**CHAPTER 5**

**SOFTWARE REQUIREMENTS SPECIFICATION**

**5.1 External Interface Requirements**

5.1.1 User Interface

* Interface for typing the document directly – Editor Window
* Interface for saving the file – Save Dialogue Box
* Interface for entering text into Debate format – Debate Dialogue Box
* Interface for entering text into Formal Letter format – Formal Letter Dialogue Box
* Interface for entering text into School Essay format – School Essay Dialogue Box
* Interface for viewing generated statistics and comments – Editor Window upon pressing/clicking the relevant buttons/options.

5.1.2 Software Interfaces

* Operating system : 64–bit linux 2.4 or 64-bit windows
* Language : Python
* NLTK package for python
* PyQT version 4.10.4

**5.2 Functional Requirements**

5.2.1 Applying rules from Elements of Style and giving suggestions

5.2.1.1 Introduction: The software should be able to detect wrong usage, common errors, etc. based on the rules mentioned in Elements Of Style and give suggestions accordingly.

5.2.1.2 Input: The text being typed by the user or an existing text document will be the input for the software.

5.2.1.3 Processing: The software will analyse the text after every few minutes and check for its conformance to the rules of style. If the text does not follow these rules, the software will generate comments associated with the respective line numbers for which the comment is meant. It will also give suggestions for such cases, where it is feasible.

5.2.1.4 Output: The output will be a set of comments generated by the software based on the text written and its conformance to the rules of style.

5.2.1.5 Error Handling: The software will ensure that it doesn’t crash for invalid input. Languages and scripts other than English and Alphabet will be ignored so as to not generate comments that are not applicable for the text.

5.2.2 Evaluating the readability score of the document

5.2.2.1 Introduction: The text will also be analyzed for its readability and check if it matches the target readability of the document.

5.2.2.2 Input: The text being typed by the user or an existing text document along with the target readability will be the input for the software.

5.2.2.3 Processing: Based on different formulae for readability score, the software will analyze the text, and generate its readability score. This will then be matched with the target readability score and if the difference is huge, a comment will be generated to inform the user of the same.

5.2.2.4 Output: Various readability scores of the document. Comments if any of it doesn’t match the target readability.

5.2.2.5 Error Handling: Since the readability test is evaluated for American English, words that are not found in the CMU Dictionary are ignored.

5.2.3 Giving Suggestions to make the text conform to the target writing style

5.2.3.1 Introduction: If the user has selected a particular writing style, the software will check its conformance to the writing style chosen and give suggestion as the user types.

5.2.3.2 Input: The text being typed by the user or an existing text document, along with the style of writing chosen for the document will be the input for the software.

5.2.3.3 Processing: Different rules will be hardcoded for each style. The software will analyze the text based on the rules of the writing style chosen. Comments will be generated accordingly.

5.2.3.4 Output: The output will be a set of comments generated by the software based on the text written and its conformance to the rules of the writing style chosen by the user.

5.2.4 Evaluating the floweriness of the text

5.2.4.1 Introduction: The software will analyse the text for its floweriness.

5.2.4.2 Input: The text being typed by the user or an existing text document will be the input for the software.

5.2.4.3 Processing: Based on the number of adjectives and adverbs present in the text, the software will evaluate its floweriness. Based on that, comments will be generated.

5.2.4.4 Output: Comments about the floweriness of the text.

5.2.4.5 Error Handling: Languages and scripts other than English and Alphabet will be ignored so as to not generate comments that are not applicable for the text.

5.2.5 Evaluating the obscurity of the text

5.2.4.1 Introduction: The software will analyse the text for its obscurity.

5.2.4.2 Input: The text being typed by the user or an existing text document will be the input for the software.

5.2.4.3 Processing: Based on the frequency of each word in the text, the total number of rare/difficult words will be calculated. This number, divided by the total number of sentences, will be displayed as the obscurity of the text

5.2.4.4 Output: Comments about the obscurity of the text.

5.2.4.5 Error Handling: Languages and scripts other than English and Alphabet will be ignored so as to not generate comments that are not applicable for the text.

**5.3 Non-functional Requirements**

5.3.1 Usability

The product will be a stand-alone application that will run on windows platform. It can be used on any machine that has Windows OS version 7 onwards. It will be useful for students, writers or anybody who needs to write a document in a particular style.

5.3.2 Reliability

The software will work only for English language and is not meant to be used for any other language. For certain functionalities, the usage is to be limited to American English only.

5.3.3 Implementation

The implementation of the software will be in Python language. NLTK framework will be used for all language processing work and NLTK Data comprising of corpora such as Brown, Wordnet and dictionaries such as CMU will be used for referencing purposes.

5.3.4 Interface

The software will have a GUI for the text editor to facilitate input of various parameters (Target readability, writing style, etc), text to be processed and the comments and suggestions generated by the software.