1. General Overview of the System

Purpose:

The Barebones-Twitter system is a simplified Twitter-like application that provides users with a platform to interact through posts (tweets), follow other users, and explore content through a GUI built with Python's Tkinter library. This project demonstrates basic social media interactions, including posting tweets, retweeting, following/unfollowing users, and searching for users and tweets.

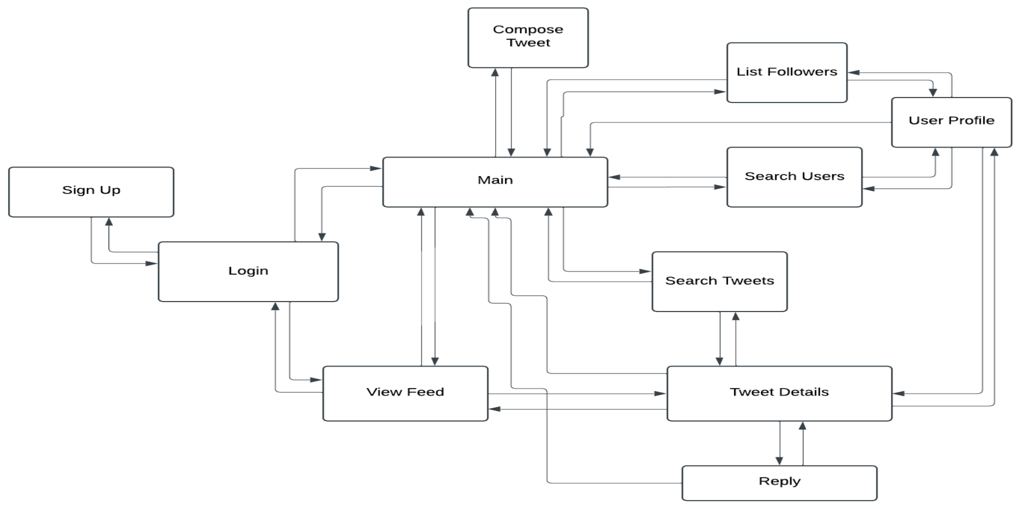
User Guide:

1. **Login**: Users can log in by entering their unique User ID (not username) and password.
   - Navigates to: Sign Up, Feed
   - Navigated from: Feed, Main Menu
2. **Sign Up**: New users can register with their name, email, phone number, and password.
   - Navigates to: Login
   - Navigated from: Login
3. **Main Menu**: Upon login, users are taken to the Main Menu, where they can: View their feed, Search for users or tweets, Compose a tweet, List followers, and Log out.
   - Navigates to: Feed, Search, Compose tweet, List followers, Login
   - Navigated from: all pages directly above it on the screen stack
4. **Feed**: The feed shows tweets and retweets from followed users, with the option to load more.
   - Navigates to: Main Menu, Login, Tweet Details
   - Navigated from: Main Menu, Login, Tweet Details
5. **Compose Tweet:** Users can post their tweets.
   - Navigate to: Main Menu
   - Navigate From: Main Menu
6. **User Profile:** Users can view other people's profiles with their following/followed status, and the tweets they have recently posted.
   - Navigates To: List Followers, Tweet Details, Search Users, Main
   - Navigates from: List Followers, Tweet Details, Search Users
7. **Reply: Allows** users to reply to someone's tweet.
   - Navigates To: Main Menu
   - Navigates From: Tweet Details
8. **Search Tweets:** Allows users to search for specific tweets in the database.
   - Navigates To: Main Menu, Tweet Details, Reply
   - Navigates From: Search Tweets, Tweet Details
9. **List Followers:** Lists the people that follow the User.
   - Navigates To: Main Menu, User Profile
   - Navigates From: Main Menu, User Profile
10. **Search Users:**
    - Navigates To: Main Menu, User Profile
    - Navigates From: Main Menu, User Profile
11. **Tweet Details**: Users can view a specific tweet's details. They can: Reply to a Tweet, View a User's Profile, and Retweet.
    a. Navigates to: Search Tweets, View Feed, Reply Field, Main Menu, User Profile
    b. Navigates From: Search Tweets, View Feed, User Profile

System Architecture:

The application's architecture follows a Model-View-Controller (MVC) pattern:

- **View**: Tkinter GUI provides an interface for user interaction.
- **Controller**: Python classes handle the logic and user interaction with the database.
- **Model**: SQLite database manages persistent data, including users, tweets, and relationships.



2. Detailed Design of the Software
Classes and Key Methods:

1. **Screen Class**
   - **Purpose:** allows a hierarchical OOP inheritance relationship between the classes
   - **Key Methods:**
     - build_user_interface(): contains the methods each specific screen needs to display an interface. This class is inherited and used by each and every single screen class
2. **App Class**:
   - **Purpose**: Manages the application's main workflow and screen switching.
   - **Key Methods**:
     - connect_db(): Establishes a secure SQLite connection.
     - show_login_screen(), show_signup_screen(), etc.: Manages screen transitions.
3. **LoginScreen Class**:
   - **Purpose**: Facilitates user login
   - **Key Methods**:
     - attempt_login(): Verifies credentials using secure SQL queries to prevent injection attacks.
4. **SignupScreen Class**:
   - **Purpose**: Handles new user registration with input validation.
   - **Key Methods**:
     - submit_signup(): Registers users after checking field entries and data validity.
5. **FeedScreen Class**:
   - **Purpose**: Displays tweets and retweets from followed users.
   - **Key Methods**:
     - load_feed(): Fetches tweets and retweets from the database based on follow relationships.
     - show_feed_items(): displays the current feed page
     - show_more_feed_items()/show_prev_feed_items(): Displays feed items from other feed screens.
     - update_button_state(): helper in updating the buttons
6. **SearchUsersScreen & SearchTweetsScreen Classes**:
   - **Purpose**: Provides search functionalities for users and tweets.
   - **Key Methods**:
     - search_users(): Filters users based on keywords.
     - search_tweets(): Retrieves tweets containing specified keywords or hashtags.
     - show_tweets()/show_users(): shows the current page of items
     - show_more_tweets()/show_prev_tweets()/show_more_users()/show_prev_users(): Displays items in other pages
     - update_button_state(): helper in updating the buttons

7. **TweetDetailScreen Class**:
    - **Purpose**: Shows tweet details, allowing retweets and replies.
    - **Key Methods**:
        - view_tweet(): Displays selected tweet information.
        - retweet(): Allows users to retweet.
        - post_reply(): Adds replies to tweets and handles associated database updates.
8. **ComposeTweetScreen Class and ReplyTweetScreen Class:**
    - **Purpose:** allows the user to write their own tweets/replies, with hashtags and such
    - **Key Methods:**
        - submit_tweet()/post_reply(): allows users to save their tweet to the tweets relation
9. **ListFollowersScreen Class:**
    - **Purpose:** lists all of the user's followers, and allows them to interact with them
    - **Key Methods:**
        - load_feed(): Fetches followers from the database based on follow relationships.
        - show_followers(): Displays the information about all the followers and allows interaction
10. **UserProfileScreen Class:**
    - **Purpose:** lists all of the information about a given user as well as their tweets
    - **Key Methods:**
        - load_tweets(): queries the users and tweets relations to get the necessary information needed to display the tweets that have been written by a user
        - show_tweets(): for a specific page index (tweets are divided into pages of 3), this function displays all of those tweets
        - show_prev_tweets()/show_more_tweets(): helper functions that help display the previous and next pages
        - update_button_state(): helper in updating the buttons

SQL Injection Protection:
Parameterized queries are used throughout the application to safeguard against SQL injection attacks, especially in the attempt_login() and submit_signup() methods where user input is directly tied to database operations.

3. Testing Strategy
Testing Types:
1. **Unit Testing**:
    - Methods like attempt_login(), submit_signup(), search_users(), and load_feed() were tested to ensure they handle expected inputs and edge cases (e.g., empty fields).
2. **Integration Testing**:
    - Tested interactions between GUI components, database queries, and screen transitions for smooth functionality.
3. **User Interface Testing**:
    - Verified that each screen displays properly in Tkinter, buttons respond as expected, and error messages are shown when necessary.

Testing Scenarios:
- **Login Validation**: Tested valid and invalid credentials, including SQL injection attempts to confirm defenses.
- **Signup Input Validation**: Checked invalid formats for phone numbers, emails, and passwords.
- **Feed Display**: Ensured tweets are correctly loaded in reverse chronological order and "Show More" displays additional tweets as expected.
- **Search Functionality**: Validated searches for users and tweets with single/multiple keywords.
- **Boundary Testing**: Checked maximum and minimum input limits in fields to ensure stability.

Bug Reports:
- **Feed Loading Issue**: Fixed an error where no tweets were displayed if the user wasn't following anyone by showing a prompt to follow users.

- **Retweet Functionality**: Addressed a bug where retweets weren't appearing in the feed due to incorrect data ordering.
- **Loading nonexistent database**

4. Group Work Breakdown Strategy
Group Members:
1. **Hasan Khan**:
   - Implemented all the primary classes, including LoginScreen, SignupScreen, and FeedScreen.
   - Integrated backend functionality and worked on SQL query handling.
   - Implemented Tkinter for a simple GUI so anyone can understand the application and use it.
   - Created design doc for code documentation etc.
   - Fixed majority of the minor bugs that popped up, such as following yourself, and # can't be alone.
   - Modularized the code by splitting the code into separate python classes and files for simplicity.
   - Estimated Time: 26 hours.
2. **Arden Monaghan**:
   - Created an alternative frontend template in PyQt and reviewed Tkinter design.
   - Assisted in debugging load_feed() and search_users() functions.
   - Worked on implementing the diagram representing the flow of the data and the User Guide.
   - Estimated Time: 9 hours.
3. **Rishabh Sharma**:
   - Core System Testing: Implemented comprehensive testing for all basic database operations including authentication, user registration, feed flow and tweet interactions. Covered searching tweets by hashtags and keywords, case-insensitive matching, listing followers and hashtag restrictions in compose and retweets. Also checked for profile statistics, counters for replies and retweets and input validation.
   - DB Injection Testing: Wrote extensive injection test cases to validate mitigations covering all input vectors. Tested login auth, user search and follower listing against union, comment as well as manipulation attacks and verified DB state after attempts.
   - Helped debug errors during development/refactoring.
   - Estimated Time: 6-10 hours.
4. **Gaurang Bhana**:
   - Implemented functionality: Added code modularity by grouping code into functions, and also resolved OOP design issues by grouping common attributes of the screen classes into a parent class. Added a screen stack functionality so that the program can remember the last visited pages so that it can go back to the pages that were visited earlier, making screen navigation easier. Added a previous button functionality to all lists and reworked the more functionality implementation.
   - Did some unit testing, functional testing, and bug fixing. Added all of the docstrings and almost all of the comments to the code and broke down complicated and messy functions and code into multiple simpler helper functions to improve legibility. Improved the layout of the code by breaking up a lot of complex functions into simple parts that logically fit together.
   - Enhancements: design improvements to the flow of the code. Fixed some of the queries that are vulnerable to injection attacks. Added more robustness to the queries and shifted a lot of data processing out of python and into SQL, including the use of regex queries.
   - Estimated Time: 22 hours

Coordination Strategy:
Weekly meetings were held via Discord to discuss progress and troubleshoot issues. A shared GitHub repository facilitated version control, and tasks were assigned through GitHub Issues for tracking.

Additional Considerations:
Our team chose Tkinter for simplicity and focused on achieving a secure database connection with robust SQL handling. The decision to add hashtag fuunicornnctionality was made to enhance search capabilities beyond the core requirements. We have used regexp extension from sqlean which allow to use regex pattern to use in Tweet keyword search query.