**Design Change Report**

**Project – Word Statistics**

**1.Abstract** – Thisproject has been developed in three phases.

**First Phase:** Develop the project which does count the frequency of all the distinct words.

**Second Phase:** This was the first design change to Phase one requirements to enhance the capabilities which returns the number of lines and characters in the input file.

**Third Phase:** This phase consists of a second requirement change to replace 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”.

**2.Implementation Design-**

**2.1 Initial Requirement Design Steps-**

2.1.1 Used input text file to get input text. Read the text file.

2.1.2 Passed it to function **get\_unique\_words\_frequency**(input File) – read the file line by line.

2.1.3. First convert the input into a list of words

2.1.4. iterate through the list of input words to find the unique words.

2.1.5. check for duplicate strings.

2.1.6. if a unique word, then add it to the unique\_string\_list

2.1.7. Iterate through the unique\_string\_list

2.1.8. if a unique word then add it to the unique\_string\_list list

2.1.9.

5.Passed List LINES to function getWords(LINES).

6.Iterate over each line and SPLIT all lines based on separator “ ”(space), “.”, “-”

7.Add all spilt words to List- **WORDS**

8.count each word length in list-WORDS – add each length- totalCharInFile.

9.Print totalCharInFile- total number of Char in file.

**2.2.First Requirement Design Steps-**

1.Used WORDS list created in function getWords()

2.Pass WORDS list to getUniqueWordsWithCount(WORDS)

3.Create a MAP with String as Key , Integer as Value – **UniqueWordCountMap**

4.Create a MAP with Key as Integer , String as Value – **AllWordwithLocationMap**

5.it will add WORD to **UniqueWordCountMap** as key and count its occurrences and add to value of map.

6.add all words and its location to **AllWordwithLoactionMap**. Key is location of Word and value is word itself.

7.print key and value of **UniqueWordCountMap –** it will give all unique word and its count.

**2.3.Second Requirement Design Steps**

1.Use **AllWordwithLoactionMap** map from above function and pass it to replaceSamePatternOccurence(AllWordWithLocationMap).

2.Iterate over and find same pattern words in above map. Replace the value with first occurrence location word.

3.Create String and add one by one all value from map according it its location in file.

4 Print above String – it will be give output – text with all replace same pattern occurences.

**3.Design Tools-**

1. Programming Language – JAVA.

2. Eclipse

**4.Testing Tools-**

1.Junit- Unit testing

2.Static bug detection – SonarLint(Eclipse Plugin) & SonarQube

3.Code coverage – EclEmma(Eclipse Plugin)

4.Code Clone detection – SonarQube

**5.Conclusion-**

Implementing this project, helped to learn different steps of software process – Analysis, Requirement gathering, specification , implementation , Testing .