(a) a general overview of your system with a small user guide

The program starts with a login interface, giving the user the option to register, log in, or exit the program. The register option prompts the user to enter a username, a correctly formatted email, a 10 digit phone number, and a password, before sending them back to the login interface. The login option prompts the user for their username and password, and allows them access to the main interface if they match.

The main interface shows the logged in user’s most recent 5 tweets, and gives the user the option to search for tweets, search for users, compose a new tweet, see their followers, and log out.

The tweet search interface prompts the user for a list of keywords to search for, separated by commas, and prints out a maximum of the 5 newest tweets, showing the tweet type (t for tweet, rt for retweet), tweet date/time, and tweet text. They are given the option to show the next 5. The user is then asked if they want to see more statistics of a tweet, and is given the option to pick a tweet based on their position on the list to see more values like retweet count and reply count. Once the user is satisfied and does not want to see more statistics, they are asked if they want to reply to a tweet, selecting a tweet with the same system as before. They are then sent to the compose tweet interface, described below, then back to the reply tweet prompt. Once the user does not want to reply to a tweet, they are then asked if they want to retweet a tweet, using the same tweet selecting system as before and repeated until they do not want to retweet any more tweets. Finally, they are then sent back to the main interface.

The user search interface prompts the user for a search keyword, then prints a list of a maximum of 5 matching users, giving the user the option to show the next 5 users. The user can then select a user with their position on the page, and see more details such as their tweet count, number of users following/followed, and their recent tweets. They can choose to follow that user too if they wish. The user is then sent back to the list of users as before. Once the user navigates to the end of the user list, they are sent back to the main interface.

The compose tweet interface prompts the user for their tweet text. If the tweet has any hashtags with the same text, the tweet will be rejected and the user will be asked for a new tweet. After the tweet is valid, the tweet will be posted and the user will be sent back to the main interface.

The view followers interface shows a maximum of 5 followers. A follower can be selected by typing their name, and it will show more detail like their tweet count, followers/following counts, and some of their recent tweets. The user is then given the option to see more of their tweets, follow them back, or go back to the main interface.

The logout option sends the user back to the login interface as before.

(b) a detailed design of your software with a focus on the components required to deliver the major functions of your application

Please note:

- This program imports multiple functions from designated files (e.g. part1.py, part2.py, etc.) into main to help separate each code by their function. mp1.py is the main file that is run by the makefile to initialize the project.
- It also imports the ‘getpass’ module (from standard library) which makes the password non-visible when typing it in the register and login screen.

Main (mp1.py):
Sets up sqlite3 connection, creates tables if needed, provides the interface for the user to proceed to login/register functions, and once the user is logged in, it shows the 5 most recent tweets/retweets of the users followed by the logged in user, it also provides the interface for the user to use any of the 4 other functions (search tweets, search users, compose tweets, check followers) or show 5 more of the latest tweets/retweets or log out.

The login functionality accepts a string for name, and another string for password, the password is obtained by ‘getpass()’ function from the ‘getpass’ module so that the password is non-visible when typing. Then it is checked with the database and if the username and password is correct, the user is logged in.

register.py:
Sets up sqlite3 connection, accepts a string for name, string for email which is checked to see if it is in the format of ‘example@example.com’ by using regex from the ‘re’ module, string for phone number, checks if phone number is exactly 10 digits and will give the corresponding error messages if it is not, and password which is received with the help of getpass function imported from getpass module so that it is non-visible when typing. Then it imports the user to the database with user id being the user id of the latest user + 1. For example, if there are 5 users, and a sixth user is being imported, then the user id of the sixth user will be ‘6’ to ensure the user id is unique for every user.

Search Tweets (part1.py):
Accepts a string for keywords list, uses split() to turn it into an easily iterable list. Iterates through each keyword, creating a 2D list “tweets” each time to store the relevant data in the format needed by the project. Checks whether the first character is a hashtag or not. If yes, uses 2 queries to get data from the tweets table and the retweets table based on terms from the hashtag_mentions table, and orders them together using the sorted() function and datetime module. If not, it directly gets data using select queries from the tweets and retweets table and orders them together using the sorted() function. This additional step was required since adding a “type” column in SQL was somewhat difficult to distinguish the two tables. In the second case, it also checks whether the keyword is EXACTLY present in the tweet text using string properties in Python after the LIKE operator shortlists the texts. This is further documented in the code itself. Then, the program prints the top 5 values and uses the mod operator to decide the values currently in view. This is followed by questions about the statistics, replying to a tweet and retweeting a tweet for every keyword for the user’s convenience. The program also checks and returns errors for invalid input in each step.

Search Users (part2.py):
The script allows users to search for other users by keyword, view detailed user information, and follow them. It uses the input keyword to query the database, matching names partially using the SQL LIKE operator, and orders the results by name length. Results are displayed in batches of 5. Selecting a user retrieves their tweet count, following count, follower count, and up to three recent tweets. Users can follow another user if the relationship doesn't already exist. The program ensures all input is valid, validates database constraints (e.g., foreign keys), and prevents duplicate relationships, with errors handled gracefully.

Compose Tweet (part3.py):
Takes the sqlite3 connection, the user id, and a reply id that is set to None if the tweet is not a reply. The tweet text is collected from the user and the hashtags are extracted from the tweet text, checking for any duplicates and asking the user for input again if duplicates are present. Then, the tweet ID is set by getting the highest tweet ID in the tweet table and adding 1 to it, and the date/time is obtained by using the datetime library to find the current local date and time. Finally, this info is added as a new entry in the tweets table, and each hashtag found earlier is added as a separate entry in the hashtags table, and the user is notified that their tweet has been posted.

Check Followers (part4.py):
Takes the sqlite3 connection, and the user id. Checks whether the user has any followers and exits the function if there are none. Loops over the followers in chunks of 5 and asks the user which one they want to select or if they want to get more users. Displays how many tweets, follows, and followers the user has. Shows tweets from the people that follow the user from most recent to oldest in chunks of 3 and asks if they want to view the next 3 tweets or follow the person or go back to the main menu.

(c) your testing strategy
Based on the project specifications, we added our own data to the database (specified in mp1.py) and our own user inputs. Some edge cases we tested for include:

1. Users with no followers
   - Implemented by checking if they are a followee in the follows table
2. Entering whitespace or nothing as a keyword to search in tweets
   - Implemented by checking for length of keyword
3. Entering accidental whitespace entered by users when entered keywords to search
   - Implemented by using strip() to eliminate whitespace
4. Checking whether an element is searched for appropriately (for example, “#CMPUT291” should not find “#CMPUT291Feedback” and “CMPUT291” should not find “CMPUT291Feedback”)
   - Implemented using string properties in Python and equality operator in query for hashtag_mention table

5. Checking whether the program is case-insensitive
   - Implemented using LOWER() in query and string.lower() in Python
6. Checking if someone attempts to follow someone they already follow.
   - Implemented by referencing from the follows table
7. Checking whether the follower or the followee does not exist anymore.
   - Implemented by checking if they exist in the users table

(d) your group work break-down strategy
Login: Dion Alex Mathew (5 hours)
Part 1: (Search Tweets): Adit Sinha (6 hours)
Part 2: (Search Users): Arnav Sachdeva (4 hours)
Part 3: (Compose Tweet): Joshua Wong (3 hours)
Part 4: (List Followers): Jordan Kwan (6 hours)
Part 5: (Logout): Dion Alex Mathew (1 hour)
Design Doc: Everyone (about 30 min each, 1.5 hours for Joshua)

We had online google meet meetings on Monday, Wednesday, and Friday, as well as a group chat to organize everything.

Decisions made for the project that were not in the project specification:

- Since the retweets table does NOT store retweet time, each retweet is stored or printed with a value of “00:00:00” as an alternative of a “NULL” value to order data in the search_tweet() function appropriately.