**SOFE 4790U**

**ASSIGNMENT 1**

**Plagiarism Checker with Built in File Information Loader**

**Matthew Brown 100670025**

Text

Description automatically generatedI wanted to develop a plagiarism checker that compares the similarity between two files within a directory and a way for users to return certain information about a file such as name, size, etc. Plagiarism is a very huge issue in the world we live in today and it also prevalent in schools as well. This functionality can be used, and hopefully improved upon in the future, to flag plagiarism and encourage the submission of honest work. The second functionality is very simple in which will write certain information about a file and save it to a file on the client side.

***This is the plagiarism checker in action. Although the result should be 1 not 0.6 because the files are identical. A minor arithmetic error.***

Text, letter

Description automatically generatedI had an abundance of issues when completing this assignment because my initial idea kept failing. I wanted to send two files sequentially, via user input through the socket connection and save them to variables on the server side. Unfortunately, after hours of research and countless errors, I discovered that it is only viable to send one stream of data through the socket at a time. Whenever I sent the two files, they would be saved to the same buffer and written to the connection and saved within the same variable. I was frustrated because my idea appeared to be so simple and yet it was not possible. I decided to simplify my idea and just send one file over the socket and compare it to a file that can be read on the server end. This way I could still achieve what I wanted to do, and I would not have any issues.

***This was one of the preliminary issues I had. The content of my files would be saved to the same variable and then printed.***

Now for the overview of my code, it is quite simple but quite elegant at the same time. I prompt the user with a question of which service they would like to use. They would type “1” for plagiarism checker and “2” for file info. I create the usual socket connection on a designated port and that will establish the connection between server and client codes. The Client side when, option 1 is selected, will take the name of the file and save it into a file object. This will be passed to the FileInputStream object will be written to the socket connection using a BufferedOutputStream object. After is done, the server side will be implementing some code that looks a little complicated but is not very hard to grasp. My server is multithreaded to handle connection from multiple clients. Although it does still need to be refined, it is enough at least for this assignment. I instantiated two instances of the BufferedReader class and had one acquire data from the socket and the other one simply has a FIleReader that will read a specified file to compare from the client. This is where I needed to get creative, to compare the files, I wanted to place them in an ArrayList. So first, I needed to save them both to a string. The input from the socket was stored using the readLine(0 method. But The information from the file could not be saved in this same way. I needed to instantiate an instance from the StringBuilder class that read added words from the file and eventually converted that to a string. Now that was finished, I split the strings at each whitespace to convert them to an Array and the I finally converted them to an ArrayList. I chose the List object because I came up with a clever way to determine the percentage of plagiarism. I create a blank list and wrote an enhance for loop that runs through the first list. For every word that is In the first list that also matches the second, it will be added to the empty list. But using this method would create duplicate entries which would throw off the percentage. To rectify this, I created a LinkedHashSet which by default removes any redundancies in the list. I passed this back to an ArrayList and now it is finally time to calculate percentage. The equation is simple, the size of the NoDuplicate list is divided by the size of myList2. This works because the large NoDuplicates is, the larger the answer is which indicates the majority of words from the first file are indeed present in the second file.