Design Document

This mini-project leverages SQL within Python to manage an enterprise database, providing users with services like tweet searches, follower lists, and user account management. Using a SQLite database with Python, the program offers a command-line interface for user interactions.

In this document we will go through the following:
I) General Overview (& User Guide)
II) Design of Software
III) Testing Strategy
IV) Work break-down

I) General Overview (& User Guide)

The application is a command-line tool designed to manage user tweets, followers, and lists using a SQLite database, accessible via Python. It integrates SQL queries with Python functions to process case-insensitive searches and secure logins. The tool includes a login system for registered and unregistered users and offers operations like composing tweets, searching users and tweets, viewing follower lists, and managing user-specific lists of favorite tweets.

User Guide:

i) Getting Started
- Ensure Python and SQLite are installed on your machine.
- Clone or download the project files.
- Open terminal and navigate to the project directory.

ii) Logging in
- For existing users, enter your usr (user ID) and pwd (password) to log in.
- For new users, choose the registration option, and provide your name, email, phone number, and password.

iii) Functionalities
- To search for a tweet, enter the corresponding number to the function, then enter keywords to search for the tweets.
    - To search for tweets containing certain hashtag(s), make sure to include the "#" symbol before each hashtag term.
    - To search for tweets containing certain words, do not include the "#" symbol.
    - Keywords are case insensitive.
- To search for a user, enter the corresponding number to the function, then enter an identifying alphanumeric keyword. This keyword should be contained in the name of the user you are searching for and is case insensitive.
    - If more than 5 users match, you can navigate through the pages of users using the given commands.

- To make a tweet, enter the corresponding number to the function, then enter the text of your tweet, including hashtags.
    - You will not be able to compose a tweet that uses the same hashtag (case insensitive) twice. If you do, an error message will appear and you will need to recompose the tweet.
    - You must enter at least one character in the tweet.
- To list the users who follow you, enter the corresponding number to the function. You will then see a list of up to 5 of your followers. If you have more than 5 followers, you can follow the corresponding commands to navigate between the lists.
    - You can select one of your followers to see more information, such as the number of tweets, followers, following, and recent tweets. You can follow the user, using the corresponding command, or see more recent tweets.

iv) Logging off

- To exit the program, ensure you are not logged into an account and enter the given instruction.

II) <u>Design of Software</u>

Our software is designed using a multi-file format for ease of adjustments.
The following are the files that are used for the functioning of our software:

i) **login.py -** This is responsible for handling user login and registration. It provides an interface to register a new user, authenticate an existing user, and display a login menu for user interaction. The functions that we have used in this file are:

- register_user(conn) - Registers a new user by asking for their name, email, phone number and password. It then generates a new unique user ID and saves the user information to the database.
- login_user(conn) - Validates an existing user by asking for their user ID and password and prints a corresponding message.
- login_screen(conn) - Provides a main menu for users to choose to log in, register, or exit.
- email_validity(email) - Validates if the user has entered a valid email address.

ii) **search_tweets.py** - This file handles the requirements of the first functionality. Allows users to search for tweets by keywords(including hashtags), view statistics on individual tweets and perform actions like replying or retweeting. This file has the following functions:

- search_tweets(conn, keywords, page =1) - Searches for tweets containing specified keywords or hashtags. Returns a list of tweets matching the search criteria.
- get_tweet_stats(conn, tid) - This function returns a tuple containing the retweet count and reply count for the specified tweet.

- compose_reply(conn, user_id, original_tid) - Allows a user to compose and post a reply to a specific tweet.
- retweet(conn, user_id, original_tid) - Allows a user to retweet a specific tweet, ensuring they haven't already retweeted it, and marks retweets as spam if the user tries to retweet more than once.
- show_tweets(conn, keywords, user_id) - Searches for tweets based on keywords and displays them to the user with options to reply, retweet or view statistics.

iii) **search_users.py** - This file handles the second functionality of searching for a specific user using a keyword. The functions used in this file are:

- user_flwee_tweets(conn, searched_id, offset) - Returns all tweets and necessary information tweeted by a searched user.
- follow_user(conn, user_id, searched_id) - Adds the current user as a follower of the searched user.
- user_details(conn, searched_id, user_id, user_name) - Displays the necessary information of the user and recent tweets. This function also provides a menu to follow the user, see more tweets, or go back to the main menu.
- follower_details(conn, searched_id) - Returns all necessary information of the searched user.
- search_user(conn, user_id) - Requests a keyword from the user and returns all users who match the keyword. If there are more than 5 matching users, provide an option to see more users. Also allows the user to select one of the searched users.

iv) **tweets.py** - This file handles the composition of tweets, the third functionality. It prevents invalid tweets, such as empty tweets or tweets that repeat the same hashtag. Upon invalid input, the file returns a proper error message. Upon valid input, the file adds the new tweet to the database along with any hashtags used in the tweet.

v) **list_followers.py** - This file handles the functionality of listing a user's followers and doing specific actions related to these followers. The functions in the file are:

- follower_tweets(conn, follower_id, offset) - This function returns the tweets that are to be displayed, with a limit of 3 tweets, including retweets.
- follow_user(conn, user_id, follower_id) - Checks if the user is already following the follower. If not, then the user follows them, otherwise an error message.
- follower_details(conn, follower_id, user_id, user_name) - This function displays the necessary information about a follower after the user has selected a follower. It also provides a menu for further user actions.
- user_details(conn, searched_id) - This function returns following information of a selected follower: the number of tweets, how many users they are following, and how many followers the user has.
- followers_list(conn, user_id) - This function returns a page of the user's followers, with a limit of 5 followers per page. It also provides a menu for the user to look at more of their followers or to select a follower.

vi) **display_tweets.py** - This file gets the necessary information of each tweet (the date, time, writer, id, text, and retweeter if needed) and then properly displays it for the main page. The function also allows the user to return to the main menu or to see more tweets.
vii) **main.py** - This file is used to combine all other files into one proper program. The functions in this file are:

- initialize_db(db_name) - This function initializes the database that will be used for the program.
- main_menu(conn, user_id) - This function displays the main menu that allows the user to use the various functionalities of the program as well as log out of the game or return to the main feed (list of tweets).
- main() - This is the main function of the program. It gets the database as a command line input and then displays a login screen.

III) Testing Strategy

Our group tested the program through a list of potential test cases and trials. First, we tested all the functionalities and requirements specified in the assignment overview on eClass. Afterwards, we listed potential edge cases and scenarios then tested those. We used both the sample database provided in the assignment description as well as another testing database with more users and tweets.

IV) Work break-down

We divided the program's functionalities into separate Python files and assigned them among members, along with tasks like writing the design document and compiling test cases. All members participated in testing, communicated via group chat, and made regular Git commits to avoid issues. Each member developed and tested their part, and the group collaboratively tested the full program, resolving issues together.
The breakdown of each part of the mini-group project is as follows:
Coding:

- main.py - all (30 minutes)
- display_tweets.py - Fredrik (45 minutes)
- list_followers.py - Fredrik (4 hours)
- login.py - Fredrik (2 hours)
- search_tweets.py - Basvi (approx 5 hours)
- tweet.py - Annie (1.5 hour)
- search_user.py - Fredrik (2.5 hours)

Other:

- Testing - all (4 hours)
- Potential test cases - Annie (1 hour)
- designDoc - Annie, Basvi (3 hours)
- README - Annie (1 hour)