### ÇANKAYA UNIVERSITY

### SOFTWARE ENGINEERING DEPARTMENT

### SOFTWARE PROJECT I



|  |  |
| --- | --- |
| **Name Surname** | **Ferhan Burak Özkan** |
| **Identity Number** | **202028018** |
| **Course** | **SENG 383** |
| **Experiment** | **Search Engine with LinkedList** |
| **E-mail** | **c2028018@student.cankaya.edu.tr** |

# **PROBLEM STATEMENT**

**Goals of the Programming Assignment:**

The goal of this programming assignment was to create a simple search engine application that can load documents from a file, search for specific terms in the documents, remove occurrences of a specific string from a text file match certain criterion.

**Inputs:**

1. Loading Documents: Application anticipates the user providing a path to an info file containing the documents to be loaded into the search engine.
2. Searching Documents: Users enter comma-separated string search query and an output file location to save the search results.
3. Removing Documents: User provides the document to be deleted as well as the file location from which the document should be removed to remove it.

**Outputs:**

Application generates many forms of output:

1. Search Results: After searching for documents, the program creates an output file containing the documents that were found.
2. Removed Documents: Remove documents, the program updates the original file by excluding the documents you specify.

**Error Handling:**

The program should handle invalid input gracefully. If the input file does not exist, the program should print an error message and terminate. If the user does not provide exactly two command-line arguments, the program should also print an error message and terminate.

**DESIGN**

**Design Decisions:**

* The program is designed to use a custom LinkedList data structure to store and manipulate the documents.
* For searching, it employs a simple algorithm that iterates through the documents and checks for the presence of search terms.
* The program uses simple menu system to create a ease for user.

**Data Structures:**

* LinkedList: I used a LinkedList to store the lines of the input file.

**Algorithms:**

* Used a while loop to iterate through the lines of the input file and a for loop to check each line for the presence of the specified string.
* Loading Documents: Iterating through lines of the info file and splitting documents based on "<" and ">" tags.
* Searching Documents: Recreating through documents and looking for the presence of search terms.
* Removing Documents: Removes certain items from the file and overwrites the file with the new version.

**Pros/Cons of Choices:**

The choice of using a LinkedList allowed for efficient addition and removal of lines. However, accessing a line in the middle of the list would be inefficient compared to using an array.

**IMPLEMENTATION DETAILS**

**Flowcharts:**

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, çizgi, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Implementation Process:**

* Started by establishing the LinkedList data structure and the relevant functions linked to it.
* Wrote code to load content from files, segmenting it according to specific tags.
* Created a search mechanism that involves going through documents and finding matches for the search terms.
* Implemented the ability to eliminate specific documents by reading all documents, excluding the ones meant for removal, and rewriting the file.
* Designed a straightforward command-line interface to facilitate user interaction.
* Incorporated error handling for both file input/output operations and user inputs.

**Adaptation:**

Added code to read the input file line by line, check each line for the presence of the specified string, and remove the line if it contains the string, also added code to create a new

LinkedList and to add the lines to the LinkedList.

**Development Timeline:**

The implementation process was completed over the course of a few days:

* Day 1: Data structure setup and document loading.
* Day 2: Search functionality and error handling.
* Day 3: Document removal and menu.
* Day 4: Testing

**Testing Notes**

**Testing:** I tested the program by running it with different inputs, including empty files, files with only one line, and files with multiple lines, some of which contain the specified string and some of which do not.

**Normal Inputs:** Tested the program with input files that contained various combinations of lines with and without the specified string.

**Special Cases:** Tested the program with invalid input, such as providing more or fewer than two command-line arguments.

**Sample Input/Output:**

1. **Load Function**

**Input:**

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. **Search Function**

**Inputs:**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Outputs:**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. **Remove Function:**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Before:

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

After:

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Result:** The program worked as expected in all test cases.

**Comments**

**Overall Result:** The programming project was a success.

**Future Improvements:** Next time, I would consider using an array instead of a LinkedList, as it might be more efficient for this type of problem. Adding a graphical user interface (GUI) may be more user-friendly.