**USERS MANUAL**

In this manual I will show you how to use the search engine and how each part of the engine reacts with one another. The first part of this manual will describe the separate parts of the user interface and what they do. The second part of this manual will talk about what happens with each command in the user interface and how it interacts with the index and/or index interface.

For an in depth-description of every class there is a Doxygen folder on GitHub. Check index.html which contains descriptions for every class and what it does. In this I will only be talking abstractly on how data is handled and utilized.

**REGULAR USER INFORMATION**

**Interactive Mode:**

* The first thing that the interactive mode does is it checks if indexInterface.txt has been created, if not a new index file is created and the program continues as usual.

1. The user selects if they want the index loaded into a HashMap or an AVL tree interface.
2. Once the index interface has been selected, the user is prompted to enter a search query. It may start with the word AND/OR or with just one word
3. One word means that we will be returning the highest results that contain the word that was inputted.
4. If a user Inputs AND first, the program will take every word following AND and returns the highest results that contain all of the selected words
5. If a user inputs OR first, the program will then analyze all questions that contain any of the selected words and return the highest results.
6. At any point in the query the user can input the word NOT and the search engine will delete any questions that contain any of the words that follow the NOT keyword.
7. Once the query has been processed, a user can select any one of the questions out of the top 15 and display the questions full contents.
8. The user will then be prompted if they want to display any stats or do another search query. The stats are as follows…

Total Questions - this displays the total amount of questions that exist in the index file.

Unique Words – this displays the total amount of important words that exist in the index file for searching.

Top 50 Words – this displays the 50 words with the most index entries in order from most frequent.

**Maintenance Mode:**

* Maintenance mode handles adding files to the index and creating the new index file. It also allows the user to completely clear the index file.

1. The user is asked if they would either like to add new files to the index or clear it.
2. If user selects to add new files. The user will be prompted to input the directory containing ONLY tag files, then the user will once again be prompted to enter a directory containing ONLY question files.
3. After the user inputs the paths, the files are then saved to the index file.
4. The user may also select if the index file should be deleted, which resets the program. However the paths the user selected to add will still be created on the next creation of the index file.

**MANAGEMENT PIECE**

* This part takes user information, but goes more in-depth on what happens and what to fix if something is not working as expected.

**Interactive Mode:**

Creating new index interface

* The first thing that is prompted is if the index file exists. Do not worry about loading the index manually as the program immediately creates a new index file. However if the file does not exist and the program terminates, look into errors in parsing. There may have been a strange error when reading in an unknown type of file in the reader class. Double check and make sure that all of the tag files and question files are formatted correctly and continue.
* Once the index file has been created the user will be prompted to enter a Boolean search query. The first thing that happens is the queryProcessor takes the user input and chunks it into separate words and separate containers for each word that follow a(n) AND/OR/NOT keyword. The query processor then takes each individual word and looks for that word in the index file, it then stores every ID# and score value that follows the word. *If some results are appearing twice, make sure that there are not any repeated files in the questionNames.txt or tagNames.txt files since these will store each of the files twice.*
* Once all the words have had their IDS stored and their scores for each word, then we take all of them together and if the query was preceded with an OR keyword, we just combine all of the scores and IDS together into one container and sort it in order of the top results.
* If the query was preceded by AND we use a separate container that counts how often each ID appears and to be stored in the final container, the container must have the same number as the total number of words that followed the AND keyword.
* If the query contains a NOT keyword, we do the same process as the OR words were handled except these are stored in their own separate container. We then take our result container from the AND words or OR words and subtract any IDS that exist in the NOT container. *If there are still results showing up that should have been deleted, double check and make sure that the containers are storing the iterating through all the IDS correctly, since a incorrect container order will cause the deletes to not work as expected*.
* The user will then be prompted to read one of the top 15 results completely. *This is handled completely by the reader class. If the ID is not appearing correctly then that means that the reader class is running into an error with a specific question which causes IDS to skip. This should not happen. Double check and make sure that all files are formatted correctly as the parser stops at the <>!<>!<> delimiter at the end of a question.*
* The user will then be prompted to select stats to display. These are all stored in the first two lines of the index file. This is pretty straightforward since there is nearly no interaction with the other classes to retrieve this information. The index file has first line of total questions, total words, and all words(used for maintenance mode) respectively, and the second line is all the top 50 words in order.
* If the user enters an incorrect value for stats, the program breaks and interactive mode ends.

**Maintenance mode:**

* This mode mainly handles storing of new information and computing the new index file.
* The user will first be prompted to add new files to the index. Both tags and questions are handled very differently which requires the two of them to be in different folders.
* Tags will be read from a directory with only tag files, then the tagNames.txt file is updated, and then the UI prompts the user to enter a question directory.
* The question directory must be separate from the tags directory. The filesystem reads the directory and updates QuestionNames.txt
* Then the reader reads ONLY the new files that the user added which then means a new index file has to be created.
* The program has serialized files that contain all of the previous words and their scores, the reader then has a flag that this time we’re adding new files, so the program then loads the serialized values into the respected data sets to make parsing files much faster. We also load the first line of the index file to receive the total amount of words, so computing TFIDF scores is easier when adding in new information.
* *If the new index file contains only the new information, double check and make sure that there was another index file created before the new files were added, otherwise the old information will not be serialized into their respective containers, which results in only the new information in the index file.*
* The user will also have the option to clear all the important files. This will not only clear the index file, but also the serialized text and tag files as well.