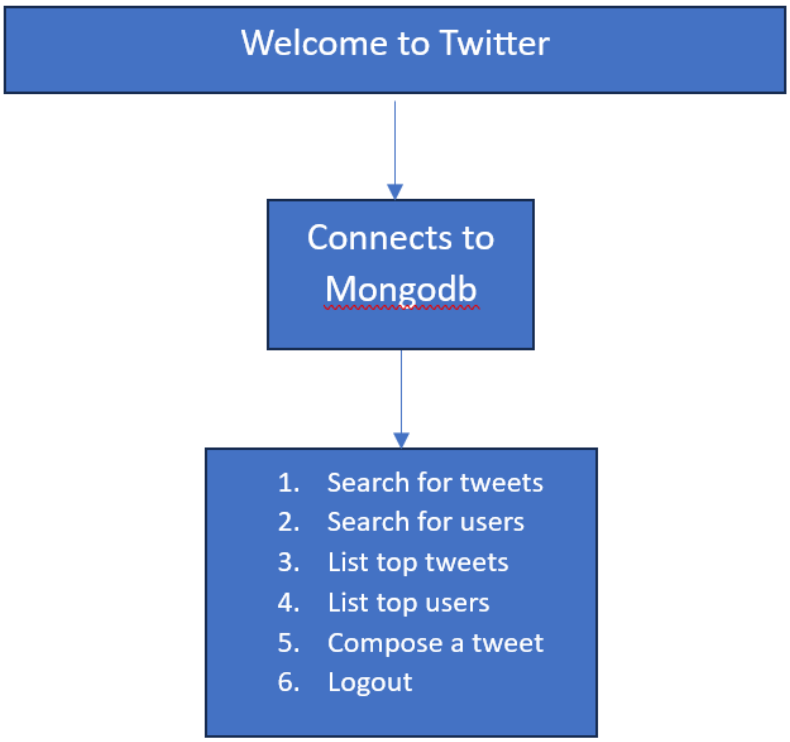
Design Document

Team Byte: Dhruv Kairon, Jaspreet Singh Chhabra, Rithwik Korukonda Bhattar

General Overview



There are two programs in this project: load-json.py and main.py, which handles all the functionality of the project.

User Manual

The user will run load-json first with the following command line arguments: name of json file, port number used to connect to mongodb. The load-json will load the database with collection as per in the given json file. The batch size can also be adjusted in the file. Once the database is filled with the collections, the user can now run the main.py to access the database and utilize its functionality. Once the user runs the program with port number as a command line argument, the user will be given a menu with 6 options: 1. Search for tweets, 2. Search for users, 3. List top tweets, 4. List top users, 5. Compose a tweet, 6. logout.

Detailed Design

Main functions breakdown:

1) Main
2) Search for tweets
3) Search for users
4) Compose Tweets
5) List top tweets
6) List top users

Main: main function has the menu for our database. The user is given 6 options to choose from and based on what the user chooses the database is manipulated.

Search for tweets: We used split to break up the comma separated inputs to a list, and then we use regex to input them in the query which gets the results which match in the content key. After this, we handle the output to print the ID, date, content, and user, and then ask the user if they want to see all the details. If so, we print everything for that id.

Search for Users: We input to get the keyword, and then we use regex to input them in the query which gets the results which match to the displayname or location key. After this, we handle the output to print the displayname, username, and location,, and then ask the user if they want to see all the details. If so, we print everything in that user in the dictionary using pprint

List Top tweets: We first use aggregation in our query to remove duplicates based off id and then order it in descending order based off type. After that, we print the details accordingly and then print the entire thing using pprint if the user requested for it.

List top users; We first use aggregation in our query to remove duplicates based off user_id and then order it in descending order based off followers count. After that, we print the details accordingly and then print the entire thing using pprint if the user requested for it.

Compose Tweets: We ask the user to input the content and we get the time using the datetime library and edit the entire tweet dictionary accordingly and add it to the collection using insert_one.

Group Work-Breakdown Strategy

To ensure that everyone had similar amount of work to be done, we split up the work:
Dhruv (11 hours): Function for list_top_tweets and list_top_users

I spent time doing these two functions because of their similar approach. After using load-json that Jaspreet finished before, we used all given datasets to test these functions for duplicates and other edge cases. I also worked on formatting and making sure our output was flowing correctly by using pprint and editing main()

Jaspreet(11 hours): Contributed towards Functions for searching tweets and composing tweets along with load json
I spent time coding up the functions for the tasks search for tweets and compose tweets. I used the sample test data to test my functions separately, handling edge cases and ensuring that it works smoothly. I also contributed towards bringing together all the work which the three of us have done, and contributed towards the design document. Additionally, I contributed towards writing load_json.py.

Rithwik(11 hours): Contributed towards Functions for Search for Users contributed towards list_top_users
I spent time coding up the functions for the tasks search for users and list followers. I used sample test data to test my functions separately, handling edge cases such as duplicates and ensuring that it works smoothly. I also contributed towards bringing together all the work which the three of us have done and contributed ttowards writing the design document and the user manual.

Work we did together: Debugging, testing, and integrating individual functions together. We all dissected the problem and made sure functionalities worked individually before putting it all together. Unsurprisingly, there were so many bugs initially when we put everyone's individual functions together but we eventually resolved all issues and finished the documentation.

Testing strategy

We mainly relied on the json files given to test our data, checking to see if our results are accurate. Additionally, we made sure to have a lot of print statements in our code whenever we want to find a mistake in the code. While testing we found plenty of bugs. We used the 100.json file for smaller edge cases. Here are some examples:

1. Tackling duplicates was difficult. Editing MongoDB queries proved more challenging than sqlite
2. Edge case errors were also significant as we tried to map unlikely responses and pruned those errors.