

TwitterButNotSwag Design Doc

Overview

TwitterButNotSwag is a program designed to allow a user to interact with a database created with a json file. Information is queried through mongoDB, allowing the user to search for users and tweets, as well as make their own tweets.

User Guide

To use TwitterButNotSwag, the user must indicate the name of the json file containing the database information. The user must also specify the port to connect with mongoDB.

Once the user successfully generates a database with the json file, the user is brought to a menu in which different activities can be chosen. The user can navigate through the program by entering the corresponding choice letters, or 'x' to go back.

Upon entering "t", the user can formulate a tweet to post. It will be posted under the username "291user".

Upon entering "st", the user may insert a keyword to search for tweets. Tweets matching the keyword will appear. The user may enter the tweet's number to view more information about the tweet.

Upon entering "su", the user may enter a keyword to search for users with the keyword in their display name or location. A user may be selected by entering their number on the sidebar to view more information about the user.

Upon entering "tt", the user may choose whether to view top tweets by retweet count, like count or quote count. The user can also select how many tweets they would like to see. The user may enter the tweet's number to view more information about the tweet.

Upon entering "tu", the user can view users with the most followers. The user may select how many users they see. A user may be selected by entering their number on the sidebar to view more information about the user.

Upon entering "x" at the main menu screen, the application will end.

Design/Algorithm Details

Our program is designed around the main menu for ease of use and ease of understanding for both users and developers alike. Different activities are separated into different files that handle everything involving that activity, from the querying of data to the processing of data and the display of data. Tweets and user information from queries are passed from their initial function into functions that handle their display. Users are also able to view more information about their search results through designated functions.

In our data handling process, we employed the use of mongoDB queries to find users and tweets by different criteria. These queries sort and limit results to limit data processing in the code itself. Based on the structure of mongoDB, querying documents is quick and allows for easy information retrieval. Tweets are searchable by several keywords which are queried to the database, similarly to singular keyword search. New tweets are simply inserted to the database as-is with the tweet text provided by the user.

Testing Strategy

For testing purposes, we used one of the smaller json files provided in the google drive to set up the basics of our program. Using this small database, we tested that tweets were able to be inserted, and that searching tweets and displaying top tweets work in their most basic functionality. As we started to implement more of the specific aspects, such as displaying x amount of users, we started working with the larger databases to test for time efficiency and if the correct values are displayed.

Test Cases (10.json)

- 1. Test tweet create and search
 - Create new tweet: #Life is a toughy
 - Search Tweet: Life (should display previous tweet)
 - Search Tweet: Pineapple (nothing should display)
 - Search Tweet: life (not a whole word, nothing should display)
 - Exit
- 2. Test user search
 - Search users 'kaur' (display KAUR SUKH user)
 - Select user: 1 (kaur) (display user info)
 - Search users 'india' (display 3 user info)
 - Select user: 3 (display user info)
 - Search users: ind (not complete word, nothing should display)
 - Exit
- 3. Test top users
 - Select view 10 users for top users (should display 10 in descending order)
 - Select user: kaur (display user info, max followersCount)
 - Exit top users
 - Select view 5 users for top users (should display 5 in descending order)
 - Exit

- 4. Test top tweets
 - Select top tweets -> based on retweets
 - Select view 10 tweets (should display 10 users)
 - Select top tweets -> based on likes
 - Select view 5 tweets (should display 5 users)
 - Select top tweets -> based on replies
 - Select view 3 tweets-> select tweet 3 (view tweet info)
 - Exit

Source Code Quality

Our program is broken down into separate files for ease of understanding. Each activity selected by the user has its own file in which most if not all functions required by that activity are handled. Variable naming is consistent and descriptive of what they are representing. The code is adequately organized with little to no unnecessary lines and with code blocks spaced apart for readability. Comments are descriptive and explain what is not immediately clear from a glance at the block.

Work Breakdown

Our group communicated through our project discord, actively discussing current bugs, issues, and tasks that needed to be resolved. Progress was tracked through our project planning document in which each member indicated their progress on different functionalities of the program. Tasks were highlighted in colors ranging with colors red, to orange, to green indicating completeness, and blue to indicate that the feature had been tested somewhat extensively. Regular group meetings were conducted to both work together and communicate about our progress.

All group members together spent 15 hours working together on multiple different aspects of the project, including debugging.

Zhasheken

Main focuses were creating the document store, database connection, creating the menu and creating tweets. Approximately 8 individual hours were spent working on implementing these features.

Japkirat

Main focuses were implementing tweet searching, top tweets, and top users. Approximately 8 individual hours were spent working on implementing these features.

Jenny

Main focuses were implementing user search, displaying user selection for user search and top user selection. Approximately 8 individual hours were spent working on implementing these features.