General Overview

Our twitter program integrates all functionalities required by the specification by making SQLite queries to the underlying database from within the appication. The system ensures a seamless user experience by offering a range of interactive features and intuitive navigation, all accessible from the command line.

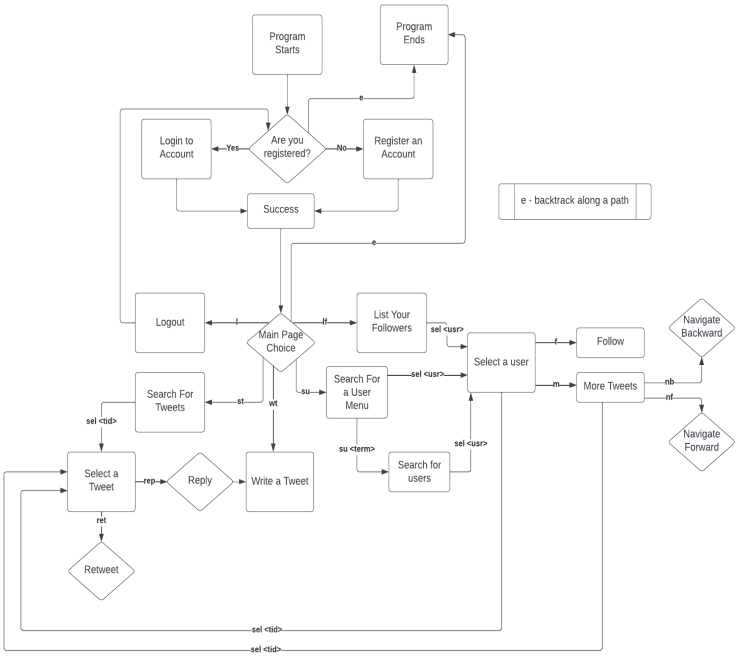
Preconditions:

1. Clone the repository
2. cd f23-proj1-61-6d-6f-67-75-73
3. pip install -r requirements.txt

Running:

1. python main.py [database.db]

If a database file is not provided then the database file test.db is automatically used.



Usage:

In the login menu, press y if you are registered and n if not registered. Press e to exit. If you press n, input a password, name, email, city, and timezone. You will be automatically logged in afterwards.

In the main menu, you can search for tweets, search for users, write a tweet, list followers, logout, or exit the program.

If you choose to search for tweets, you will be prompted again to search with a specific keyword or hashtag and the program will display the first 5 tweets with that keyword or hashtag. You can also choose to display the next 5 tweets with that keyword or hashtag, and select a tweet to reply to it or retweet it.

Similarly, if you choose to search for users you will be prompted again to search with a specific keyword and the program will display the first 5 users with that keyword in their name or city. You can also choose to display the next 5 users with that keyword, and select a user to follow them, see more of their tweets, and select one of their tweets to reply or retweet.

You can write a tweet by choosing the write tweet command and writing your tweet.

You can list your followers by typing lf. After viewing your followers you can select one and to see their metrics, including their last 3 tweets, follow them back, or see more details about one of their tweets.

You can logout with the l command or exit the program with the e command to close your session.

Detailed Design

main.py	**Input**: optional parameter which is database you are using **Function**: Handles the main event loop. If a database is specified, loads that database, otherwise loads the default test.db. Asks if user has an account already and calls on corresponding functions to handle. Once user is logged in, displays the main menu where a user can select to search for a tweet, search for a user, write a tweet, list followers, logout, or exit the program. Depending on the selection, it will call on any of the functions listed below. **Output**: None
login.py	**Input**: database connection (conn) **Function**: Prompts the user to enter their user ID and password, then performs an injection safe sql query to check if that combination matches any registered users. **Output**: The user id of the account if login is successful and None if unsuccessful
register.py	**Input**: database connection (conn) **Function**: Prompts the user to enter a password, name, email, city, and timezone. Verifies that timezone is of type float. Generates the userid (uid). Adds the new user to the database. **Output**: The uid of the account
composetweet.py	**Input**: database connection (conn), user id of writer (uid), optional parameter which is the tweet id of the tweet which we are replying to (replyto) **Function**: Prompts the user to write a tweet. When the tweet is written, the function calls upon auxiliary functions to grab any hashtags written in the tweet and generate a tweet id (tid). If there are hashtags in the tweet, the hashtag terms along with the tid are put into the mentions table. If the hashtag does not originally exist, it is also put into the hashtag tables. If there is a tid provided in replyto, then it adds the corresponding information to the tweet data. **Output**: Nothing. Database is modified to include the new tweet and potentially updates mentions and hashtags tables.

search_users.py	**Input**: database connection (conn), the userid of the user who is currently logged in (user_id) **Function**: Prompts the user to search for users by a specific keyword that matches their name or city. Upon choosing a keyword a helper query_users() is called which returns 5 results matching the search. The user is now given the option to navigate forward and backward to see more pages of 5 results each. A user can also be selected which will call select_usr() that displays more information about the user and allows for additional interaction such as following the user or seeing all their tweets. **Output**: Nothing, Database may be modified by functions called by this such as follow but this is not necessary or always true. All outputs are printed directly within the function.
searchTweet.py	**Input**: database connection (conn), the userid of the currently logged in user (usrID) **Function**: Prompts user to search for a tweet matching a keyword or hashtag or exit back to the main menu. If the user selects to search for a tweet they must enter as many keywords as they choose following the command (i.e. st this is a test). If a user puts a hashtag as a prefix for any of the keywords, they'll be treated as hashtags (i.e. "st #this is" "this" is a hashtag and "is" is a keyword). The function will search for tweets matching the keyword and tweets that match the hashtag and return them, ordered latest to oldest. Only 5 are displayed. The user is then prompted if they would like to select a tweet, navigate forward, navigate backward, or search for a new tweet. If a tweet is selected, they have the option of retweeting or replying to the tweet. **Output:** Nothing. May modify database if a tweet is retweeted or replied.

Testing Strategy

In general, we continuously tested the components of our program as we wrote them. To test our program we created a testdb.txt file which had all the SQL statements necessary for setting up the tables. Then we created a SQLite database using sqlite3 and the .read command. At the bottom of the testdb.txt file we added INSERT INTO statements for populating the tables with data necessary for testing. We also extensively utilized the sqlite3 command line tool for manually making queries to the database to check if our program was modifying the database correctly. Altogether this strategy was flexible enough to help us test all the components of the program and robust enough to allow each member to add their own test cases. We also tested the user input and output to ensure stability when given erroneous input.

The scenarios we tested were: registering users, writing tweets, displaying tweets, displaying users, displaying followers, searching keywords, searching hashtags and writing hashtags. A bug we encountered was the database not being updated to do a missing conn.commit().

Group Work Strategy

In our group we used an open and collaborative approach to make sure our project was completed successfully and on-time. Our first step was to divide the work up so we could each take responsibility for portions of the entire project. As some portions were larger than others, as we finished our portions we assisted the others in finishing theirs. The portions of the project are shown in the bullet points below with the time estimate and contributors show.

Planning:

- Time Estimate: 2 Hours
- Completed by Stefan (25%), David (25%), Andy (25%), and Shaheer (25%)

Login Screen:

- Time Estimate: 2 Hours
- Completed by Stefan (50%) and David (50%)

Search for Tweets:

- Time Estimate: 10 Hours
- Completed by Andy (75%), Shaheer (15%) and David (10%)

Search for Users:

- Time Estimate: 7 Hours
- Completed by David (75%), Shaheer (15%) and Andy (10%)

Compose Tweet:

- Time Estimate: 3 Hours
- Completed by Shaheer (100%)

List Followers:

- Time Estimate: 6 Hours
- Completed by Stefan (80%), Shaheer (10%) and David (10%)

Error Checking and Testing:

- Time Estimate: 12 Hours
- Completed By Andy (25%), David (25%), Stefan (25%) and Shaheer (25%)

Design Document:

- Time Estimate 4 Hours
- Completed By Andy (25%), David (25%), Stefan (25%) and Shaheer (25%)

As a group, we used GitHub for version control, which allowed us to all work on different parts of the project at the same time while keeping the project organized. To keep the project on track we maintained regular online group meetings where we would check in with status updates and brainstorm solutions to maintain a focused and collaborative environment. This allowed all group members to work efficiently and communicate effectively. Overall contributing to the success of this project.

Something that was not mentioned in the specification that we added was the ability to select a user's tweets after searching for a user.