Project Report: Mini Project 2

Database Masters

System overview & User Guide

This program is a mockup of a Twitter application, which acts on a database storing data for tweets. Users can perform some of the actions found on twitter. The main functionalities of this application are: 1:searching for tweets, 2:searching for users, 3:listing top tweets, 4:listing top users, and 5:composing tweets.

The program consists of 2 python modules: `load-json.py` and `main.py`. `load-json.py` is used to load a json file containing a collection of tweets into a mongodb database. It contains the `load_json(json_file, port)` function which handles this. `main.py` is used to perform the main functionalities of the program. Its major functions are `tweetSearch(col), userSearch(col), listTopTweets(col), listTopUsers(col), composeTweet(col).`

To start the program the user must first load a json file containing a collection of tweets into a mongodb database, specified by its server port, by running the command

```
python3 load-json.py <json file name> <mongodb server port>
```

After loading the tweets collection into the database, we can start the main program by running the command

```
python3 main.py <mongodb server port>
```

Once started, the program can be interacted with through the terminal. The user is able to navigate through the different screens/actions in the application by inputting the numerical value corresponding to the next screen/action. A menu will be displayed each time an action is required showing the corresponding number value for the possible options.

Algorithm Description

Loading the json file: `load_json(json_file, port)`
To load the json file into the database, the file is read line by line. Each line read is stored into an array (batch). The function uses insert_many to insert all lines in batch once it reaches a set batch size (5000). Finally, after the last line is read, the function inserts the remaining un-inserted lines into the database.

Searching for tweets: `tweetSearch(col)`
The function first gets a list of comma separated keywords from user input. Then for each keyword, it finds all tweets that contain the keyword in the content field.
The search query matches all tweet documents with content field matching a regex, and projects the id,date,content, and user fields so only those fields are returned. To prevent

duplicate results, the resulting tweets are appended to an array, only if the array does not already contain the tweet.

Searching for users: `userSearch(col)`
The function first gets a keyword from user input. Then finds all tweets where the user's displayname or location matches the keyword.
The search query matches all tweet documents where the user.displayname or user.location fields match a regex, and projects the user field to get the nested user document. To prevent duplicate results, the usernames of the resulting users are appended to an array, only if the array does not already contain the user.

Listing top tweets: `listTopTweets(col)`
This function lists the top n tweets, based on retweetCount, likeCount, or quoteCount. The search query uses an aggregation pipeline. It first sorts the tweets in the collection based on the specified field, in decreasing order. Then it limits the number of results to at most n.

Listing top users: `listTopUsers(col)`
This function lists the top n users, based on user.followerCount. The search query uses an aggregation pipeline. It first groups the tweets in the collection based on the user.id, and computes the followersCount field for each user using the maximum of the followerCount in the user's tweets. The results are sorted in decreasing order of followersCount. Then it limits the number of results to at most n.

Composing a tweet: `composeTweet(col)`
Uses the insert_one command to insert a single tweet document with the date field being the current date, username field being "291user", and the content fields being the user inputted content. All other fields are set to None.

Testing Strategy

Test scenarios involved performing various actions within the application and ensuring the program behaves as expected. These include:

- Composing tweets and then searching for them
- Creating tweets with edge-case keywords to test search functions
- Listing top tweets and users under several databases
- Entering incorrect/unexpected input

The test cases performed should theoretically cover any series of actions a user might perform in the application.

Group Work Strategy

The project was first split into separate functions that could each be worked on independently. Each group member could then develop the separate functions as needed. Communication over clarifications and other coordination between group members was primarily over Discord.

Member	**Tasks**	**Time spent (hrs)**
Vero Bullis	listTopTweets(), listTopUsers(), show_user_fields(), show_tweet_fields(), testing, documentation	~6
Victor Kwok	composeTweet(), getN(), fixes, documentation	~2.5
Taran Purewal	Initial load_json, template code, tweet_search(), fixes	~3
Krish Rajani	userSearch(), batch implementation in load-json, various fixes/changes	~3