**CHAPTER 10**

**INTEGRATION**

**10.1 Integration strategies**

The development part of this project follows Bottom up approach. The bottom up approach is used for building small components, and finally to integrate them into one complete system. Initially the backend implementation was done. Each small component was implemented in python. API's were defined for better interaction and better integration with the frontend as well as for interaction between the backend modules.

To integrate all the components, top down integration approach is used. Top down approach is better for integration, and for every scenario, the corresponding backend API to be used, is also known. The User interface functionalities are linked with backend one after the other one, in a sequential manner. For the frontend, PyQT has been used and the backend coding has been done using Python. As python was used, integration was a challenging task. No python frameworks such as Django were used.

**10.2 User Data Transfer**

The data input by the user either by loading a file in the editor window or typing directly is passed to the python back end when the user chooses to evaluate the text.

This data is tokenized into sentences using the tokenise functions provided by NLTK.

The tokenisation is first done at a sentence level and then on a word level.

Get Data from Editor Window

Tokenize the data

Data in tokenized form

***Fig. 10.1*** *User data transfer*

**10.3 Word frequency analysis**

Once the data has been tokenised into words, each word’s frequency is fetched. This is done by loading the frequency distribution from the Brown Corpus. Each token’s frequency is then read from the distribution.

Data from the user

Load the frequency distribution of words from Brown Corpus

Fetch frequency of each token

Process obscurity

***Fig 10.2Fetching Frequency***

**10.4 Fetching syllable count**

Once the data has been tokenised into words, each word’s syllable is fetched. This is done by loading the syllable count from the CMU dictionary . Each token’s syllable count is added to get the total number of syllables, which is used for evaluating readability.

Data from the user

Get the number of syllables for each word from CMU dictionary

Total number of syllables

Calculate readability

***Fig 10.2Fetching Syllable count***

**10.5 Display of Comments**

The comments generated by the python back end are transferred to the editor window as a list. A list widget is used which displays each comment as a list item. Based on the severity of the comment, the comments are sorted.

Generate comments

Sort comments based on severity

Python Back end

Display as a list on the UI

***Fig.10.3*** *Display of Comments on the UI*