# INTRODUCTION

## Identification

This Software Requirements Specification (SRS) document establishes the overall software requirements for the Word Statistics software being created. This SRS can be used as a basis for design and testing of the Word Statistics software.

## Software Overview

“In this project you are writing a program to perform various word statistics of a given document (as a string). The initial requirement is to count the frequency of each unique word. The code should support combinations of space, tab, and newline characters as separators.”

UPDATE: “Provide two more features: Counting the number of lines (LineCount) and Counting the number of characters “

UPDATE #2: “The second requirement change is to allow 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.”

This is the guideline and requirements handed down by the instructor and it will be the leading developmental guidelines throughout the software’s lifecycle.

**2.0 REQUIREMENTS**

**2.1 Functional Requirements**

* This program is to take a block of text from the user through an input initially prompted by the program and count every unique combination of letters and symbols that form a string that is separated by white space, count the total number of characters and count how many lines are read. The program will allow the user to replace all occurrences of a given word with a replacement word. The program will output each word read and output the number of times each unique word was read. The program will output the number of lines that appear in the string given and the total number of characters in the string.
* White space is to be considered a space, tab or newline.
* The program will treat a “word” as any combination of characters (a string) that are separated by white space.
  + The characters used can be any eligible char that the user can input, such as but not limited to letters in any language, numbers or symbols.
  + Capitalized and lowercase characters are considered different and thus two strings with identical characters excluding a capitalized character and lowercase character will be considered different. (ex: “Test” and “test” will be considered different strings.)
* The input of words will be given by the user manually upon running of the program. The program will not read from a file of any sort, the program will strictly read from the input given by the user at the beginning of the program
* A “word” can be any number of characters. There is no limit on the amount of characters per “word”.
* Output will be presented in the console (or equivalent depending on compiler used.) via a print statement. Each unique “word” will have a value associated with it for how many times it appears in the given input text
* The output of the code will include counting of any lines. Since this code's input is a string and is written in Python, a line is counted by splitting the string inputs “\n”. This is the only way to incorporate a newline and count lines with this program.
* Blank lines will not be considered
* Normal lines will not be counted, “\n” will be the only way the program can identify a line.
* This program will also count every character used. This is any character at all, including symbols, non english letters, and white space.
* The total amount of characters will be the number of every single character within the given string.
* Multiple words are replaceable.
* All entries are case-sensitive.

**2.2 Non-Functional Requirements**

* There are no explicit non-functional requirements. The code will not have any memory leaks or segmentation faults.
* The code will be extremely usable, requiring only a minor read-me that explains how to use the code and what will be expected from the code’s output.
* There will be almost an infinite mean time between failures, as the code will be simple enough to not use a significant amount of memory or resources.
* There is no requirement for the compiler used for the program. All that matters is that it is a python compiler.

**2.3 Test Case Requirements**

* There will be a test case to test that the code works with a “word” composed of English letters.
* There will be a test case to test that the code works with a “word” that contains only numbers.
* There will be a test case to test that the code works with a “word” that contains only symbols
* There will be a test case to test that the code works with a “word” that contains English letters and numbers.
* There will be a test case to test that the code works with a “word” that contains English letters and symbols.
* There will be a test case to test that the code works with a “word” that contains symbols and numbers.
* There will be a test case to test that the code works with a “word” that contains English letters, symbols and numbers.
* There will be a test case to test all white space (newline, tab, space).
* There will be a test case for treating capitalization as its own unique character. (ex: “Test” and “test” will be considered different strings.)
* There will be a test testing the longest word in the English Dictionary “Pneumonoultramicroscopicsilicovolcanoconiosis” and “No.” This test will prove that any length of string is possible.
* There will be a test case that will ensure that the total amount of characters is being counted correctly.
* There will be a test case that counts the proper amount of lines
* There will be a test case to ensure only an “\n” is being counted as a line.
* Test the "replace\_word" function with a sentence containing one occurrence of the pattern word.
* Test the "replace\_word" function with a sentence containing multiple occurrences of the pattern word.
* Test the "replace\_word" function with a sentence containing the pattern word as part of another word.
* Test the main function by inputting a sentence and answering "No" to the word replacement question.
* Test the main function by inputting a sentence and answering "Yes" to the word replacement question and replacing a single word.
* Test the main function by inputting a sentence and answering with an invalid answer to the word replacement question.