CMPUT291 - Fall 2020

Mini Project II

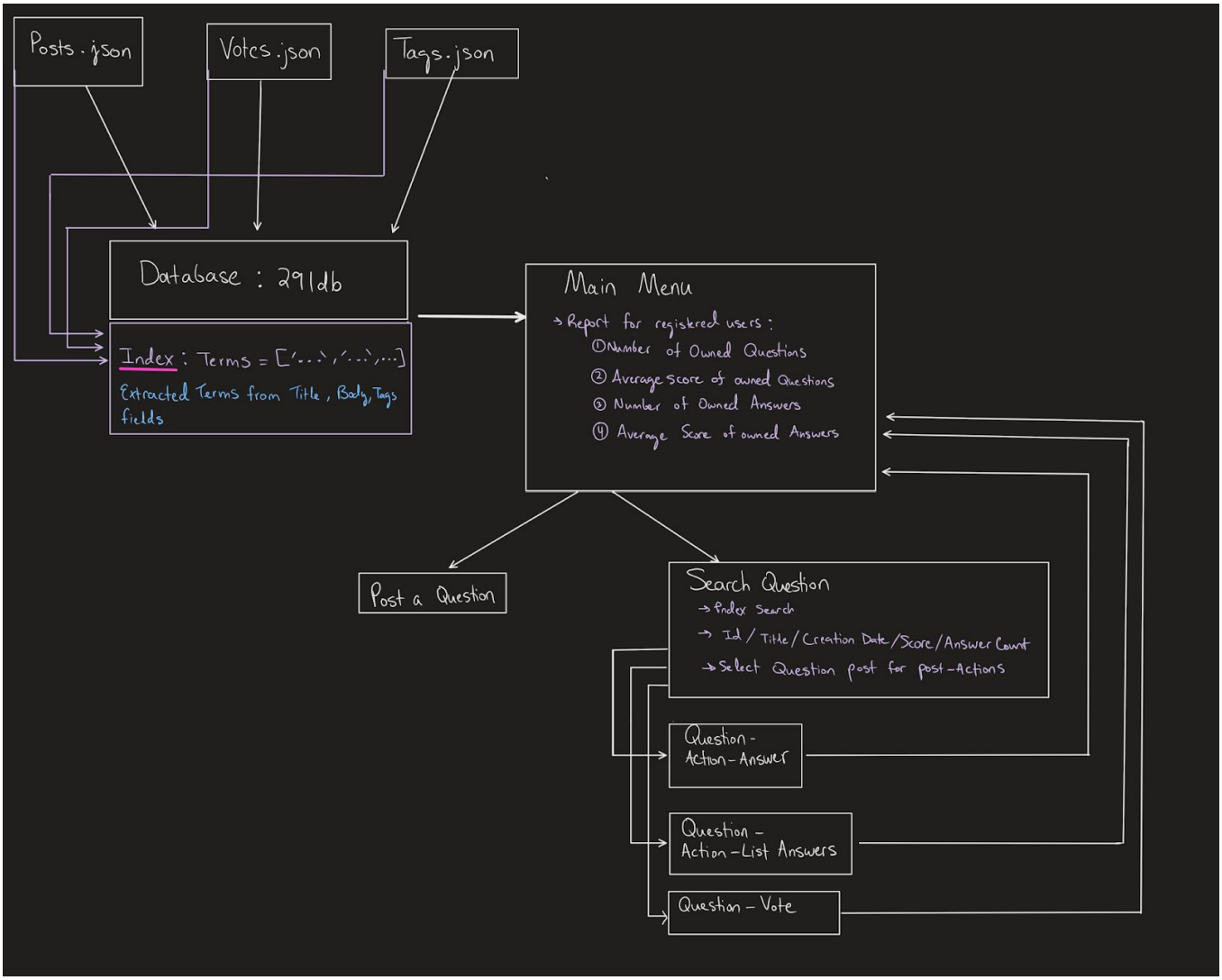
CCID: kooner, areez, sdhiman

User Guide

Firstly the MongoDB server must be started using the command: ***`sudo systemctl start mongod`***
Required files include Posts.json, Tags.json, and Votes.json.

The program will be run through the command line as follows: **python3 phase1j.py [port Number] and python3 phase2.py**. The port number can be any port that can run the MongoDB server and by **default is the port 27017 (not hardcoded).** From there the user will be asked to provide a userId for which a report showing the number questions owned, average score of questions, number answers owned, average score of answers, and votes casted. If no userId was specified then no report will be displayed. A main menu will also be displayed that will allow the user to post a question, search for a question, or exit program. If the user selects search the user can provide keywords which will be used to search all question posts. Once searched a post can be selected and then the user can perform the actions, answer a question post, list all answers of a question post, or vote on a post.

General Overview



Overall the design is a forum which allows users to post questions and then perform various functions such as answering questions and voting on posts. The flow of data throughout the program is given in the figure above which shows the streamlined outline.

SOFTWARE DESIGN CHOICES

In general when asking the user for input the program will loop until the user enters valid input and any matching that occurs in searching for terms will be case sensitive.

Phase One

The program must connect to a port to host the database server and if not already created it will create a 291db. Similarly when reading the collections info if they already exist the collections will be dropped before creating them once again. This allows the database to retain fresh copies of the data and no old data will be kept. For this step of inserting the collections by extracting the terms before inserting the json information performance of phase one drastically improved. Furthermore, the insert_many() method is preferred over insert_one() in terms of performance benefit resulting in a faster runtime. Lastly an index is created over extracted terms of length three or more from the title, body, and tags(useful when searching tags) fields of the Posts.json collection. This index will help improve the performance of search in cases where the index can be used.

Phase Two

The code for phase two is modular in the sense that it is split up into functions for each action that the user can perform on a given post. This design philosophy improves the organization of the code and increases simplicity as the function can be called when a given action is required. The user report and all other displays it the program outputted with newlines before and after the display to allow for more readability. The user can post a question which will insert a new post in the posts collection specifically in the nested array of the field row. From the **main menu** the user can also **search keywords** and a design specification chosen for this feature is that when the terms are length three or more the index will be used to query the database which increases efficiency. The user can select a question and the fields Once the results of the search are displayed and the post id was chosen to be displayed as well, in order to easily select a post. Also the user can select a post to display the full field values of the post. Afterwards a user can perform actions which include answering a question, retrieving the answer list for that question, and vote on a post.

Testing Strategy

To perform adequate testing on such a large volume of data a subset of the original data was sampled, in order to accurately compare results. This method of testing was more efficient that using a large dataset as it took less time to run the program and checkover results of the functions. Many edge cases where tested to ensure the program would run in various branches an example of such cases were testing that searching would work on title,body,or tags field as well as not being able to vote on the same post twice. Each function was tested to ensure that spec was being matched accurately. Some encountered errors were the proper splitting of terms array, passing the proper variables between functions, allowing for a no id user to be able to also use the post features.

GROUP WORK STRATEGY

Initial planning and design were done together to find a design that will best fit the programming tasks and discuss the eventual structure of the code. These discussions helped to outline the project as a whole and also clarify how we perceived the design spec and how to best approach it. To maintain the project Github and google colab were used in particular to provide a consistent version of the code among group members.

Smriti - Worked on the post question, post answer, and vote **(13 hours)**
Areez - Worked on the list answer and phase one optimization **(13 hours)**
Gurick - Worked on phase one, main menu queries, and search post **(13 hours)**
All members focused on testing and general debugging throughout the whole code