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

To start the program, you first need to load the json file you wish to use into the MongoDB database. This can be done by writing the following command in the terminal: python3 load-json.py (json file) (MongoDB port number). Once the database has been created, in the terminal you can run the main file by writing the following command in the terminal: python3 main.py (MongoDB port number), it will prompt you to the main menu of the program. Displaying options 1 to List top users, 2 to Compose Tweets, 3 to List top Tweets, 4 to Search Tweets, 5 to Search Users and 6 to exit. If you enter 1 it will prompt you to enter the number of top users you want to list, once you enter a number it will display n number of users, allowing prompting the user to enter a username to see additional information. Once you enter a username all the user's information will be displayed, prompting the user to exit back to the main menu or enter another number n. From the main menu if you enter 2 the program will prompt you to enter the tweet description, once you press enter to compose the tweet you will be exited back to the main menu. If you enter 3 to List top tweets, the program will prompt you to enter either retweetCount, likeCount or quoteCount. Once you enter one of the responses it will prompt you to enter a number value n. Once you enter n, the program will display n number of tweets, you are then able to enter a tweet ID to see more details of a tweet. After seeing the tweet details you can enter exit to return to the main menu or enter any key to get prompted to enter another tweet id. To search for tweets you enter 4 and you'll be prompted to enter any number of keywords. Once you enter the keywords the program will display the id, date, content, and username from the tweet, prompting the user to enter one of the given tweet id. Once you enter the tweet id all fields from that tweet will be displayed, allowing the user to enter another tweet id entering exit to exit. Once you enter 5 you can search Users. Prompting the user to enter a keyword to search in cities or users. Once you enter a keyword the username, displayname, location and number of followers will be displayed. Prompting the user to enter a username to see additional info or enter exit to go back to the main menu. Once you enter a username all information will be displayed and you will be redirected to the main menu.

(b) details of your algorithms

The provided Python script utilizing PyMongo showcases a comprehensive approach to handling large data collections and optimizing search operations. The list_top_tweets function employs MongoDB's sorting and limiting capabilities to efficiently retrieve and display the top N tweets based on a specified field, addressing the need for quick access to relevant data. display_full_tweet function efficiently retrieves and displays the full details of a selected tweet, encompassing various fields such as date, content, and user information. In the search_tweets(db) function using the $and operator combined with a list comprehension in Python to create a regex search condition for each keyword. Using word boundaries (\\b) in the regex and case-insensitivity ($options: 'i') to make sure each keyword outputs the correct tweets. Properly tested to work for large datasets without encountering any runtime issues. In the search User function we get an input then we use a query to get all users that have that input in either their location or their display name. We use \\b and case insensitivity to get valid outputs. I then have arrays set up for usernames, displaynames, locations,follower count, and tid's to quickly and easily access all the things I need when printing or checking for duplicates. Then I loop through the result I got and start filling in the arrays and checking for duplicates as I go. After that I just print it and call expandUser function. For the expandUser function I prompt the user to either go back using 'exit' or to select a user to see more detail. Then I check the input and accordingly go back to the menu using return or print the user's information then go back to the main menu. For the case of an input that is neither of these two I prompt the user to either exit or select a valid user.

(c) your testing strategy

Initially as the project was being created, each part was tested on its own first to ensure proper results, then integrating each aspect together and retesting each individual part ensuring the output remained consistent and correct. To test search for tweets, we started out by testing an individual keyword, then to multiple keywords ensuring the output only contained a tweet if it had all of the keywords. Additionally checking for case-sensitive keywords to ensure the output remained the same. Following checking the keywords we ensured that the id, date, content, and username were shown, prompting the user to enter a tid to see all additional fields. Each step worked properly both before and after integrating with the rest of the functionality ensuring for the desired results.
To test list top tweets we first tested the fail-safes to make sure the user is inputting the correct input to not crash the code. After that, we loaded the prompt and inputted any of the respected fields that were asked of. After selecting them we checked the fail-safe for ‘n’ to make sure it had to be a number that was within 0 to the max number of documents present. If 0 no data. Once a proper n is chosen we checked to see if the count of n matched the total number of users displayed. Then we checked the fail-safe for the display full user information. We did this by inputting numbers and letters to try to break the code. After making sure that the user ID needs to be valid, the displayed user information would show.
For the Search user Id the testing strategy was to test first with the small database to see if it generally works with varying uppercase and lowercase, keywords that have no users attached to them, empty inputs, testing the exits, the user expanding to check for valid outputs, or to select a user that is not included in the selected list,and then we moved on to the larger databases to make sure the code doesn't take that long or doesn’t break when exposed to large databases. The smaller databases were good to test the functionality but the bigger databases were used to see our code’s performance and timings for the tests when for example there are 420k test cases, the code should still run with no issue.
For compose tweets I tested that it composed the tweet with no errors, with every other field being null except the username, content, and date. Also to make sure that the tweet was composed, we searched for it afterwards.
For list_top_user, we tested for whether or not the users were printed in descending order. We also made sure to test that the input entered is an int and is not a negative number. We also made sure that the input for the username was required to be a valid username shown. If any of these requirements are invalid, it will show an error and prompt the user to retry.

(d) your source code quality

Source Code quality is dependent on various factors such as Readability, efficiency, performance, documentation, comments, function headers and scalability. For example, for our code we tried to keep the code readable and space it out with proper indentation and comments and function headers help a lot by allowing the reader to understand certain components of the code. For example function headers tell you what the overall function does, while comments give you details on the smaller parts and more specific. Our code’s performance and scalability was put to the test when we used the large database with 420k entries, and if our code was not efficient enough, we would have run into problems. We optimized performance and efficiency by minimizing our uses of nested for loops or parts which lead to higher runtime which would cause issues.

Group work Strategy:

To coordinate the work between our group we used discord, discussing the different elements of each part together first and then deciding how to split up the work based on the perceived complexity of each part. In the end we decided to complete phase 1 as a group and then split up phase 2 based on the complexity of each part. In the end Phase 2 was split by having Nathan Trueman do Search for Tweets, Ilham Arbabzada doing Search for Users, Lawrence Omotosho doing List top tweets and Nathan Lin doing List top users and compose a tweet. This strategy worked fine however after finishing the project, the breakdown could have been further optimized due to similarities some of the different functionalities have between each other. We made sure to stay connected using discord both to help each other if one person didn't understand something about their part of needed help. Checking in frequently with each other to ensure the project went smoothly and each functionality worked as expected. After finishing each functionality from both phase 1 and phase 2 and working together to integrate the project as a whole, additionally testing each part multiple times for any input case, each group member worked roughly 8-10 hours on the project.