General Overview:

The project encapsulates a twitter-like interface which models many of its key features; these features include, but are not limited to, Tweet Querying, where users can search the program's database for tweets filtered by a keyword of their choice. The functionality goes as far as being able to differentiate between hashtags and regular text to enable for a greater degree of precision in the search system. In addition to just searching for specific tweets, users can also interact with features that allow them to see and track sophisticated data analytics of said tweets. Meaningful data such as the number of retweets is accessible to all users, where they can further interact by choosing to retweet a tweet themselves, or to take it a step further, even reply. Users can also find themselves interacting with other users, this can be done by searching for users where they are given a list of similar users based on what they searched; the user can then browse the suggested users and view useful information about them such as their total number of tweets, followers and following. The program also includes functionality to allow for multiple distinct users through a login system while also protecting them and their information through a protected password feature. Last but not least, of course the user has the option to terminate the program at any given time where the program safely closes the database connection and informs the user through the interface.

Code Execution Guide:

a. Run the script
b. Input the name of the SQL database you wish to use
c. Login Screen
    1. Login
    2. Sign Up
    3. Exit
d. Main Menu(Displays all tweets from followed users)
    1. Search for tweets(input keywords[must match keyword or hashtag exactly] to retrieve relevant ordered by tweet date tweets based on hashtag or tweets keyword[can enter multiple keywords separated by comma] Furthermore, can also choose to select tweet(user numbers 1-5 for corresponding tweet) and chose to reply or retweet.
    2. Search for User(input keywords[does not need to match keyword exactly] to find users whose name matches the keyword in ascending order of name length. Can then select a user using number 1-5 for each corresponding user to see more information, and follow the user.
    3. Post a new tweet(enter new tweet text, with hashtag at the end, removes repeated hashtags)
    4. Lists followers(can then select and interact with selected follower to see more information or follow the follower)
    5. logout(brings user back to login/signup page)
    6. Shows next 5 tweets from followed users if there are more

Our software design centers around primary functions, each responsible for a key feature of the application. These functions interact with the SQLite database and the user interface. The responsibilities and interfaces of the primary functions are as follows:

- login_screen(connection, cursord)
    - Responsibility: Authenticates user credentials against the users table and initializes a session. Also allows to register user
    - Interface: Takes database connection, cursor, username, and password as arguments and returns the authenticated user's ID or an error.
- search_tweets(connection, cursor, keywords, user_id)
    - Responsibility: Searches tweets and hashtags matching the input keywords, displays results in paginated form, and allows tweet selection.
    - Interface: Accepts the database connection, cursor, search keywords, and user ID. Returns the selected tweet ID or continues the pagination.
- compose_tweet(connection, cursor, user_id)
    - Responsibility: Captures and stores a new tweet in the tweets table, extracts
    - hashtags, and updates the hashtag_mentions table.
    - Interface: Takes the database connection, cursor, and user ID as arguments and writes to the database.
- retweet(connection, cursor, tid, user_id, writer_id)
    - Responsibility: Records a retweet action in the retweets table and prevents duplicate retweets.
    - Interface: Requires tweet ID, user ID, writer ID, and database connection/cursor as inputs.
- displayStats(connection, cursor, tid)
    - Responsibility: Displays the number of replies and retweets for a given tweet.
    - Interface: Takes the database connection, cursor, and tweet ID, returning statistical output.

These primary functions are interconnected, leveraging shared data (e.g., tid, user_id) and the database connection for seamless operation. They form a modular structure, ensuring maintainability and scalability of the application.

Testing

Our testing strategy largely involved utilizing and running various trials with the provided database, however, due to the limited nature of the provided data, we personalized test cases and tailored them to target specific functions.

Break Down Strategy:

Garrick

- Tasks:
    - Assisted with implementing the login screen functionality.
    - Completed Question 3, which involved displaying the first 5 tweets upon user login.
- Time Spent: 5 hours
- Progress: Implemented and tested the tweet display functionality for login, ensuring it was user-friendly.

Eric

- Tasks:
    - Developed the Login Screen Implementation.
    - Completed Question 1, implementing keyword search for tweets.
- Time Spent: 6 hours
- Progress: Built a functional login screen and implemented tweet search. Retweets were not included as the requirements did not specify this.

Ali Zain

- Tasks:
    - Completed most of the logic and functions for Question 2.
- Time Spent: 4 hours
- Progress: Developed and tested core functions for handling tweets, retweets, and user interactions.

Ahyan

- Tasks:
    - Completed Question 4.
    - Finalized and bug-fixed user search functions for Question 2.
- Time Spent: 6 hours
- Progress: Improved user search functionality and ensured it integrated seamlessly with other components.

Coordination Strategy

- Communication: Used Discord to discuss progress, assign tasks, and solve issues collaboratively.
- Task Assignment: Divided work based on what we believed needed to be completed first or used in future functions.
- Progress Tracking: Shared updates regularly on Discord and used a document to track task completion.

This approach ensured all tasks were completed efficiently while maintaining clear communication.

○

Breakdown Strategy

Each member of our group made a dedicated contribution to the project, and we collectively agreed on how to divide the work. We used Discord to keep each other updated on our progress and solve any problems that we may have needed help on. Each person worked on their assigned responsibilities as they became available. We held daily meetings to discuss progress and what needed to be