Design Document

System Overview

This system is implemented in Python as a CLI application, with SQLite as the database, and a simulation of a social media platform. Users can interact with the system through commands to perform tasks such as posting tweets, replying to tweets, retweeting, viewing tweets, and managing hashtags. All data is stored in an SQLite database, and users can search tweets by keywords and view them sorted by date and time. The goal of this project is to provide a simple, CLI-driven tweet management system that simulates some basic functionalities of a social media platform . After login, they can:

- Post tweets
- Search for tweets or users
- View followers
- Logout

Files and Modules

The program is split into several modules, each of which handles a distinct functionality:

- **main.py**: Manages user login, user registration, system functions, and program flow.
- **follower_utils.py**: Handles the display of followers, including detailed information about each follower.
- **compose_tweet.py**: Allows the user to compose and post tweets.
- **tweet_search.py**: Handles the functionality for searching tweets.
- **search_users.py**: Allows searching for users based on input criteria.

Database Design

The system uses SQLite to store the following tables:

- users(usr, name, email, phone, pwd)
- follows(flwer, flwee, start_date)
- lists(owner_id, lname)
- include(owner_id, lname, tid)
- tweets(tid, writer_id, text, tdate, ttime, replyto_tid)
- retweets(tid, retweeter_id, writer_id, spam, rdate)
- hashtag_mentions(tid,term)

Module Details:

follower_utils.py Module:

The **followe_utils.py** module contains the following functions:

1. **showFollowers(user_id, cursor)**:
    - This function shows a paginated list of followers for the given user.
    - It checks if there are more followers to display, offering the option to load more or view the details of a specific follower.
2. **getFollowerList(offset=0, limit=5)**:
    - Retrieves a list of followers for the current user using SQL queries.
    - Supports pagination with an offset and limit, displaying 5 followers at a time.
3. **showFollowerDetails(follower_id)**:
    - Displays detailed information about a specific follower, such as their tweets and contact information.
    - Allows interaction with the follower, including viewing more tweets or following/unfollowing them.

Group Work Breakdown Strategy

The work is divided among the team members as follows:

- **Luke Thomas:**
    - **Task:**
    - **Estimated Time:**

- **Yuheng Li: follower_utils.py Module**
    - **Task**: Were responsible for developing the `followers_utils` module, which manages the followers and followee relationships within the application. This includes implementing the functionality to follow and unfollow users, checking the follower status of a user, and providing necessary methods to retrieve lists of followers and the users a person is following.
    - **Estimated Time**: 10 hours

- **Anant Gupta gurbaaz:**
    - **Task:**
    - **Estimated Time:**

- **Gurbaaz Gill:**
    - **Task:**

- **Estimated Time:**

Coordination Method:

The project uses Git for version control to manage code contributions. Team meeting was held once to discuss progress and resolve any issues.