Design Change Report

**Table of Contents:**

1. Introduction

2. Functional Requirements

3. User Case Diagram

4. Test cases

**Introduction:**

This report specifies the design changes that are done successfully after the first phase of implementation.

The two requirements after the first phase are as follows:

**Second Requirement:**

The second requirement is to allow the replacement of all occurrences of a given word to a given replacement word. Note that the replacement happens only when the given pattern word matches with a whole word. For example, for text “ab cd ef”, replace “a” with “b” will result in no change, while replace “ab” with “cd” will result in “cd cd ef”.

**Third Requirement:**

The third requirement is to add a feature called grepline. This feature will take a document and keyword as its input and returns the lines in the document that contains at least one occurrence of the keyword. Note that as the second requirement, the match of keyword is the whole word only. So, ‘ab’ has one occurrence in ‘ab cd ef’, but no occurrence in ‘abc de ff’.

**Use Case Diagram:**

**Before Change:**Diagram

Description automatically generated

Before the 3rd requirement, the use case diagram has the blocks shown above in which the functional requirement grepline was not included.

In our project, I have designed it in an operational way such that each function can be used separately and gives us the best result with the proper input. This will also avoid bugs.

**After Change:**

Diagram

Description automatically generated

The above user case diagram will provide the information about the operations needed to perform on the given input file and finalize the required output.

Here the 3rd requirement is added and finalized with the three requirements. So, from here the operations.py will include all three requirements for the user.

**Functional Model:**

Diagram

Description automatically generated

The following diagram is the functional model of the project after the changes. Here we have mentioned the operational flow of the input file i.e., the Text file.

These operations should appear as operations in individual or should be supported by a combination of operations. In this, all these processes are reflected as operations on individual files and are incorporated into the design.

For these requirements, the major design change is in the operations. It has been included with this requirement implementation and provided as a function to work for different types of input files.

The functions take the input by providing the message on the command line and do the operation accordingly.

**General Test Cases:**

|  |  |  |
| --- | --- | --- |
| **Input Type** | **Description** | **Estimated Result** |
| Valid Input | Providing the file name  (Filename.txt). This should be in the same folder | Printing the unique words. |
| Valid Input | Providing the filename and replacing word and replaced word. This should be in the same folder | Print the unique words and write back the replacement operation to the file. |
| Valid Input | Provide the word to search and filename. This should be in the same folder | Print the unique words, write back the replacement operation, and returns the list of lines. |
| Valid Input | Provide the valid filename but it is not in the same folder. | Error. |
| Invalid Inputs | Providing the invalid filename | Error. |

* The Executable Test cases are provided in a separate Excel file [**Unit\_test\_cases.xlsx**].