# CHAPTER 4

# SYSTEM IMPLEMENTATION

## 4.1 PREAMBLE AND PURPOSE

## 4.2 IMPLEMENTATION

### 4.2.1 IMPLEMENTATION MODEL

#### 4.2.1.1 PACKAGE DIAGRAM

#### 4.2.1.2 COMPONENT DIAGRAM

#### 4.2.1.3 DEPLOYMENT DIAGRAM

#### 4.2.2 SYSTEM REQUIREMENT

#### 4.2.2.1 IMPLEMENTATION TOOL

#### 4.2.2.2 CHOICE OF LANGUAGE

#### 4.2.2.3 HARDWARE AND SOFTWARE REQUIREMENT

## 4.3 TESTING

### 4.3.1 UNIT COMPONENT

#### 4.3.1.1 TESTING

#### 4.3.1.2 INTEGRATION

#### 4.3.1.3 SYSTEM OR FUNCTIONAL TESTING

#### 4.3.1.4 STRUCTURAL TESTING

#### 4.3.1.5 PILOT TESTING

## 4.4 COMPARATIVE EVALUATION

### 4.4.1 RESULT VS EXISTING METHOD

### 4.4.2 RESULT VS REVISED WORK

## 4.5 USER DOCUMENTATION

## 4.0 OVERVIEW

This chapter gives the detailed views of the implementation of this project. Section 4.1 presents the implementation phase, showing the components and software tools used for the development of this project. Section 4.2 presents the testing phase showing how the objectives described in Chapter One has been delivered. Section 4.3 presents the performance evaluation results, this shows the comparisons between the existing system.

## 4.1 IMPLEMENTATION

This section presents the system implementation models, tools, and the choice of programming language adopted for this project. It also presents the hardware and software requirements necessary for the implementation phase of this phase of this project.

### 4.2.1 IMPLEMENTATION MODEL

There are 3 different models presented in this section and they are: Package diagram, component diagram and deployment diagram. These models give an abstract structural view of the summarization system.

#### 4.2.1.1 PACKAGE DIAGRAM

This shows the \_\_\_ packages that make up the\_\_\_\_, it shows the different classes under each package. A package diagram depicts the dependencies between the packages that make up the system. It shows the structure, arrangement and organization of model elements. The model elements in \_\_\_\_\_ include the Language resources, Text processing resources and Quality assurance application. Fig \_\_\_ shows the Package diagram for \_\_\_\_ which depicts the dependencies between the three packages. As shown in Figure 4.1, the Language resource which comprises of the software requirement document is dependent on the Text processing resource and Quality assurance layer.

#### 4.2.1.2 COMPONENT DIAGRAM

Component Diagram helps to model the individual components and how they are wired together to form the \_\_\_\_. Component diagram describes the static implementation view of the \_\_\_\_\_\_, it shows the various components of the \_\_\_\_ which includes packages and components that make up each package. Figure 4.2 shows the component diagram of the \_\_\_\_\_\_.

#### 4.2.1.3 DEPLOYMENT DIAGRAM

Deployment diagram also helps model the physical aspect of an object-oriented software system. It models how the system works during run-time in a static view and visualizes the distribution of components. Fig 4.3 shows the deployment diagram for \_\_\_\_. It shows the first interaction point which involves the requirement analyst interacting with a computer device which has GATE installed on it. Requirements documents are loaded by the requirements analyst and it is analysed by processing through the application layer, the application layer gets it processing resource from Text Processing layer.

#### 4.2.2 SYSTEM REQUIREMENT

#### 4.2.2.1 IMPLEMENTATION TOOL

The following tools were adopted for the implementation of this project.

#### 4.2.2.2 CHOICE OF LANGUAGE

The choice of programming language used for the backend is Python because Python has a number of rich libraries for Machine Learning and specifically NLP. For the frontend, Javascript would be used as this is the language for the web.

#### 4.2.2.3 HARDWARE REQUIREMENTS

The following are the Hardware Requirements necessary for the system

* Processor: Intel Pentium
* RAM Size: 4GB RAM
* Disk Space: 20GB free HDD space
* System Type: 64-bit OS

#### 4.2.3.4 SOFTWARE REQUIREMENTS

The following are the software requirements for the system

* Operating System: Windows 7
* Python 2 or 3

## 4.2 TESTING

### 4.3.1 UNIT COMPONENT

#### 4.3.1.1 TESTING

#### 4.3.1.2 INTEGRATION

#### 4.3.1.3 SYSTEM OR FUNCTIONAL TESTING

#### 4.3.1.4 STRUCTURAL TESTING

#### 4.3.1.5 PILOT TESTING

## 4.3 COMPARATIVE EVALUATION

### 4.4.1 RESULT VS EXISTING METHOD

### 4.4.2 RESULT VS REVISED WORK

## 4.4 USER DOCUMENTATION

NLTK for the pre-processing of natural text (i.e., gathering, wrangling, stemming, POS-tagging, filtering and ‘noise’-reduction), GENSIM as kind of base platform (for autoencoding, semantic *(topics) and* syntactic (*sequence*) pattern- and as such for similiarity- recognition, dimensionality reduction, and for multilabel classification), and SKLEARN, which easily can be mixed up with NLTK and GENSIM, for third step evaluation / ensembling / optimizing / processing issues.

* NLTK is used primarily for general NLP tasks (tokenization, POS tagging, parsing, etc.)
* Sklearn is used primarily for machine learning (classification, clustering, etc.)
* Gensim is used primarily for topic modeling and document similarity.