# Chapter 6

# Implementation

## 6.1 Introduction

The previous chapter stated the major modules of the system and their functionalities. It also describes interaction among the components. The current chapter explains how those components are implemented using different technologies.

## 6.2 Image Processing Module

Even though this module could consider as an external module of the system, performing the functionalities in this module is consider as essential task due to the noise availability. The quality of the character recognition degrades dramatically due to the noise that exists in the image. By applying various image processing techniques the noise that is available throughout the image could be minimized for certain extend.



The human user is instructed to perform the following steps during the pre processing stage.

1. Blur image.
2. Convert the image to grey scale.
3. Invert the colours.
4. Apply the thresholding technique.

All of the above mentioned image processing techniques could be applied though the GNU Image Manipulation Program (GIMP).The threshold value for the image could be changed very easily by using the slider that is available in the GIMP tool. The user is allowed to set the threshold value by observing the output. The input for this module is a digitize inscription image and the output is an inscription image where the noise is minimized.

## 6.3 Pre – Processing Module

This module could be categorized into two sub modules - Character Segmentation Module and the Feature Extraction module. In the character segmentation module the horizontal and the vertical projection graphs are sketched by analyzing the black pixels in the inscription image Figure 6.1 and Figure 6.2 present fragments of the vertical and the horizontal projection graphs. The system will consider the total number of black pixels in each row in the horizontal projection graph and if it exceeds the user defined horizontal threshold value it is considered as a segmentation point. The system will fragment the line segments base on the identified segmentation points. Then for the given extracted line segment the vertical projection graph is drawn. The system will calculate the total number of black pixels in each column in the vertical projection graph and if it exceeds the user defined vertical threshold value it is considered as a character boundary location. The processed line segment is presented to the user by applying the system identified character boundary locations as straight lines, and in the particular location a check box is applied allowing the user to check on or off based on the user preference. The selection facility is given to the user since sometimes the system could identify the patches on the rock as a character and also the system might have identified several boundaries for a single character because of the noise. Basically this provides the user some assistance to segment the characters. Finally the individual characters are extracted from the user specified boundary location. The user has given the facility to amend the horizontal and vertical thresholds using a slider. These extracted characters are resized into 30 X 30 pixel images.



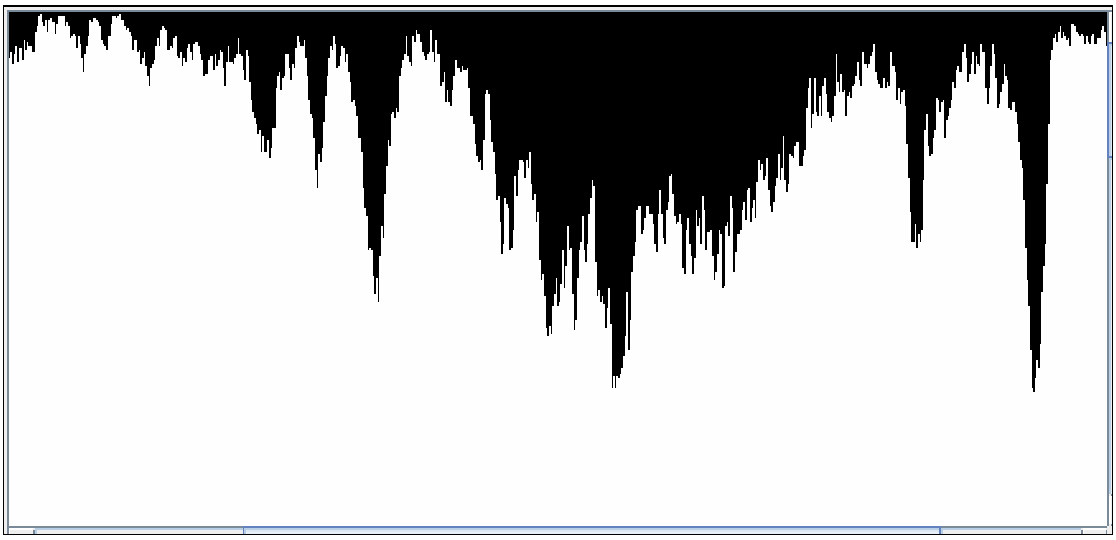


Figure 6.1 : Fragment of Vertical Projection Graph

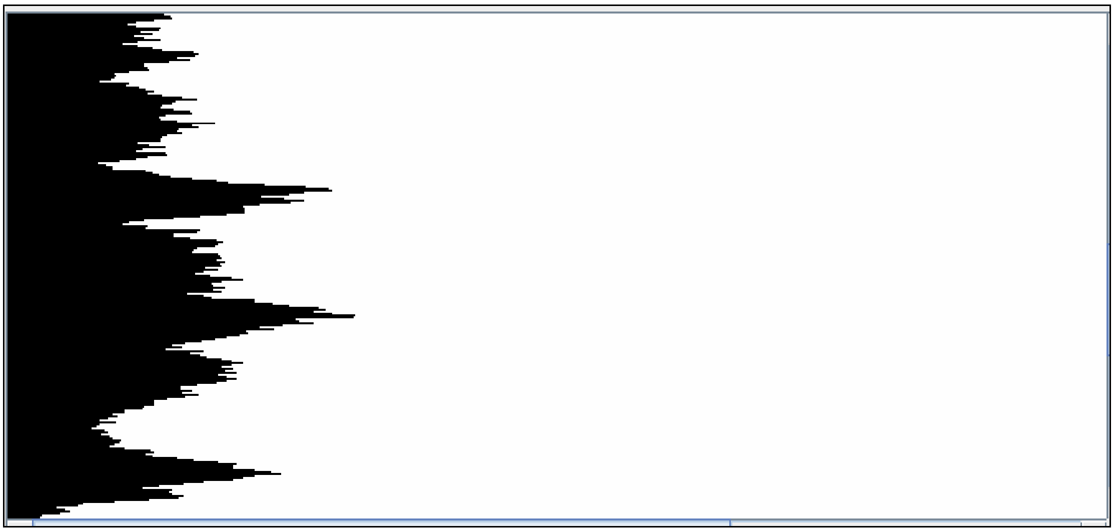


Figure 6.2 : Fragment of Horizontal Projection Graph

The resized image is then sent to the feature extraction module where the features of the 30 X 30 image are extracted. As the first step without any further processing total number of pixels in the image, maximum number of horizontal black pixels in the image, maximum number of vertical black pixels in the image were extracted. Then the 30 X 30 image is segmented into 9 squares as shown in Figure 6.3 and the above features were extracted for each and every square. And finally 30 X 30 pixel image is segmented into 3 layers as displayed in Figure 6.4 and the total number of pixels in each and every layer is calculated.



| | | |
|---|---|---|
| P9 | P2 | P3 |
| P6 | P1 | P4 |
| P7 | P8 | P5 |

Figure 6.3 : 30 X 30 Pixel Image Segmented into 9 Squares

| |
|---|
| **LAYER 01** |
| **LAYER 02** |
| **LAYER 03** |

Figure 6.4 : 30 X 30 Pixel Image Segmented into 3 Layers

As the final step of the feature extraction module the 30 X 30 pixel image is converted into zeros and ones base on the pixel colour. Along with the pixel colour information the features that were extracted in the step 1 of the feature extraction module is send to the recognition module.

Under the feature extraction module a separate user interface is developed to gather the features of the characters for training purposes. In addition to the features of the character this front-end also convert the given image into ones and zeros base on the pixel colour. These values along with the features of the character are written to a .CSV (comma delimited) file which could be used to train the artificial neural network.

The Net Beans – Integrated Development Environment and Java Development Kit 1.6 was used in the feature extraction module. The Graphical User Interfaces were developed using Java Swing.

### 6.4 Recognition Module



The extracted features from the feature extraction module and the digitize values of the given character image are used as the input for the recognition phase. A neural network is build to recognize the characters based on the extracted features and the digitize values of the image. This network takes input-output vector pairs during training phase. The extracted features of the character and the digitize values of the character image are used as the input and the output is a binary representation of a digit which represents each and every character uniquely. The network trains its weight array to minimize error.

#### 6.4.1 Preparation of Training Data

The neural network was trained with the Brahmi characters that were taken from the inscriptions. Random noise were added and erased in these inscriptions in order to construct training data for the neural network. For each character, 100 -200 sample character images were created. The network trains its weight array to minimize error. Figure 6.5 illustrate the training process of the designed artificial neural network.

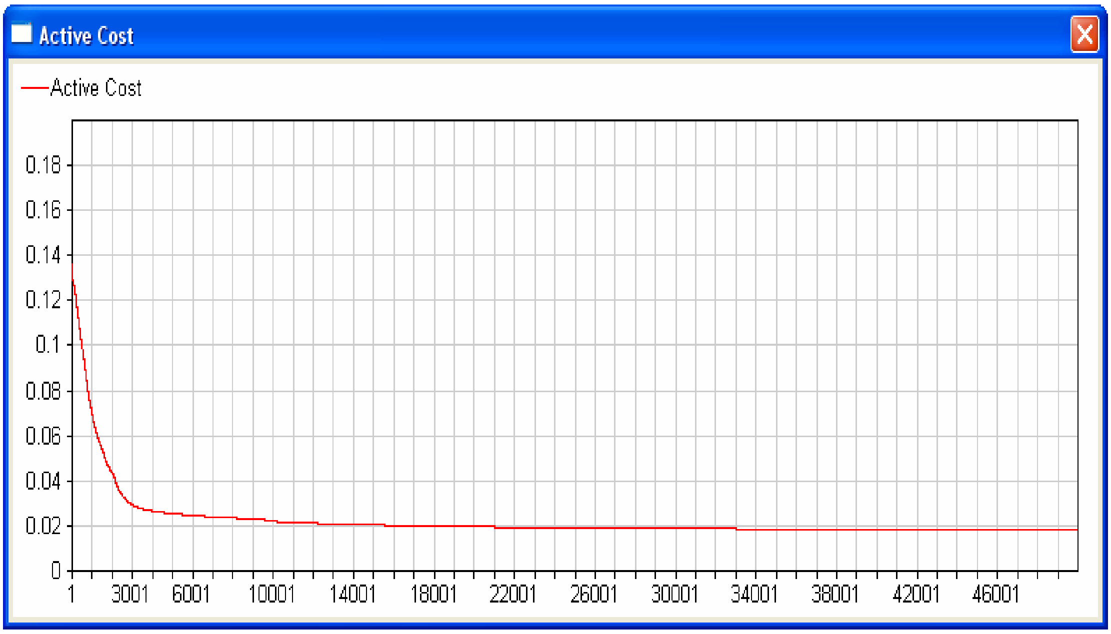



Figure 6.5 : The Training Process of the Neural Network

### 6.4.2 Architecture of the Neural Network

The network designed for this project is a multi layer perceptron model which is a layered feed forward networks typically trained with static backpropagation. The architecture of this network was implemented with 3 layers: input layer, 1 hidden layers and the output layer. The input layer contains 914 neurons to represent the 14 features of the Brahmi character and colour of 900 pixels in the 30 X 30 image. The output layer contains 1 neuron to provide the recognized character. The hidden layer is implemented with the Transfer function – TanhAxon and the learning rule – Momentum. The selection of parameters such as number of hidden layers, type of the transfer function, the learning rule was identified by trial and error sessions.



The Neuro solution software is used to model the neural network, train the network and test for the accuracy. The ANN agent acts as the request agent for the multi agent system that resides in the post processing module. It captures the binary representation output from the neural network convert into the relevant letter and write to the message space in the multi agent system as a character string. The ANN agent is implemented using Java Development Kit 1.6.

### 6.4.3 Intermediate File Format

By the feature extraction module the extracted features of the resized segmented characters and the digitize values of the resized segmented characters are written into a comma separated file which will be used as the input of the neural network during the recognition phase. Table 6.1 presents the format of this file.

| Column Number | Content |
|---|---|
| Column 01 – Column 900 | Digitize values of the 30X30 character image |
| Column 901 | Total number of black pixels in the 30X30 character image |
| Column 902 | Maximum number of vertical black pixels in the 30X30 character image |
| Column 903 | Maximum number of horizontal black pixels in the 30X30 character image |
| Column 904 | Maximum number of vertical black pixels in the top left square |
| Column 905 | Maximum number of horizontal black pixels in the top left square |
| Column 906 | Maximum number of vertical black pixels in the top right square |
| Column 907 | Maximum number of horizontal black pixels in the top right square |
| Column 908 | Maximum number of vertical black pixels in the bottom left square |
| Column 909 | Maximum number of horizontal black pixels in the bottom left square |
| Column 910 | Maximum number of vertical black pixels in the bottom right square |
| Column 911 | Maximum number of horizontal black pixels in the bottom right square |
| Column 912 | Total number of black pixels in layer 01 |
| Column 913 | Total number of black pixels in layer 02 |
| Column 914 | Total number of black pixels in layer 03 |

Table 6.1 : Input File Format of the Artificial Neural Network

The output of the neural network is written to another comma separated file which will print the binary representation of the recognized character. This file is used by the ANN agent to extract the relevant character from the database that belongs to the specified binary representation in the file.

### 6.5 Post Processing Module

The main task of the post processing module is to rectify the output given by the recognition module and produce sentences with minimal error. Post processing module is a multi agent system that operates on the input given by the ANN agent.

#### 6.5.1 Ontology

The ontology consists of a small database and several text files. The database consists of dictionary where the Sinhala words are stored in English letters. Table 6.2 presents some of the sample data in the dictionary table. The variations of the letters such as Kombuwa , Aela Pilla , Al Lakuna are not handled by this dictionary.



| Word in the Dictionary | Relevant Sinhala Word |
|---|---|
| GNS | ගණස |
| SGS | සගස |
| AGH | අගහ |
| BT | බට |
| MARM | මනරම |
| LN | ලෙණ |

Table 6.2 : Sample Data in the Dictionary Table

The database also consists of a table that store the Sinhala letter and the relevant Unicode value. For efficient access two text files are used to store the other relevant information. One text file store the Sinhala words with the variations - Kombuwa , Aela Pilla , Al Lakuna etc. And the other text file is a comma separated file which stores the statistical information of the words.

Figure 6,6 displays a sample record that is in the statistical information file. As a pre requirement these values are already saved by the user by analyzing the sentences in the inscriptions.

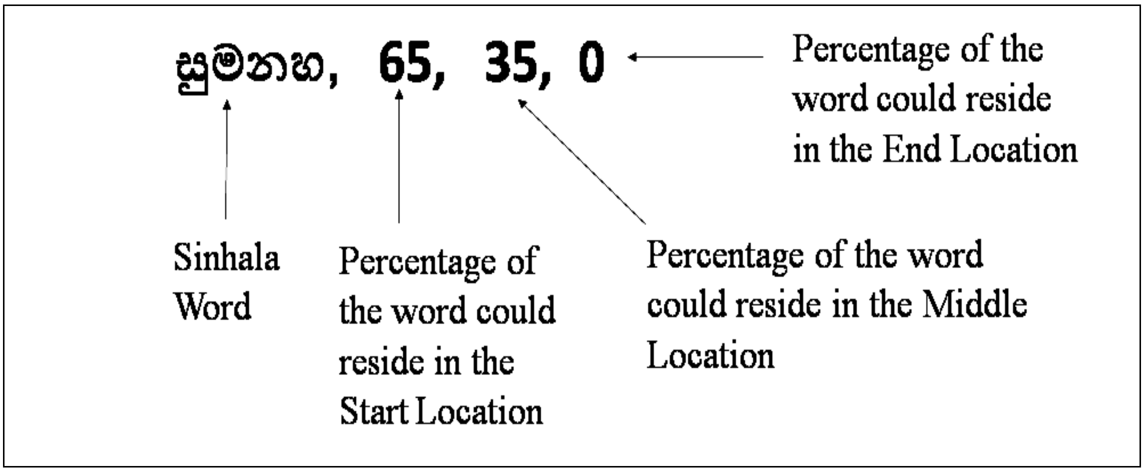



Figure 6.6 : A Sample Record in the Word Statistical Information File

### 6.5.2 Common Message Space

The output of the agents in the multi agent system is displayed in the message space. This is developed using Java Development Kit 1.6 and the Graphical User Interfaces were developed using Java Swing.



### 6.5.3 Lexical Agent

This agent is implemented using the java technology. The input for this agent is a character string that might or might not contain the incorrect characters. The lexical agent analyzes the input sentence, compare with the dictionary and identify the matching words that could replace the incorrectly recognized characters in the input string. After this process a sentence that contain words with the suggested words are created. Subsequently all the words in the sentence are represented in a graph and then using the breath first search algorithm sentences are formulated with all the possible combinations.

Table 6.3 illustrates the process flow of the lexical agent.

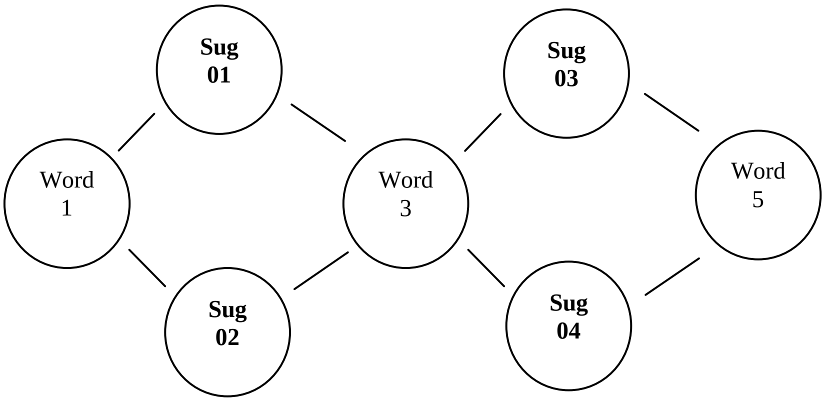
| | |
|---|---|
| **Initial String** | Word1 word2 word3 word4 word5<br>Assumption: word 2 and word 4 contains characters that are incorrectly identified |
| **Step 01 :** Suggestions Provided | Word1 **[Suggestion 01] [Suggestion 02]** word3 **[Suggestion 03][Suggestion 04]** word5 |
| **Step 02 :** Directed graph is constructed |   |
| **Step 03 :** Sentences are generated | Word1 **[Suggestion 01]** word3 **[Suggestion 03]** word5<br>Word1 **[Suggestion 01]** word3 **[Suggestion 04]** word5<br>Word1 **[Suggestion 02]** word3 **[Suggestion 03]** word5<br>Word1 **[Suggestion 02]** word3 **[Suggestion 04]** word5 |



Table 6.3 : Process Flow of the Lexical Agent

After this process a set of sentences are created with the suggested words. Then these sentences go through a spelling correction process where the variations of the letters are handled. Then the words in the formed sentence are converted into standard Sinhala Unicode characters. By acquiring word by word in a sentence the system create a regular expression with the variations that could exist for a Sinhala word. This regular expression is compared with the Sinhala dictionary that resides in a text file. Base on the comparison the relevant word is modified if needed. Figure 6.7 illustrates the spelling correction process of the word ‘LN’.

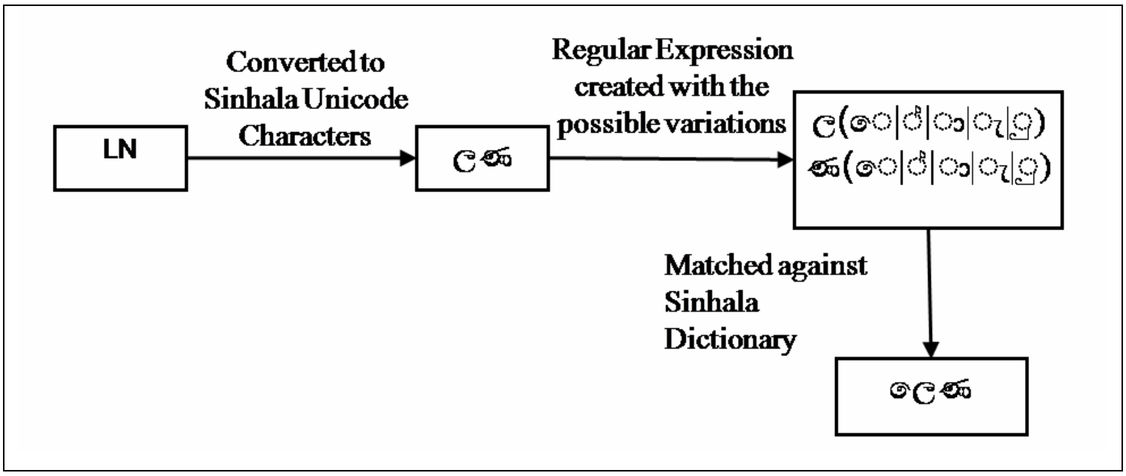



Figure 6.7 : Spelling Correction Process

After the spelling correction process the sentences are presented to the message space for other agents' response.

### 6.5.4 Structure Analyst Agent

Structure Analyst agent is responsible for the structure of the sentence. This agent assigns marks for the sentences based on a particular schema. This whole schema is based on the starting, middle and ending portions of the sentences. The marking scheme process works as follows:



Firstly the total number of words in the acquired sentences is calculated. Then it is divided by 3 and the result value is the number of words that is considered in the starting and the ending portion of the sentence and the rest of the words belong to the middle portion. Then the structure analyst agent uses the word statistics information that resides in the ontology to find out the marks for each and every word when it resides in different sections. Based on the stored marks in the word statistics information file, the marks for the words in the acquired sentence are calculated. The summation of the marks achieved by the words is final marks achieved by the whole sentence. If the final mark of the sentence is greater than threshold mark, then it is considered as a sentence that is structural wise correct. If the sentence is structural wise correct the structure analyst agent reply as 'I ACCEPT' else it will reply as 'I REJECT'. The whole algorithm is implemented using Java. Table 6.4 illustrates the process flow of the structure analyst agent.

**System Setup**

**Values in the Statistical Database**

| Word | Start Percentage | Middle Percentage | End Percentage |
|---|---|---|---|
| Word 1 | 20 | 80 | 4 |
| Word 2 | 40 | 78 | 20 |
| Word 3 | 14 | 88 | 12 |
| Word 4 | 45 | 10 | 50 |
| Word 5 | 32 | 30 | 16 |
| Word 6 | 90 | 32 | 8 |
| Word 7 | 80 | 40 | 10 |

**Acquired Sentences**

Sentence 01 - Word 1 Word 2 Word 3 Word 4 Word 5 Word 6 Word 7
Sentence 02 -Word 6 Word 2 Word 3 Word 1 Word 5 Word 4 Word 7

**Step 01:**
The words are segmented into sections.

Total number of words in the sentence = 7
Number of words that could include in one section (Starting and Ending) = 7 /3 = 2

| Starting Section | | Middle Section | | | Ending Section | |
|---|---|---|---|---|---|---|
| Word 1 | Word 2 | Word 3 | Word 4 | Word 5 | Word 6 | Word 7 |
| Word 6 | Word 2 | Word 3 | Word 1 | Word 5 | Word 4 | Word 7 |



**Step 02:**
Assign marks to the words base on the Statistical Database

**Sentence 01**

| Starting Section | | Middle Section | | | Ending Section | |
|---|---|---|---|---|---|---|
| Word 1 | Word 2 | Word 3 | Word 4 | Word 5 | Word 6 | Word 7 |
| 20 | 40 | 88 | 10 | 30 | 8 | 10 |

Total marks achieved by Sentence 01 - 206

**Sentence 02**

| Starting Section | | Middle Section | | | Ending Section | |
|---|---|---|---|---|---|---|
| Word 6 | Word 2 | Word 3 | Word 1 | Word 5 | Word 4 | Word 7 |
| 90 | 40 | 88 | 80 | 30 | 50 | 10 |

Total marks achieved by Sentence 02 - 388

**Step 03 :**
Select the valid sentences

Threshold Mark – 300

The mark for sentence 01 is lesser than the threshold mark therefore the sentence 01 is considered as an invalid sentence.

The mark for sentence 02 is greater than the threshold mark therefore the sentence 02 is considered as a valid sentence.

Table 6.4: Process Flow of the Structure Analyst

### 6.5.5 Semantic Agent

Semantic agent is a human expert agent who is responsible to accept or reject the sentence that was provided by the lexical agent. This agent selects reply to the sentence base on the knowledge of the context.

Base on the negotiation that occur between the agents the sentence provided by the lexical agent could accept and consider as the recognized inscription character string or reject and consider the next option given by the lexical agent.

## 6.6 Brahmi to Sinhala Converter Module

A Microsoft Access database is maintained in this section. The database contains the mapping between Brahmi characters with the relevant Sinhala Unicode characters. Using the knowledge stored in the Microsoft Access database the recognized Brahmi character will be converted to the relevant Sinhala Unicode character. This is performed using the Net Beans – Integrated Development Environment and Java Development Kit 1.6.



## 6.7 Summary

This chapter provides implementation details of each module that is stated in the design diagram in Chapter 05. Extra details of implementation such as screen shots are presented in the appendix section. The next chapter reports how the system is evaluated using different test approaches.