General Overview of the System:

These are some of the general features of the system:

1. **User Registration:** Users can sign up by providing their name, password, email, city, and timezone. The system assigns a unique user ID (usr) to each user.
2. **User Login:** Registered users can log in using their user ID and password. Upon successful login, they can access their personalized feed.
3. **Compose Tweet:** Users can compose and post tweets. Tweets can optionally be replies to other tweets.
4. **Follow and List Followers:** Users can follow other users, view their followers, and see who is following them.
5. **Search Tweets:** Users can search for tweets containing specific keywords, and the system returns relevant tweets.
6. **Search Users:** Users can search for other users based on their names or cities.
7. **View More Details:** Users can view additional details about other users, their tweets, and perform actions like following them.

User Guide:

1. **User Registration:**
    a. When you run the script, you will be prompted to input the name, password, email, city, and timezone.
    b. After registration, you can choose to log in by typing "Y" or "N" when prompted.
2. **User Login:**
    a. You can log in with your user ID and password.
    b. If you fail three times then it will automatically stop the login process.
    c. If you successfully log in, you will see your personalized feed with tweets from users you follow.
    d. You can then select one of the following tweets and see more information about them
3. **Compose Tweet:**
    a. In the main menu, choose option 3 to compose a tweet.
    b. Enter your tweet text, and it will be posted to your feed.
    c. Optionally, you can reply to another tweet by specifying its tweet ID.
4. **List Followers:**
    a. In the main menu, choose option 4 to list your followers.
    b. You can see the usernames and names of users who follow you.
    c. You can select a follower to see more details about them.
5. **Search Tweets:**
    a. In the main menu, choose option 1 to search for tweets.
    b. You can enter keywords to filter tweets, and the system will display relevant tweets.
    c. You can also choose to see more details about a specific tweet.
6. **Search Users:**
    a. In the main menu, choose option 2 to search for other users.
    b. You can enter keywords to search for users based on their names or cities
    c. You can select a user to see more details about them.
7. **Log Out:**
    a. On the selection screen, you can log out by choosing option 5 in the main menu.

Software Design

Our design of the software was functional based programming. Each function in our program does a small part of the required functionality. Most main functions in the program would take the database connection as the argument to have access to the data.

Login Screen

- The login screen used 3 main functions, the login_screen(conn), login(conn), and the signup(conn). The login screen also returns the user id in which most of the program relies upon.
    - login_screen(conn) was the main screen giving the user options to either log in, sign up, or exit the program.
    - login(conn) allowed a user to login with a unique user id and password. Upon logging in 5 tweets of people they follow would be printed (the feed). Then signup(conn) prompted the user to enter their information to create their own profile. This creates a unique user id for the user so they can log in anytime.

Search for tweets

- The functionality allows users to search for tweets by entering keywords, retrieving relevant tweets based on hashtags or text content, displaying them in reverse chronological order, and limiting the visible results to five at a time with the option to view more. Users can select tweets to view statistics, such as retweets and replies, and can also reply to or retweet them.
    - The main function used was search_tweets(conn, current_user). It first saves the keywords into an array then loops through each word to determine how it must be treated. If it starts with a '#' it will be searched in the mentions table, otherwise it will be searched within the tweet texts. We then save the keywords into an array to use within a query.
    - If there are any keyword_conditions or mention_conditions at the same time, they are combined into a single string with OR to form the complete SQL WHERE clause. We then select the tweet id, writer id, text, date, and if they were replies. Otherwise there is a query that fetches all tweet data.
    - Once we have the tweets we can print them 5 at a time. The user will be prompted if they want more tweets or not. If they respond with yes it will print 5 more, or the remaining amount of tweets. If the user doesn't want more tweets they will be prompted to inspect a tweet and see more details about it, reply to it, or retweet it.
    - Replying calls the compose_tweet() function that allows a user to create a tweet, and retweeting calls the retweet() function.

Search for users

- The system allows a user to search for other users by a keyword that matches their names or cities, displaying results in ascending order by name then city length, with a cap of five users at a time. Upon selection, it shows detailed information about the user, including tweet count, followers, and an option to follow the user or view more tweets. It mainly used the function called search_user(conn, current_usr) that took the database connection and the current user id.
    - The function first takes the keyword input that the user wants to search for. Then it uses the keyword in a query to search for the user's name or city that may contain the keyword. It selects the user, name, and city.
    - The query also orders the users by prioritizing the keywords matching users name by length, then orders based on the city length.
    - If no user is found it returns the user to the main screen. Otherwise it displays the users 5 at a time, prompting the user to display more users, or inspect a user. This would display the analytics of a user, then allowing the user to follow, see more tweets from the user, or go back.
    - When inspecting a user it used the function display_user_details(conn, user, current_user) to display the analytics of the user. It shows users id, name, city they're from, the tweet count, amount of users they're following, and their follower count.
    - When following the user, it calls follow_user(conn, current_user, target_user). First it checks if the user is already following the target. If they are, it prints that you're already following, otherwise it updates the database that the current_user follows the target_user. It does this using a simple query that inserts into the follows table.

- Displaying more tweets of the user calls display_more_tweets(conn, user_id). This fetches up to 5 more tweets of a user and saves it in a list. If the list is empty it means the user doesn't have any more tweets and we print "No more tweets from this user.", otherwise we print the tweets.

Compose tweet

- The user can create a tweet with hashtags identified by a preceding #, and these hashtags are recorded in the 'mentions' and, if necessary, 'hashtags' database tables. The main function used was the compose_tweet(conn, usr, text, replyto)
  - compose_tweet(conn, usr, text, replyto) is called anytime a user makes a tweet, this can be a new tweet or a reply. The function first gets the largest tweet id and adds 1 to it to get a new unique tweet id.
  - After creating the id it inserts the tweet text into the tweets table with the tweet id, usr id, tweet text, and reply to id.
  - It then searches for any hashtags in the tweet text and adds them to the mentions table in lowercase if they don't already exist in the table.

List followers

- The feature allows a user to view all their followers and select any to see detailed information, including the follower's tweets, who they're following, their followers, and their latest three tweets, with options to follow back or view more tweets. The main function used for this feature is the list_followers(conn, current_user).
  - The function first gets all the followers of the user, if they have no followers it prints you have no followers. Otherwise it displays all your followers.
  - The user is then prompted to inspect a follower, or go back. If they select a user to see more details, display_follower_details(conn, follower_usr, current_user) is called.
    - display_follower_details(conn, follower_usr, current_user) takes in the database connection, the user that's following the current user, and the current user. It first gets the users number of tweets, users being followed, and followers for the selected user then prints those details.
    - It then fetches the 3 most recent tweets of the user and prints them if they have any tweets. After it will prompt the user if they want more tweets or not. If not it will bring them back to the main screen.

Logout

- Log out uses the main() function in our program. The main program contains the main screen a user sees after logging in or signing up. This is where they can search for tweets, search for users, compose tweets, list followers, or log out. Each option is listed by a number where the user can choose from. Choosing one will call its respective function.
- The main function first calls the login screen to get the user's id. The rest of the functionality is then wrapped in a while loop that only breaks when the user logs out. This allows the screen to continuously be shown, allowing the user to continuously interact with the program.
- If the user logs out, it breaks the while loop, then closes the database, saving all the changes to

Testing Strategy:

Unit Testing

- Checking that each of the separate functions are working correctly
- Examples
  - connect_db() properly requests for a database to use, and creates a connection to the database
  - signup(conn) correctly takes a connection and creates a cursor along with user details to add to the database.
  - compose_tweet(conn, usr, text, replyto) gets the correct arguments and adds tweets or replies to the database

Integration Testing

- Testing the interaction between functions, such as signing up and logging into the program
- Testing that the number of followers, number of replies, or number of retweets is being updated if someone is followed, someone replies to a tweet, or retweets a tweet respectively

Functional Testing

- Testing the main menu options by simulating how a user would interact with it using different sequences of actions
- Making sure that the main menu options correctly go to the corresponding functions
- Verifying that users can log in and log out, and that the database connection closes when the system is exited.

Error Handling Testing

- Test the code by providing erroneous inputs to see how it handles exceptions and errors.
- Test edge cases for input
- Making sure that only valid options are given to the functions

Database testing

- Test interactions with the program and the database, such as the storage of user data which includes tweets, followers, retweets, etc.
- Verify that the tweets and respective user information are correctly stored and retrieved

Security Testing

- Checking for any SQL injection vulnerabilities

Usability testing

- Ensuring that the visual appearance of the program is friendly and clear
- Look for any interactions that feel uncomfortable or tedious in the user's perspective

Work Breakdown Strategy:

1. **Define and breakdown the project into its key components:**
    a. Clearly define the project or task's objectives, scope, and deliverables. Understand the goals and expectations of the work.
    b. Break down the project into its key components, phases, or major tasks. These are the high-level categories that encompass the work.
2. **Assigning members tasks:**
    a. Assign responsibilities for each sub-task to one of us. Ensure that each task has a clear owner or responsible person.
3. **Set Deadlines:**
    a. Establish deadlines for each sub-task. This helped us understand the project timeline and each of our time constraints.
4. **Communication:**
    a. We had daily meetings to report our progress on our current work in progress.
    b. We will give code reviews to each other to maximize efficiency in our code.
5. **Work in platform:**
    a. We used Github branches to merge our work into our main branch. This is used to keep track of what everyone is working on, and we don't merge any errors, or even be able to code review easier.
    b. Discord for our main communication channel. Used it daily for meetings, and if there were any problems we will report in our text channel.

Group Hours:

Login, signup, main: Andrew (10 hours, 10/31 - 11/02), Search for tweet and compose tweet: Reimark (12 hours, 10/31 - 11/02), list followers, Search for users, Login, Main and logout: Kevin (12 hours, 10/31 - 11/02), commenting, debugging and testing was done all together (8 hours, 11/03 - 11/06).