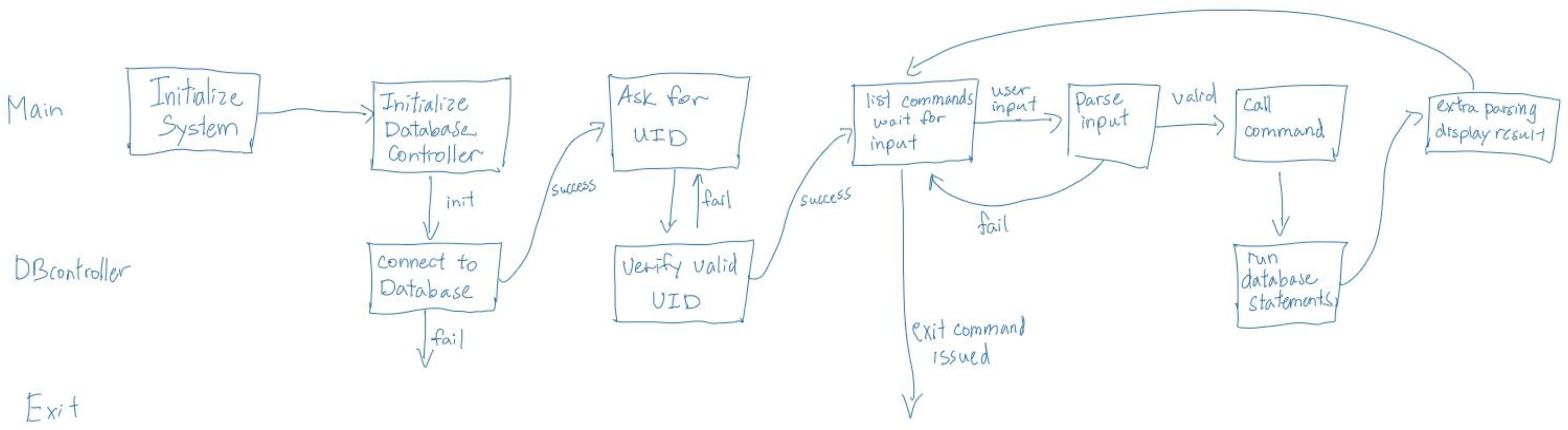

General Overview of the system:

The system begins in Main.java and starts by initializing connection to the database. After this connection is successful, we initialize a map of the available inputs and their corresponding function calls.

An input loop then begins and the user can now either select to use a uid or not, at which point they are asked to input their uid. From there they are given a list of possible commands to input. Each command function will then request any required user inputs and do any parsing that is required before calling the corresponding database controller function to interact with the database. After interacting with the database, the controller will return any data that is expected and required. The command function will then do any last minute parsing or displaying of the data to the user.

Upon completion of the steps above, the system will return back to the input loop and the user will be allowed to start another command.

When the user enters the exit command, the program will then break from the input loop and terminate the program.

Detailed Design

Note: minor/helper functions were not included.

DBController

This class functions as the connection between the database and the program, creating functions for fetching data in several forms and working with MongoDB to fetch data.

postAnswer

This method functions to take the information from the front end, formatting the answer post and inserting it into the database. This function generates the random id for the post as well as the date of the post.

postQuestion

This method functions to take the information from the front end, formats the tags and inserts them into the database, as well as formats the question post and inserts it into the database. It also generates the random id as well as the date of the posting.

Search

This function parses the keywords into mongo searchable strings. The function checks if any of the keywords is less than 3. Depending on this, different search methods will be called. If the keys are all greater than 3 then the function will search on index Terms. Otherwise it will return all posts and search for the keywords in Java.

Vote

Whether the user has a uid or not, the user can vote on a post. If the user has a uid, they won't be able to vote on a post more than once. But there is no such check for an anonymous user. This method inputs a vote into the database after generating the votes id and attaching it to the post it voted for.

<u>Main</u>

This class is used as the interface for the user to call on functions and navigate through the program

answerPost

This function can be only used when a post has been selected. The function will prompt the user for an answer. After, the function will take the user's id and based on the selected post will call postAnswer from the dataBase. If no post is selected or the post is an answer the method will reject the request.

postQuestion

This function will prompt the user to enter a title, body and tags. The function will then generate an id. The function will then call dbController.postQuestion to post the question.

searchPost

This function will ask for user keywords separated by a comma. The function will then parse the input and call dbController.search. Upon returning, the function will display the first 5 results and allow the user to scroll through the pages using ">". The user can also input "s" to select a post where the user will be prompted to input the id of the post. If the id is valid, the function will exit. The user also has the option to quit without selecting any posts by inputting "q". If a post is selected, a detailed summary of the post will be displayed.

vote

This function will check if a post is selected and if the user has already voted on the post. If there is no post selected or the user has already voted, the method will reject the request. If the user has no uid, they can vote as much as they like. The vote will be stored in the db, and the score of the post will increase.

<u>Testing Strategies</u>

The testing strategy that we chose to employ was largely one of dirty testing, seeking to break the system in any way we could, inputting incorrectly or under the wrong conditions to see how the program would react. Additionally, we would test each segment as we would complete it. Since our work is relatively compartmentalized, each segment of work could be tested thoroughly before adding it to the program. This resulted in the final product being very close to bugless once it was pieced together.

<u>Group Work Breakdown</u>

Andrew Chen

Andrew did the searching and posting of a question ~8 hours. He also did some testing ~2 hours. Andrew also drew the architecture diagram and contributed to the docs.

Ming Mao

Ming did phase 1 of the project and set up the initial architecture ~4 hours. He also implemented showing the user report and listing the answers to a question ~3 hours. Lastly he provided some support for members and conducted testing ~2 hours.

Ben Wagg

Ben worked on the answer question and the vote on post functionality. The answer question was ~ 3 hours, while the vote functionality was about ~3.5 hours. Beyond that Ben conducted 2 hours of testing and bug fixing.

For the most part, responsibilities were decided upon on a case by case basis. Core functionalities were tackled first, with each member ensuring that the other members had work, but were willing to step in to help in the event that another member was not able to work on that section or were struggling to complete it themselves.

Additional Details:

This project is coded in Java and uses maven to resolve any dependencies.