - Search keywords with prefix "#"
   - User searches with two of the same keywords
   - Ensure the matching tweets are sorted based on date
   - Ensure searches matching more than 5 tweets are broken down
   - Select a tweet to see more details
   - Reply to a tweet
   - Retweet a tweet
4. Search for users
   - Search using terms from name, city, or both
   - Ensure matches on name are returned first and are sorted on name length
   - Ensure matches on city are returned next and are sorted on length
   - Ensure searches matching more than 5 users are broken down
   - Select a user to see more details
   - Follow a user or see more tweets
5. Compose a tweet
   - Compose a tweet without a hashtag
   - Compose a tweet with hashtags
   - Compose a tweet with two of the same hashtags
   - Ensure hashtags are stored in tables "mentions" and, if needed, in "hashtags"
6. List followers
   - List the user's followers
   - Select a follower for more information
   - Follow a selected user
7. Handling string matches and SQL injection attacks
   - Ensure string matches are all case-insensitive except those for passwords
   - Ensure SQL injection attacks are handled
8. The user logs out

## Beyond what was required:

In the original specification the user was only able to view user information by searching for users and listing followers and tweet information only by searching for tweets. But in this finished program the user is able to select a tweet or user on any screen they are presented to see their information.
When searching for users if no keyword is entered then the program will return every user in the users table. This was useful in testing and it felt natural to be able to see every user and choose which to follow.
When the program displays confirmation that an action was successful it sleeps for 1 second. This allows the user to read the display before transitioning screens.