TOPIC	:	AUTOMATED ESSAY GRADING SYSTEM
NAME	:	CHONG THENG HUI
MATRIC NO.	:	WEK020037
SUPERVISOR	:	PN. NORISMA IDRIS
MODERATOR 1	:	A. P. DR. SYED MALEK FAKAR DUANI
MODERATOR 2	:	DR. RUKAINI
DURATION	:	28/06/2004 – 27/03/2005

Abstract

Automated Essay Grading System is a web-based assessor for History subject. It is designed and developed for the local History teachers and their students. One of the special feature is this system is Malay language oriented. Teacher who is a grader can upload his question together with a model answer into the database. Students then can use the system to do some exercises or sitting for test. Moreover, the outcome of the result will be known by the student as soon as he or she finished the test.

Nevertheless, the development of this automated essay grading system is aim to enhance the information retrieval research on the Malay language stemmer through the indexing module of the system. Hence, many researches had been carried out to study existing worldwide automated essay grading system. Generally, this system is built using the next generation of programming methodology, i.e. ASP.NET with Visual Basic.NET as its backbone.

This automated essay grading system is developed using modified classical analysis methodology, i.e. waterfall model with prototyping. It is not only enhancing the old waterfall model, but also giving us a good system development life-cycle model. Besides, the automated essay grading system has a database which constructed on a client-server architecture basis.

In conclusion, this automated essay grading system should give a good impact in the local AI research field in Malaysia, instead of playing the role of a machine that replace the human grader only.

Acknowledgement

In the process of completing this proposal, I have gained a lot of guardian and advices from the lecturers, coursemates and friends. Hereby, I would like to express my greatest appreciation to the following persons.

First of all, I would like to thank to my dedicated supervisor, Pn. Norisma Idris, for her guidance, invaluable advice and instruction throughout the project. Under her continuous support and supervision, the project has been carried out smoothly and successfully.

Secondly, I would like to express my gratitude to Assoc. Prof. Dr. Syed Malek Fakar Duani as my first moderator for spending his precious time and giving advice during viva. Also, thanks to my second moderator, Dr. Rukaini for her patience, constructive criticism and suggestions given.

Thirdly, I want to show my sincere appreciation to Mr. Cheong Tau Pau, as my project partner, for giving countless ideas and guides regarding to the project. I have gained a lot of useful knowledge related to project from his advices and suggestions. I feel happy to working with him.

Besides, I would like to thank to my room-mate, K.T. Lee and ex-roommate, Terry Tee, for giving me the highest moral supporting spirit; Henry Law for his hardware supporting while my laptop was down; F.K. Hang, P.H. Lee and Geam for their great entertainment in Bridge and *Cho Tai Ti* game.

Last but not least, I wish to thank my parents, friends and coursemates who have been considerate and supportive through the whole period of project developments.

Without you all, it wouldn't have been possible.

Table of Contents

Content	*Page*
Abstract	ii
Acknowledgement	iii
List of Figures	ix
List of Tables	x
Chapter 1: Introduction	1
1.1 Problems definition	1
1.2 Objectives	2
1.3 Scope and Domain	2
1.4 Limitations Preview	2
1.5 Timeline	3
1.6 Chapter Summary	3
Chapter 2: Literature Review	4
2.1 Morphology of Malay Language	4
2.2 Review of the Existing Systems	5
2.2.1 Project Essay Grade (PEG)	5
2.2.2 Intelligent Essay Assessor (a LSA model)	6
2.2.3 Electronic Essay Rater (E-Rater)	8
2.2.4 Comparisons	9
2.3 Reviews of the Existing Stemming Algorithms	10
2.3.1 Porter Stemming	10
2.3.2 Othman's Stemming	11
2.3.3 Tengku Sembok's Stemming	12
2.3.4 Stemmer for Indonesian Language	12
2.4 Chapter Summary	13
Chapter 3: Methodology	14
3.1 Waterfall Life-Cycle Model	15
3.2 Prototyping Model	16

Content *Page*

3.3 Waterfall Model with Prototyping 17
3.3.1 Requirements Analysis and Definition 19
3.3.2 System and Software Design 19
3.3.3 Implementation and Unit Testing 19
3.3.4 Integration and System Testing 20
3.3.5 Operation and Maintenance 20
3.4 Justification of Adopted Methodology 21
3.4.1 Simple and Easy to Understand 21
3.4.2 Easy-Gathered User Requirements 21
3.4.3 Systematic 22
3.5 Chapter Summary 23

Chapter 4: System Analysis 24
4.1 Functional Requirement 24
4.1.1 Use-Case of the System 24
4.2 Non-Functional Requirement 28
4.2.1 Accuracy 28
4.2.2 Consistency 28
4.2.3 Reliability 28
4.2.4 Manageability 29
4.2.5 Security 29
4.3 Proposed Tools and Technologies 29
4.3.1 Client-Server Architecture 30
4.3.2 Internet Information Services (IIS) 5.1 31
4.3.3 ASP.NET 31
4.3.4 Visual Basic.NET 32
4.3.5 JavaScript 32
4.3.6 Microsoft® SQL Server 2000 32
4.3.7 Microsoft® Visual Studio.NET 2003 33
4.4 Hardware and Software Requirements 34
4.5 Chapter Summary 35

Content *Page*

Chapter 5: System Design 36

5.1 Definition of Assigned Module 36

5.2 System Architecture 37

5.2.1 General System 37

5.3 System Functionality Design 38

5.3.1 Indexing 38

5.3.1.1 Lexical Analysis 39

5.3.1.1.1 Date and Numbers Removal 40

5.3.1.1.2 Hyphens Removal 40

5.3.1.1.3 Punctuation Removal 40

5.3.1.1.4 Case Conversion 40

5.3.1.1.5 Stopwords Eliminator 41

5.3.1.2 Stemming 41

5.3.1.3 Feature Recognition 44

5.3.1.4 Word Occurrence Counter 44

5.4 Database Design 45

5.4.1 Tables of Database 46

5.5 Prototypes of User Interface Design 48

5.6 Chapter Summary 51

Chapter 6: System Development and Implementation 52

6.1 Development Environment 52

6.1.1 Hardware 52

6.1.2 Software 52

6.2 System Implementation 53

6.2.1 Programming Languages and Approach 53

6.2.2 System Modules and Functionalities 54

6.2.2.1 Lexical Analysis Processes 54

6.2.2.1.1 Punctuation Removal 55

6.2.2.1.2 Date Normalization 55

6.2.2.1.3 Hyphen Removal 56

6.2.2.1.4 Case Conversion 57

Content *Page*

6.2.2.1.5 Stopwords Eliminator 58
6.2.2.2 Stemming Process 59
6.2.2.3 Feature Recognition Process 61
6.2.2.4 Word Occurrence Counter Process 62
6.3 Chapter Summary 64

Chapter 7: System Testing 65
7.1 Unit Testing 66
7.1.1 Ad Hoc Test 67
7.1.2 Black Box Test 67
7.1.3 White Box Test 71
7.2 Integration Testing 73
7.3 System Testing 74
7.3.1 Function Testing 75
7.3.2 Performance Testing 75
7.3.2.1 Stress Testing 75
7.3.2.2 Human Factor Testing 75
7.3.2.3 Response Time Testing 75
7.4 Chapter Summary 77

Chapter 8: System Evaluation and Conclusion 78
8.1 System Evaluation 78
8.1.1 Evaluation of Stemming Algorithm 78
8.1.1.1 Overstemming Errors 79
8.1.1.2 Understemming Errors 81
8.1.1.3 Discussion 81
8.1.2 Evaluation of Stopwords List 82
8.1.3 Encountered Problems and Solutions 83
8.1.3.1 Difficulties in Designing User Interface 83
8.1.3.2 Timer Architecture 83
8.1.4 System Strengths 83
8.1.4.1 In Specific Term 84

Content *Page*

8.1.4.2 In General Term 84

8.1.5 System Limitations 85

8.1.5.1 In Specific Term 85

8.1.5.2 In General Term 85

8.1.6 Future Enhancement 86

8.1.6.1 In Specific Term 86

8.1.6.2 In General Term 86

8.2 System Conclusion 87

8.3 Chapter Summary 87

Appendix A 88

Appendix B 90

Appendix C 93

Appendix D 100

Appendix E 101

References 120

List of Tables

Name of Table	*Page*
Table 2.1 Comparison Among the Existing Systems	9
Table 4.1: Description of Use-Case of Automated Essay Grading System	26
Table 4.2: Proposed Tools and Technologies	30
Table 4.3: Hardware and Software Requirement	34
Table 5.1: Table of Teacher	46
Table 5.2: Table of Student	46
Table 5.3: Table of EssayQuestions	47
Table 5.4: Table of EssayAnswers	47
Table 7.1: Response Time VS Words	76
Table 8.1: Overstemming Word Pairs with Stemmed Words	79
Table 8.2: Indication of Overstemming Word Pairs	80
Table 8.3: Indication of Understemming Word Pairs	81
Table A.1: ASP.NET Object Definition Standard	88
Table C.1: Word Frequency Analysis	93
Table D.1: Suggested Stopword List	100

List of Figures

Name of Figure	*Page*
Figure 1.1: Gantt Chart of the Project Milestones	3
Figure 2.1: Demonstration Page of IEA	7
Figure 2.2: Demonstration Page of E-Rater	9
Figure 2.3: Structure of Porter Stemmer	10
Figure 2.4: Basic Design of Porter Stemmer for Indonesian Language	13
Figure 3.1: Basic System Development Process Model	14
Figure 3.2: The Full Waterfall Life-Cycle Model	15
Figure 3.3: Prototyping Model	16
Figure 3.4: Waterfall Model with Prototyping	18
Figure 4.1: Use-case Diagram of Automated Essay Grading System	25
Figure 5.1: The Framework of Automated Essay Grading System	37
Figure 5.2: Four Major Processes in Indexing Module	38
Figure 5.3: Structure of Lexical Analysis	39
Figure 5.4: The Algorithm of Malay Stemming	43
Figure 5.5: The Database Design for the Automated Essay Grading System	45
Figure 5.6: The Set Question Screen	48
Figure 5.7: Teachers Upload	49
Figure 5.8: Students Essays Composer	49
Figure 5.9: Questions Selection Screen	50
Figure 6.1: Design of Engines and Function in Indexing Module	54
Figure 7.1: System Testing Flow Example	65
Figure 7.2: Testing Platform of AEGS	66
Figure 7.3: Lexical Analysis Unit Testing	68
Figure 7.4: Stemming Unit Testing	69
Figure 7.5: Feature Recognition Unit Testing	70
Figure 7.6: Word Occurrence Counter Unit Testing	71
Figure 7.7: Data Flow of Stopword Eliminator Engine	72
Figure 7.8: Top-Down Testing Process of Indexing Module	74
Figure 7.9: Graph of Response Time VS Amount of Words	76

Chapter 1: Introduction

Here begins the introduction to my system, i.e. Automated Essay Grading System (AEGS) through its problem definition, objectives, scope and domain, as well as the pre-analyzed limitation.

1.1 Problems definition

Since information retrieval of language plays an important role in cyberspace, especially, many ethnic from any part of the world want to preprocess their native language according to their own morphological rules. A suitable information retrieval system should be development to fulfill all needs.

As we know, essay is the most useful tool to express and organize one's view or points. But sometimes, handwritten essay is not that legible to read by every marker. So, it is better to use a web-based examination. Candidates are required to type their answer using any word processor.

Meanwhile, there are some grading difficulties if using traditional marking. Absolutely, that is a time-consuming activity, i.e. to mark answer paper, especially for public examination that involves thousands of students.

As you know, human-beings are always influenced by the physical and mental factor. Thus, the outcome of the result will be inconsistent due to the emotional effect of human grader.

Nowadays, the traditional marking system could not give the result or outcome of the examination to a student instantly. It takes weeks or months to show the result and analysis to students.

1.2 Objectives

In general, for the whole system:

- To create an automated essay grading system for History subject.
- To reduce time of grading process that done by human marker.

Meanwhile, in specific definition which is the indexing part:

- To enhance the indexing part in an automated essay grading system
- To create a less-error stemmer for Malay Stemming.

1.3 Scope and Domain

This automated essay grading system is mainly designed for two main users:

- Teachers/graders
- Students.

This system is mainly focus on factual subject, i.e. History. Besides, this system is restricted to Malay language only.

1.4 Limitations Preview

After pre-analysis, we found some limitation on this automated essay grading system:

- This automated essay grading system is aimed for certain factual subject, i.e. History, but not for literature subject.
- Secondly, this system cannot grade according to the writing style, but according to the contents and facts only.
- The stemming process might not reach 100% efficiency. Some stemming failure can cause error as well. The problem of understemming and overstemming could only be minimized.

- Also, the system is based on a model answer. So, if student provides different answer that couldn't be found on the answer scheme, so it might affect the grading process too.
- Moreover, this system could not recognize handwritten words. It uses computer-typed words only.

1.5 Timeline

An estimated project timeline is needed to describe the software development cycle. It enumerates the phase of a project and breaking each of the phases into discrete tasks that need to be carried out. It is essential as it acted as a time management and control to the developer to determine what tasks to be carried out and what goals should be achieved when a certain milestone is met.

ID	Task Name	Start	Finish	Duration
1	Preliminary Study and Planning	28/06/2004	05/07/2004	8d
2	Literature Study and Research	05/07/2004	03/08/2004	30d
3	System Analysis	04/08/2004	02/09/2004	30d
4	System Design	02/09/2004	01/10/2004	30d
5	Implementation and Coding	04/10/2004	31/01/2005	120d
6	System Testing	01/02/2005	14/02/2005	14d
7	System Evaluation	08/02/2005	14/02/2005	7d
8	Documentation	28/06/2004	14/02/2005	232d

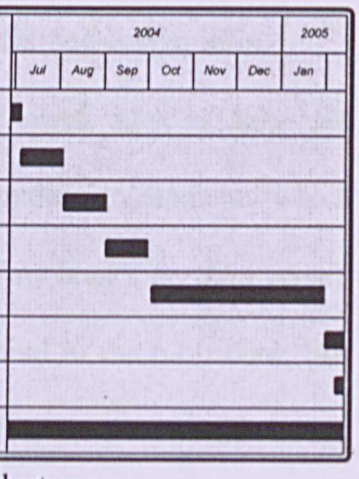


Figure 1.1: Gantt Chart of the Project Milestones

1.6 Chapter Summary

This chapter gives an overview to the project, i.e. Automated Essay Grading System. Besides stating the problem definition and scope, there are some main objectives in way of the general and specific term. A estimated timeline for this project also been attached.

Chapter 2: Literature Review

A literature review is an evaluative report of information found in the literature related to selected area of study. Thus, in this chapter, you could know what I have figured out during my literature reading and research. Besides knowing the morphological structure of Malay language, I came across some existing and available automated essay grading system as well. Anyway most of them are designed and developed for English. Nevertheless, I found some algorithm for Malay words stemming which had been done by few local Professors.

2.1 Morphology of Malay Language

Every language has its own structure and grammar, Malay language has no exception too, with its own morphology or word formation process. Basically, Malay language has four main types of affixes. These are prefix, suffix, prefix-and-suffix, infix.

Prefix refers to element added in front of a root word, such as ***ber****gaduh* (argue), and ***mem****erintah* (rule). Meanwhile suffix is regarding to elements which added at the behind of a root word. For example, *buku****ku*** (my book), *beri****kan*** (give). Prefix-and-suffix are elements that added in front and behind of the root word, like ***mem****ain****kan*** (play), ***ber****lain****an*** (difference). Lastly, infix refers to elements that added in the middle of a root word, such as *t****el****unjuk* (index finger). However, the infixed word has totally changed the meaning of itself. So, our proposed system will not stem them.

Apart from that, there are some special situations that after removing the prefix, we need to replace it with another alphabet if the first letter is a vowel. For example, ***men****unggu* (wait) became *tunggu* (wait), in which the prefix, "men" will be replaced with "t" when the first letter is a vowel, "u".

2.2 Reviews of the Existing Systems

2.2.1 Project Essay Grade (PEG)

Project Essay Grade (PEG) is among the earliest automated essay grading systems. It was introduced by Ellis Page in 1966. It focused on the writing quality, but not to the context. It based upon the concept of "proxes", approximations of "trins", intrinsic variables of interest within the essay, whereby human grader would look for but a computer are unable to do that. The example of proxes includes average word length to represent trins of fluency; number of semicolons and relative pronouns to indicate complexity of sentence structure; word rarity to show diction of essay.

A huge number of essay samples (100 to 400) are selected and marked by human graders, in order to determine values for up to 30 proxes (*http://edres.org/betsy/three_prominent.htm*, 15/09/2004). A multiple regression equation is then developed from these measures. This equation is then used, along with the appropriate measures from each student essay to be graded, to predict the grade which a human grader would assign.

As you aware, this PEG system is rely on statistical approach with an assumption that quality of an essay is reflected by the measurable proxes. There is no Natural Language Processing (NLP) technique and no lexical context in use. Thus, PEG requires training. It needs to assess a large number of previously marked by human essays for proxes, in order to evaluate the regression coefficients, which enables for marking new essays.

To conclude, Page has over 30 years of research consistently showing exceptionally high achievement in his latest experiments that reaching a multiple regression correlation as high as 0.87 with human graders (Salvatore *et al*, 2003).

2.2.2 Intelligent Essay Assessor (a LSA model)

Intelligent Essay Assessor (IEA) was developed by Laudauer, Foltz & Laham in 1998. IEA is based on the Latent Semantic Analysis (LSA) technique that was originally designed for indexing documents and text retrieval (Deerwester, Dumais, Landauer, Furnas & Harshman, 1990). LSA represents documents and their word contents in a large two-dimensional matrix semantic space (Williams, 2001). Using a matrix algebra technique known as Singular Value Decomposition (SVD), new relationships between words and documents are uncovered, and existing relationship are modified to more accurately represent their true significance.

A matrix represents the words and their contexts. Each row of the matrix represents words, while each column represents the sentences, paragraphs, and other subdivisions of context in which the word occurs. The cells of the matrix contain the frequencies of the words in each context.

The SVD is then applied to the matrix. SVD breaks the original matrix into three component matrices, which, when matrices multiplied, reproduce the original matrix. Using a reduced dimension of these three matrices in which the word-context associations can be represented, new relationship between words and contexts are induced when reconstructing a close approximation to the original matrix from the reduced dimension component SVD matrices. These new relationships are made manifest, whereas prior to the SVD, they were hidden or latent.

To grade an essay, a matrix for the essay document is built, and then transformed by the SVD technique to approximately reproduce the matrix using the reduced dimensional matrices built for the essay topic domain semantic space (Palmer *et al*, 2002). The semantic space typically consists of human graded essays. Vectors are then computed from a student's essays data. The vectors for the essay

document, and all the documents in the semantic space are compared, and the mark of the graded essays with the lowest cosine value in relation to the essay to be graded is assigned.

As you notice, LSA doesn't lay store by words order because words order is not important for sense of a passage. Besides, it needs a huge number of data to construct the suitable matrix for all the word occurrences.

In conclusion, IEA is good from aspect low unit cost, quick customized feedback and plagiarism detection. It is suitable for analyzing expository essays on topics such as science, social studies, medicine or business but not for factual knowledge.

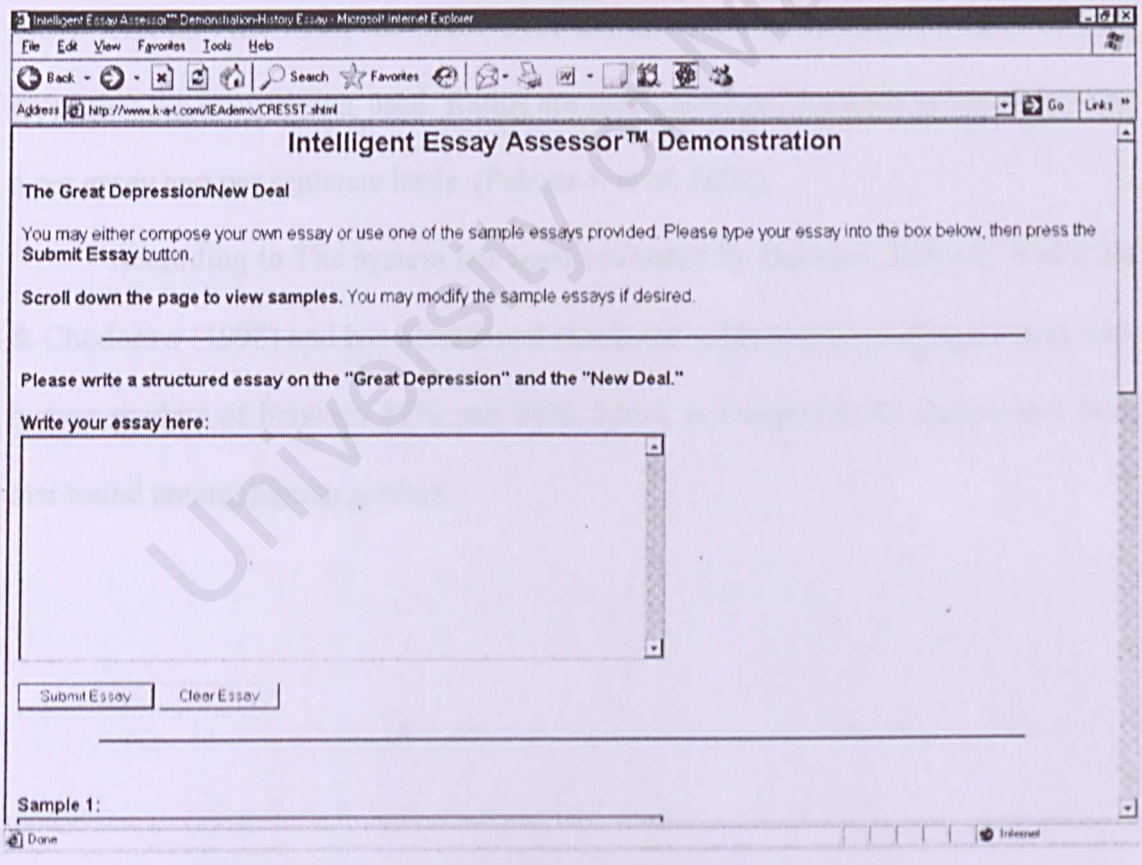


Figure 2.1: Demonstration Page of IEA

2.2.3 Electronic Essay Rater (E-Rater)

The Educational Testing Service (ETS)'s Electronic Essay Rater (E-Rater) was developed by Burstein and others in late nineties. This system is a sophisticated "Hybrid Feature Technology" that uses syntactic variety, discourse structure (like PEG) and content analysis (like LSA).

This system uses a combination of statistical and NLP techniques to extract linguistic features of the essays to be graded. Like other available systems, E-Rater also evaluates student essays from a benchmark set of human graded essays. Multiple linear regression techniques are then used to predict a score for the essay.

One of the scoring guide criteria is essay syntactic variety. After parsing the essay with an NLP tool, the parse trees are analysed to determine clause or verb types that the essay writer used. Ratios are then calculated for each syntactic type on a per essay and per sentence basis. (Palmer J. *et al*, 2002).

According to The system has been evaluated by Burstein, Kukich, Wolff, Lu & Chodorow (1998) and has been found that it can achieve a level of agreement with human graders of between 87% and 94%, which is claimed to be comparable with that found among human graders.

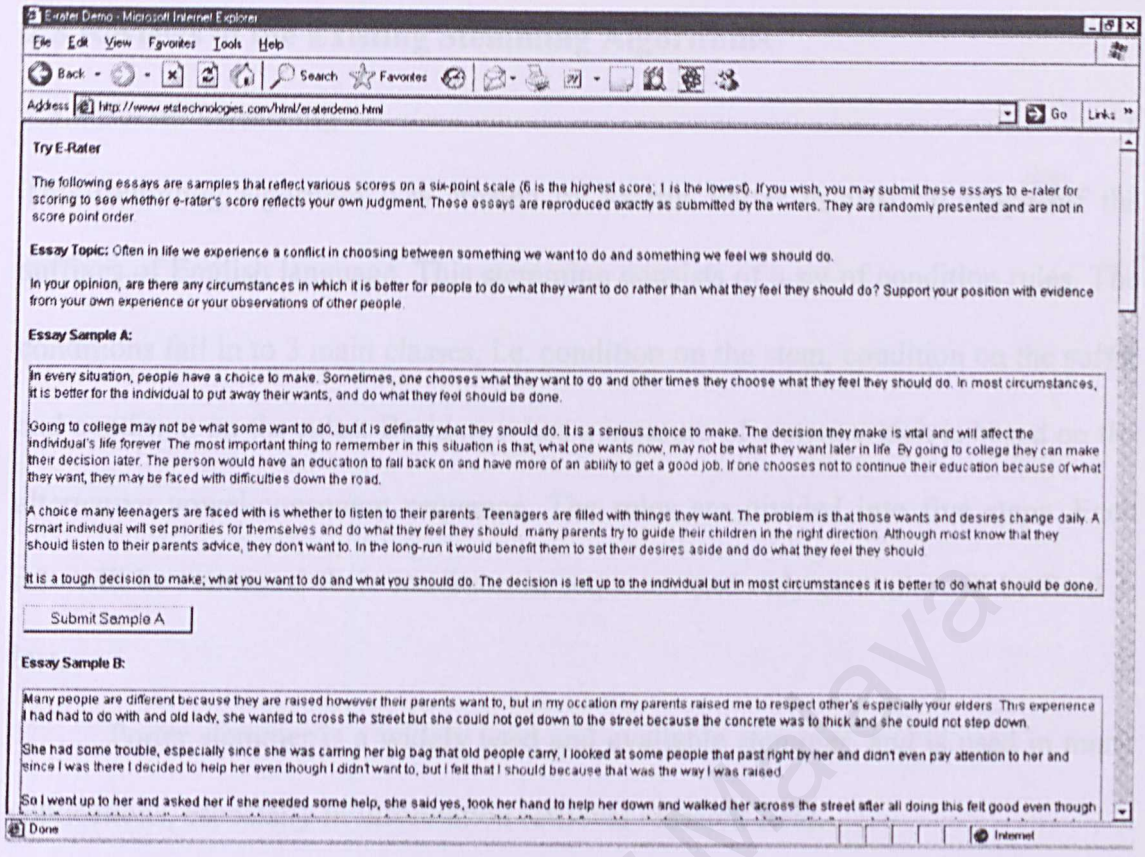


Figure 2.2: Demonstration Page of E-Rater

2.2.4 Comparisons

Comparing to the three existing automated essay grading systems above, you could make a conclusion as below:

Table 2.1 Comparison Among the Existing Systems

	PEG	***IEA***	***E-Rater***
1. Writing style	Able	Unable	Able
2. Content	Unable	Able	Able

2.3 Reviews of the Existing Stemming Algorithms

2.3.1 Porter Stemming

This stemming algorithm was created by Martin Porter in 1980. It strips off the suffixes of English language. This stemming consists of a set of condition rules. The conditions fall in to 3 main classes, i.e. condition on the stem, condition on the suffix and condition on the rules. Besides, it has a measure of a stem which is based on the alternative vowel-consonant sequence. The rules are divided into five steps. Each rule will be examined their condition in sequence and only one rule will be fired at last.

Porter stemmer is a widely used and available stemmer, and is used in many applications, especially in information retrieval research field.

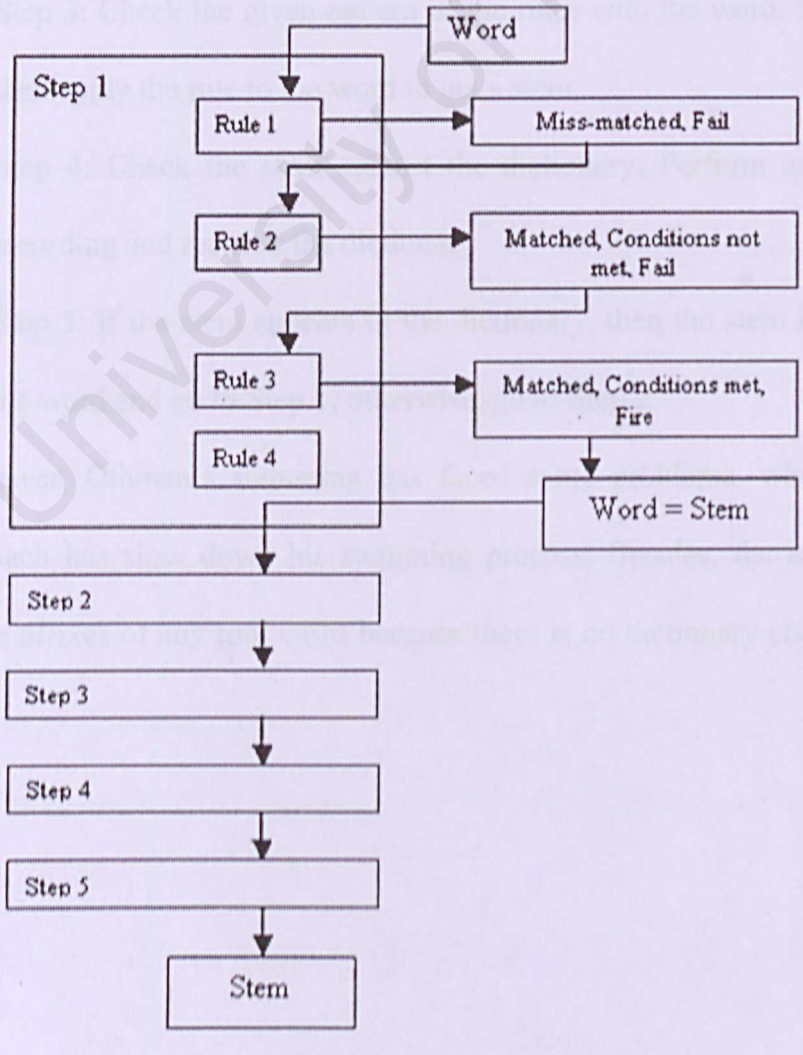


Figure 2.3: Structure of Porter Stemmer

2.3.2 Othman's Stemming

This stemming algorithm was created by Asim Othman in 1993. This is a stemming algorithm for Malay language. He has used 121 morphological rules in his algorithm for stripping affixes from Malay words. Moreover, he stated that the dictionary plays an important role in Malay language stemming. So, this is the unique part of Malay words stemming because Malay language not only has suffixes, it has prefixes and combination of prefixes and suffixes too.

According to Othman, there are several steps for the stemmer:

- Step 1: If there are no word, then stops; otherwise set the next word
- Step 2: If there are no more rules, then accept the word as root word and go to Step 1.
- Step 3: Check the given pattern of the rules with the word. If it matches, then apply the rule to the word to get a stem.
- Step 4: Check the stem against the dictionary. Perform any necessary recording and recheck the dictionary
- Step 5: If the stem appears in the dictionary, then the stem is the root of the word and go to Step 1; otherwise, go to Step 2.

However, Othman's stemming has faced some problems, where his rule-based approach has slow down his stemming process. Besides, the stemmer may strips off the affixes of any root word because there is no dictionary checking at the first.

2.3.3 Tengku Sembok's Stemming

Tengku Mohd. Tengku Sembok has modified A. Othman's stemming algorithm by increasing to two new set of rules. There are 432 rules for first set and 561 rules for another set. The most efficient part of Tengku Sembok's stemming part is it first check against dictionary before stem on any Malay words. So, the problem that occurred in Othman's algorithm had been overcome and hence overstemming had been reduced.

Also, it's important to know that Sombok's algorithm relies on the order of the rule in which the affixes need to be checked and removed first. This algorithm has been tested for Malay worded al-Quran and abstract data sets. This algorithm has show a significant improvement in stemming on Malay text.

2.3.4 Stemmer for Indonesian Language

Indonesian language is similar to Malay language. So, I include it as one of my research algorithm as well. This stemming algorithm was developed by Nazief and Adriani (Fadillah Z Tala, 2003). It was a derivation of Porter algorithm for Indonesian language. The Porter stemming is chosen because its idea fits the morphological rules in Indonesian language.

Some addition condition had been done to handle prefix stripping, prefix-suffix stripping and spelling adjustment. Flow chart below shows the design of this Indonesian language stemming.

Moreover, this is a stemmer inherits from Porter's stemming, not a dictionary-based stemmer.

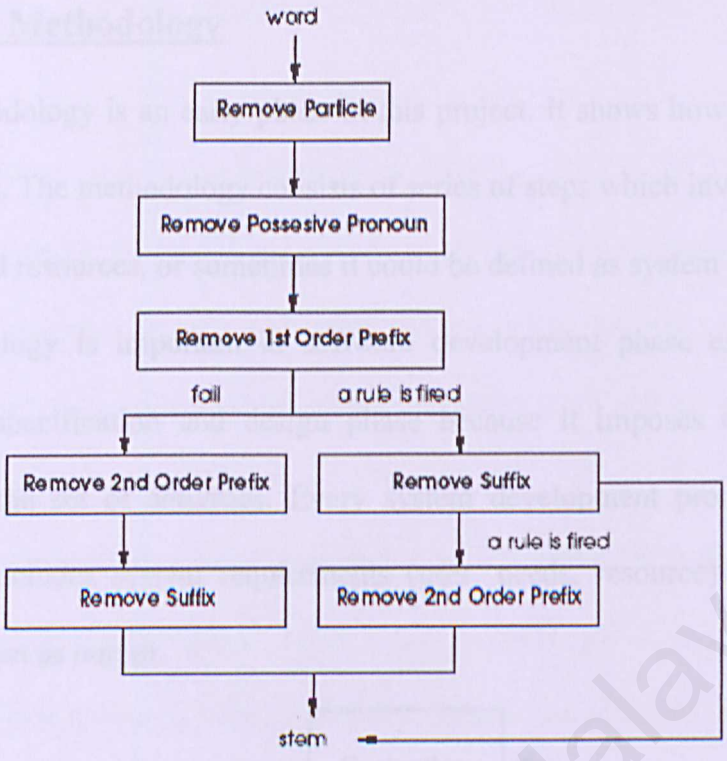


Figure 2.4: Basic Design of Porter Stemmer for Indonesian Language

2.4 Chapter Summary

As you reviewed, many essay grading systems has been created and developed since early in 1900s. Besides, we had reviewed some algorithm for Malay language but there are still some overstemming and understemming errors in all the existing algorithms.

Chapter 3: Methodology

Methodology is an early phase in this project. It shows how the project will be carried out. The methodology consists of series of steps which involving activities, constrains and resources, or sometimes it could be defined as system life cycle model. The methodology is important in software development phase especially in the requirement specification and design phase because it imposes consistency and structure on the set of activities. Every system development process model (see Figure 3.1) includes system requirements (user, needs, resource) as input and a finished product as output.

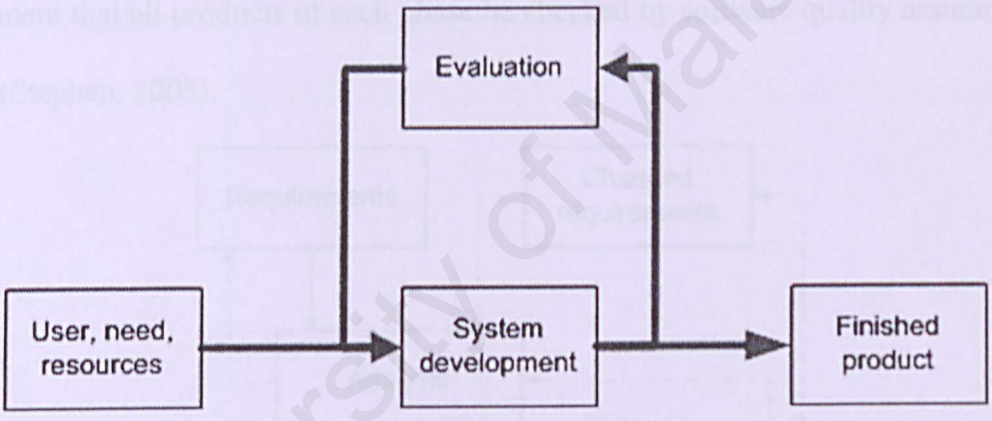


Figure 3.1: Basic System Development Process Model

There are several process models in system development:

- Waterfall Model with Prototyping
- Waterfall Model
- V Model
- Prototyping Model
- Transformational Model
- Spiral Model
- Specification Operation Model
- Phase Development Model

3.1 Waterfall Life-Cycle Model

The Waterfall Life-Cycle-Model was first put forward by Royce in 1970. Figure 3.2 shows the feedback loops for maintenance while the product is being developed. Figure 3.2 also shows the feedback loops for postdelivery maintenance.

A critical point regarding the waterfall model is that no phase is complete until the documentation for that phase has been completed and the products of that phase have been approved.

The waterfall model have many strengths, including the enforced disciplined approach – the stipulation that documentation be provided at each phase and the requirement that all products of each phase be checked by software quality assurance (SQA) (Stephen, 2005).

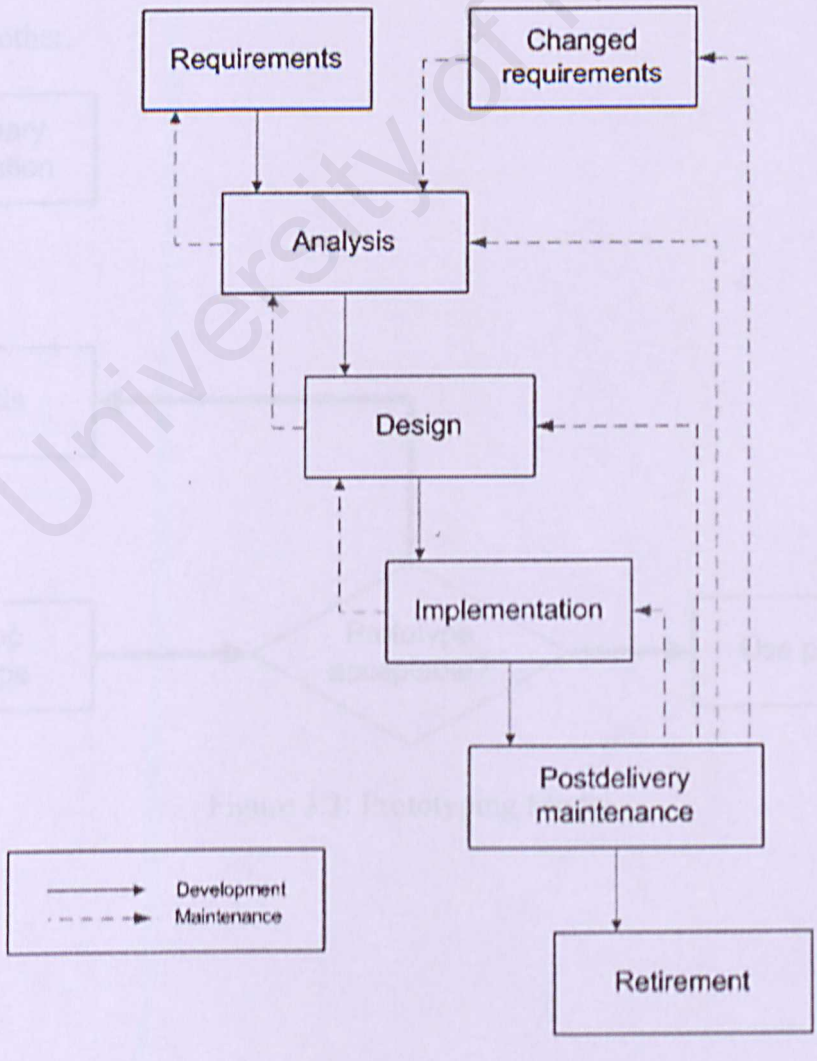


Figure 3.2: The Full Waterfall Life-Cycle Model

3.2 Prototyping Model

Prototyping is a sub-process and prototype is a partially developed product or a simple simulator of the actual system to examine the proposed system and overview on the functionalities. A prototype of automated essay grading system will be built regarding to the project scope and the analysis of the system before start to build the actual system.

Prototyping is very important because:

- To ensure the system achieves the performance goals or constraints.
- To ensure the system fulfills the function requirements
- To ensure the system is practical and flexible.
- To have an insight of how the module and sub-modules interact with each other.

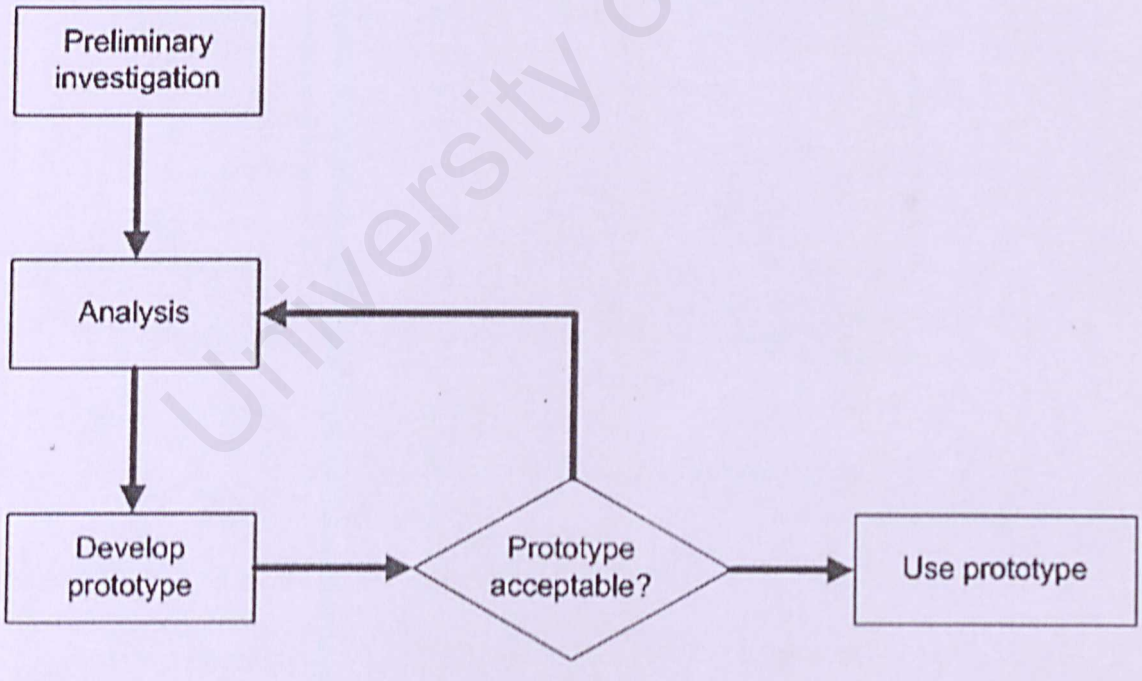


Figure 3.3: Prototyping Model

3.3 Waterfall Model with Prototyping

Prototyping model can be combined with the waterfall methodology to make it more precise. Look at Figure 3.2, the arrows between adjacent stages are bi-directional. Bi-directional means that there is feedback between stages. If a problem is discovered in one stage, developers can return to the previous stages so that suitable corrective action can be taken.

There is a cascading effect where developers can go back further and further up the waterfall until the problem can be corrected properly. Waterfall Model with Prototyping consists of few stages that are depicted as cascading from one to another (see Figure 3.4). Each development stage should be completed before the next begins. The stages are as described below:

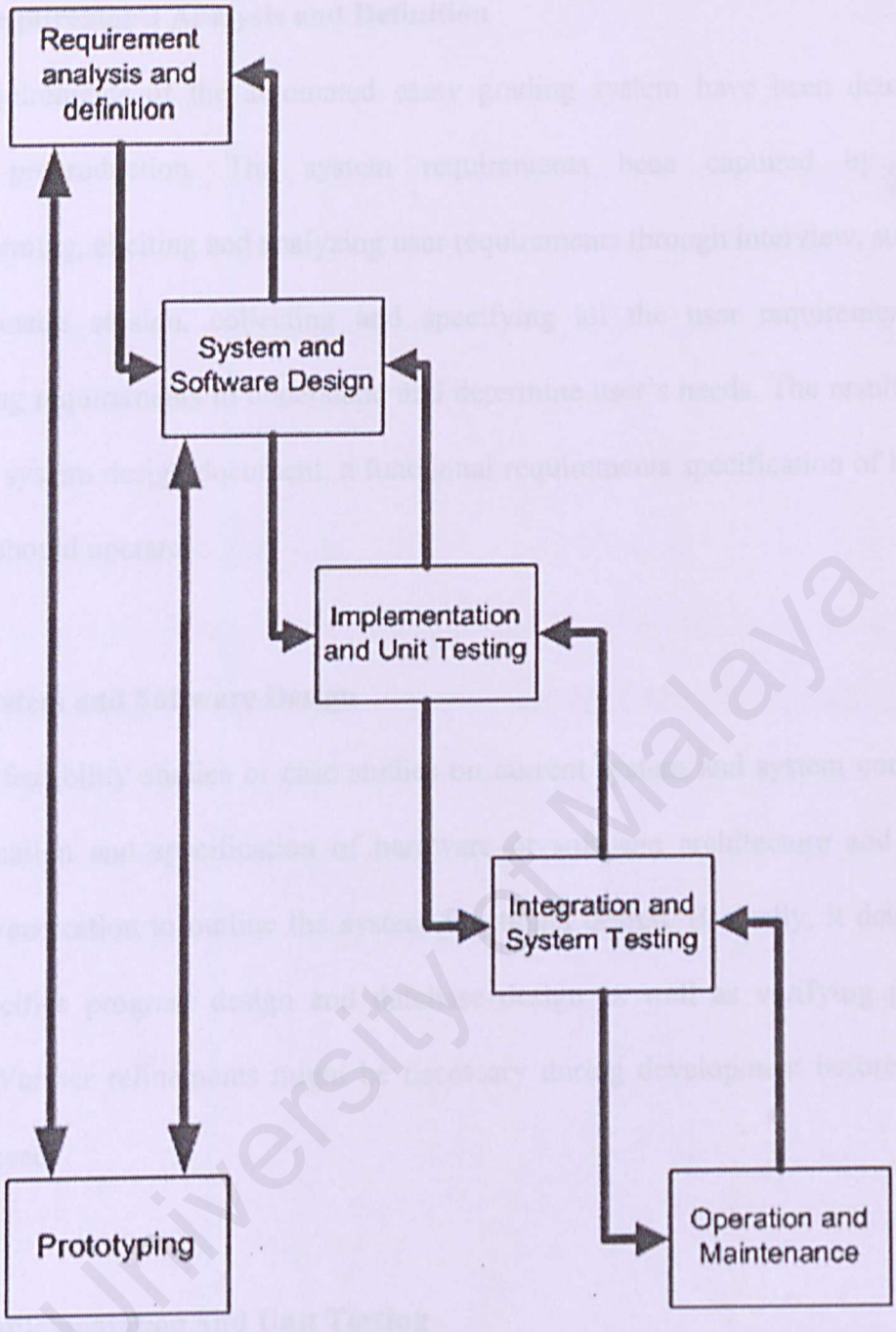


Figure 3.4: Waterfall Model with Prototyping

3.3.1 Requirements Analysis and Definition

The requirements of the automated essay grading system have been determined during preproduction. The system requirements been captured by having brainstorming, eliciting and analyzing user requirements through interview, survey or questionnaire session, collecting and specifying all the user requirements and validating requirements to understand and determine user's needs. The result of this was the system design document, a functional requirements specification of how the system should operate.

3.3.2 System and Software Design

Having feasibility studies or case studies on current system and system constrains, determination and specification of hardware or software architecture and system design verification to outline the system functional design. Basically, it determines and specifies program design and database design as well as verifying program design. Further refinements might be necessary during development before coding can proceed.

3.3.3 Implementation and Unit Testing

This is where the components of the system are actually coded and built. It involves programming standard and procedure that help in translating design to code, programming language, personal planning, tool acquisition, database development, component level documentation and programming management.

3.3.4 Integration and System Testing

Each unit is tested separately and combined it into integrated units. Then, follow by the integrated testing. Bugs discovered during testing are naturally removed. Specifying, reviewing and updating of the system test and validating of system. When testing on the system is completed and proved that it meets the system requirements without error, the system is then delivered.

3.3.5 Operation and Maintenance

This stage involves maintaining control over the system's day- to-day functions and system modifications, perfecting existing acceptable functions and preventing system from degrading to unacceptable level, and revalidating of system. Certain maintenance may be necessary to add features and new content, or to fix bugs that were not removed during the testing phase

3.4 Justification of Adopted Methodology

After analysis and comparison among the process models, a suitable model had been chosen to be adopted in our system development process, i.e. *Waterfall Model with Prototyping*.

Waterfall Model with Prototyping has been chosen due to some following reasons:

3.4.1 Simple and easy to understand

The development methodology is simple and easy to understand. It helps the developers to lay out what they need to do easily. Therefore in the process, they no need to burden themselves with the upcoming stage. In addition, one can has a better understanding and clearer guideline on what he or she should do during the development process. Moreover, it is easier to associate and identify each milestone with its deliverer.

As the methodology is simple and easy to understand, it will be easier to present or explain to the users, especially those who are not familiar with software development. Therefore, developers can give the users or customers a clearer view on what is going on.

3.4.2 Easy-gathered user requirements

With prototyping, users' participations are involved where user requirements can be gathered. It helps the developers to ensure that the requirements are feasible and practical. Furthermore, it helps to access alternative design strategies and decide which is best for the project.

3.4.3 Systematic

This development methodology will ensure that the developer building the right system according to the specification and verification checks the quality of the implementation. It also enables developers to develop more accurate system according to the user's discretion. This would help the developers to learn about the system and gain better understanding of the entire system.

3.5 Chapter Summary

This chapter has reviewed some existing software development life-cycle models. The modified waterfall model, i.e. waterfall model with prototyping works better in most cases than the classic waterfall model. It has the same breakdown of development tasks, but has the additional feature of feedback to make things better. Consequently, this approach can handle more dynamic projects. For most new system, however, this approach is still too linear in nature, even with the addition of feedback.

Chapter 4: System Analysis

System analysis is the most critical phase in a Software Development Life Cycle (SDLC). It is not only involving preliminary investigation and problem analysis as showed in previous chapters, but also to the requirement analysis. Hence, in this analysis workflow, I include functional requirement, non-functional requirement investigation, tools and technologies used, in order to refine my proposed system requirement to fulfill users' need.

4.1 Functional Requirement

A functional requirement is an essential part in system analysis workflow. It specifies the target product which must be performed by user. In addition, it shows the relationships between the system and its environment. A use-case has been drawn out to show how the users can use the available function or interaction with the Automated Essay Grading System.

4.1.1 Use-Case of the System

The use-case is shown as below for the requirement specification of Automated Essay Grading System. Followed by is a table of description of each system function that associates with the users.

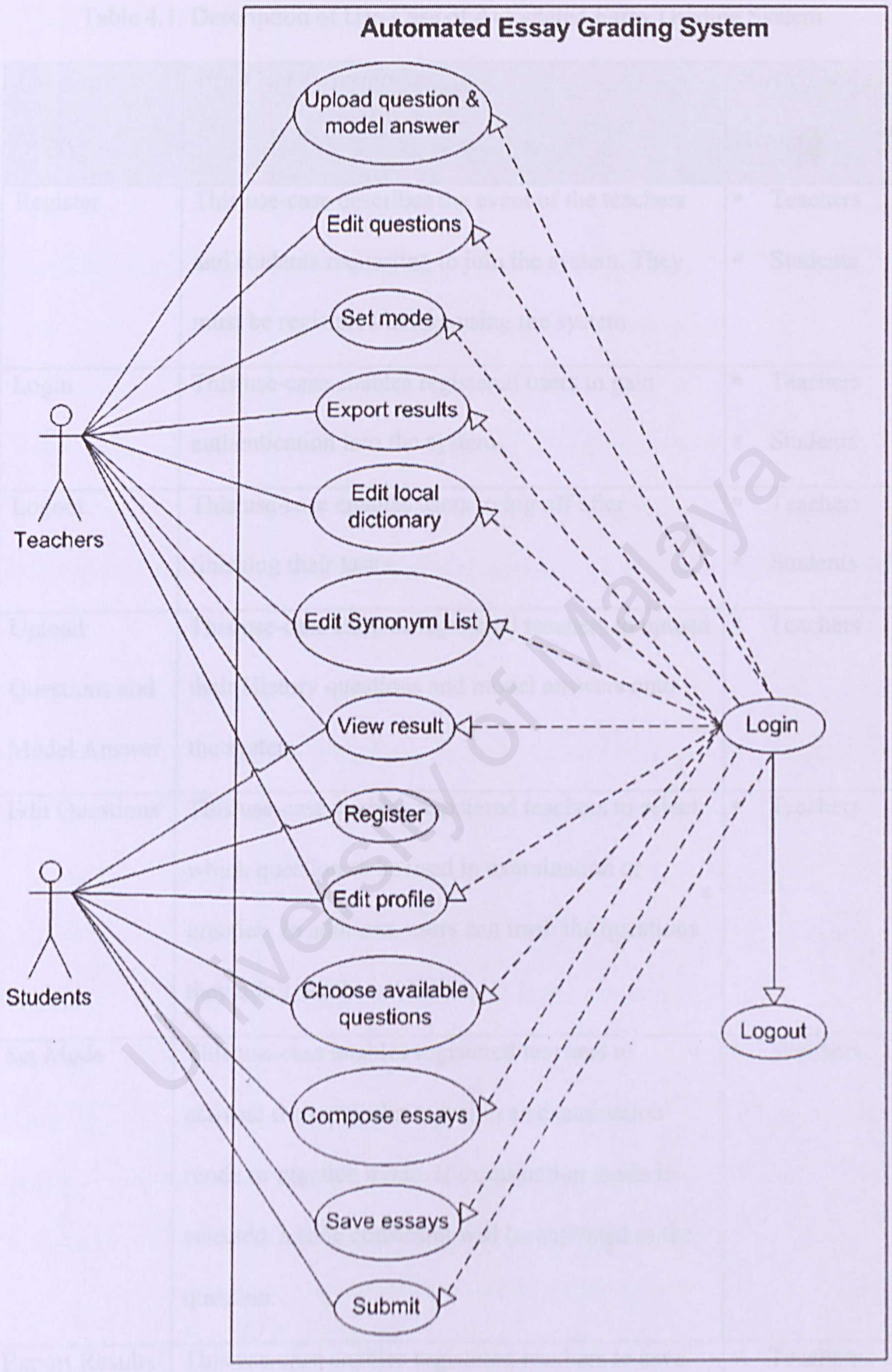


Figure 4.1: Use-case Diagram of Automated Essay Grading System

Table 4.1: Description of Use-Case of Automated Essay Grading System

Use-Case Name	*Use-Case Description*	*Participating Actors*
Register	This use-case describes the event of the teachers and students requesting to join the system. They must be registered before using the system.	▪ Teachers ▪ Students
Login	This use-case enables registered users to gain authentication into the system.	▪ Teachers ▪ Students
Logout	This use-case enables users to log off after finishing their tasks.	▪ Teachers ▪ Students
Upload Questions and Model Answer	This use-case enables registered teachers to upload their History questions and model answers onto the system.	▪ Teachers
Edit Questions	This use-case enables registered teachers to select which question to be used in examination or practice. In addition, users can trash the questions they don't want.	▪ Teachers
Set Mode	This use-case enables registered teachers to activate the particular question as examination mode or practice mode. If examination mode is selected, a time constraint will be activated to the question.	▪ Teachers
Export Results	This use-case enables registered teachers to save the results of their students into another file format such as excel or to be printed out.	▪ Teachers

Edit Local Dictionary	This use-case enables registered teachers to add or remove the contents of the local dictionary which is a collection of History terms and phrases, in order to smoothen the indexing part.	▪ Teachers
Edit Synonym List	This use-case enables registered teachers to add or remove synonyms from the list which is existed purposely for grading usage.	▪ Teachers
View Results	This use-case enables registered teachers to view their students' results; meanwhile the students can also view their results instantaneously. In addition, teachers also can view the students' answer script by using this function.	▪ Teachers ▪ Students
Edit Profile	This use-case enables registered teachers and students to update their personal details and their authentication access property like password.	▪ Teachers ▪ Students
Choose Available Questions	This use-case enables registered students to select their choices during the examination or practice.	▪ Students
Compose Essays	This use-case enables registered students to compose their essays by using a text editor.	▪ Students
Save Essays	This use-case enables registered students to save their essays on half-way during the examination, in order to prevent data loss.	▪ Students
Submit	This use-case will make sure registered students have answered all questions and submit their essays before leaving.	▪ Students

4.2 Non-Functional Requirement

Conversely, non-functional requirement specifies properties of the target product itself, such as platform constraints, response times or reliability. Thus, besides functional requirements, both our system, especially the indexing part also need to fulfill the non-functional requirement.

4.2.1 Accuracy

The system must operate to provide a less-error output during the stemming process. In order to increase the accuracy, the frequency of overstemming and understemming must be reduced.

4.2.2 Consistency

The system must ensure a consistent output as well. The outcome must be the same if same set of data is used for input. For example, if we use the same variation of a word, then the outcome or root word must always be the same.

4.2.3 Reliability

Another aspect that a system must be concerned is reliability. Reliability is a measure of the frequency and critically of product failure where it is an unacceptable effect under permissible condition. If a system is down, the recovery process also taken in account of a reliability measure.

4.2.4 Manageability

The system must be easy-to-manage, especially for a teacher. A teacher is allowed to add or remove their students' name in a class. Besides, students could also keep track on their result outcome.

4.2.5 Security

Security is always a concerned issue in every system or software product. All the questions, answers and results of the students as well as teachers must be only viewed or modified by rightful users only. It couldn't be easily hacked by other irresponsible person. To strengthen the security function, another local domain for registration is enabled for teachers only.

4.3 Proposed Tools and Technologies

A suitable tool is important to develop a system or software. Thus, after an analysis of not only to the indexing part but the whole system, a list of proposed tools and technologies used has been came out. Many aspects had been considered in order to choose the right and efficient tool to develop this system.

Table 4.2: Proposed Tools and Technologies

No.	*Tools and Technologies*	*Description*
1.	System Architecture	Client-server architecture
2.	Web Server	Internet Information Services (IIS)
3.	Programming Language	ASP.net, VisualBasic.net, Javascript
4.	Database Management System	Microsoft® SQL Server 2000
5.	Application Platform	Microsoft® Windows XP Professional with Internet Explorer compatible.
6.	Authoring Tool	Microsoft® Visual Studio.net 2003

4.3.1 Client-Server Architecture

For this project, automated essay grading system is designed to be client-server architecture as it is more suitable to be applied in the development of the system to perform its interactive functionality.

There are some advantages of client-server architecture:

- Processing can be centralized in the middle-tier, if three-tiered is using
- Easier to organize the implementation
- Allow different tiers to be developed in different languages
- Enhanced secure
- Support hundreds of users, making it more scalable
- Provides for more flexible resource allocation.
- Performance balancing
- Hides the complexity of deploying and supporting underlying services and network communications.

4.3.2 Internet Information Services (IIS) 5.1

Internet Information Services (IIS) 5.1 is complete personal web server available for all Microsoft® Windows XP Professional. It is designed for intranets, the Internet, and extranets, IIS 5.1 makes it possible for organizations of all sizes to quickly and easily deploy powerful Web sites and applications. In addition, IIS 5.1 provides a high-performance platform for applications built using the Microsoft .NET Framework.

Some other advantages of Microsoft Internet Information Services (IIS) are:

- Indexing, performance and security enhancements
- Easy to install and uninstall
- Well integrated server administration tools
- Easy to configure
- Web DAV support makes for easies collaborative publishing
- Support window based web authoring and development tool

4.3.3 ASP.NET

ASP.NET is a new generation of Active Server Page language. ASP.NET is a server-side language. It is compiled at the server before sending to the client PC. The ASP.NET engine provides a robust object model for creating dynamic content and is loosely integrated into the .NET framework. This integration makes it easy to change the implementation when the .NET framework migrates to platforms other than Windows (Russell Jones A., 2002).

4.3.4 Visual Basic.NET

Visual Basic.NET (VB.NET) is also a new generation of Visual Basic (VB). For your information, a VB.NET web application is equivalent to an ASP.NET application. In addition, VB.NET is more object-oriented.

VB.NET could be a code-behind to ASP.NET. Beside that, it plays an important role in database connection and data reader. Most of the user control, such as dropdownlist, list, text field, button and panel, could be done in dynamic-driven. They were formed depending on the database maintenance.

4.3.5 JavaScript

This is a client-side language. It is useful when composing some client-side function, such as message box and alert pop-up windows.

4.3.6 Microsoft® SQL Server 2000

Microsoft® SQL Server 2000 is the suitable choice for the development of automated essay grading system as it works well with databases of any size. Additionally, Microsoft SQL Server 2000 is the most robust database for the windows family.

Advantages of Microsoft® SQL Server 2000

- Able to support large-scale database
- Allow future expansion
- User-friendly queries
- High scalability, availability and reliable
- Ease of installation, deployment and use
- Work well with other Microsoft's component

- Can be queried and updated via Web browsers through integration with IIS

4.3.7 Microsoft® Visual Studio.NET 2003

Visual Studio .NET 2003 enables you to rapidly build a broad range of applications for Microsoft Windows, the Web, and mobile devices. Built to address today's most challenging software development needs, Visual Studio .NET 2003 enhances, further refines, and is highly compatible with its predecessor.

With Visual Studio .NET 2003, developers can deliver a range of professional software in record time. The integrated development environment (IDE) provides a consistent interface for all languages, including Microsoft Visual Basic .NET, Microsoft Visual C++ .NET, Microsoft Visual C# .NET, and Microsoft Visual J# .NET. Using the language best suited to your skill set, you can take advantage of shared visual designers to build rich Windows-based applications and dynamic Web applications that render in any browser.

Visual Studio .NET 2003 contains an updated version of the .NET Framework, version 1.1, that builds on the previous version with new capabilities and improved scalability, reliability, security, and performance. These core enhancements in the underlying development framework combine with improved IDE responsiveness to enable reduced costs associated with application development, deployment, and maintenance (*http://www.studentdiscounts.com/index.asp?PageAction=VIEWPROD&ProdID=3237*, 15/08/2004).

4.4 Hardware and Software Requirements

Both hardware and software requirements describe the constraints on computers and peripheral equipments. Hardware and software requirements need to be decided to determine the performance requirements feasibility. They are divided into runtime and development requirements.

Table 4.3: Hardware and Software Requirements

	Run Time	***Development***
Hardware Requirements	▪ 233 MHz Pentium / higher microprocessor / or equivalent ▪ Random Access Memory : 64 MB and above (128MB recommended) ▪ Hard disk : 2.5 GB and above ▪ Standard input and output ▪ Others standard computer peripherals	▪ 1.4 GHz Pentium 4 ▪ Random Access Memory : 128 MB ▪ Hard disk : 20GB ▪ Display : VGC display card ▪ Others standard computer peripherals
Software Requirements	▪ Windows 2000 server (Windows XP Professional is recommended) ▪ Any web browser (Internet Explorer 5.5 or above is recommended)	▪ Windows XP Professional ▪ Internet Explorer 6.0 ▪ Internet Information Server 5.1 with .NET framework

		▪ Microsoft SQL server 2000 ▪ ADO.NET ▪ Dreamweaver MX

4.5 Chapter Summary

Besides the tools and technologies that used for the system development, this chapter has also evaluated the functional requirement and non-functional requirement. But all these requirements needed to be implemented within budget and time constraints, so it's hard to satisfy all requirements in practical. As a result, the client or user may have to relax some requirement or increase the budget.

Chapter 5: System Design

System design is known as creative process of transforming the problem into a solution. The description of the system is called design. System design is the important stage of the waterfall because it translates the entire requirements for the system into the system characteristics.

5.1 Definition of Assigned Module

This system consists of two modules, i.e. preprocessing of query (indexing) and grading processing. The text in document and query will go through the same pre-processing before they are submitted to a grading processor to determine grade. I was assigned to the preprocessing of query which is the indexing, which includes sub-processes, like functional removals, stemming and feature recognition.

Functional removal is a form of lexical analysis. It removes the words or phrases which are functional and do not contribute to the contextual meaning of a document. Articles such as "the", "a", "an" and prepositions such as "at", "to" are examples of stop-word while phrases such as "in order to" or "such that" are called stop-phrase.

Stemming plays the most important part. Words in Malay language are reduced to probable root word by removing affixes such as prefix or suffix or both. Stemming is essential in information retrieval because spelling variations on a word are having equivalent in meaning. For example, the words such as "computerization", "computing" and "computation" are within the equivalent context of "compute".

Feature recognition shows the context meaning by adding tags label to the stemmed words. For example, #COLOR Black defines red as color while #NAME Black shows one's name.

5.2 System Architecture

5.2.1 General System

This will explain to you the architectural design of the Automated Essay Grading System in general term, which is about the whole system. Figure 5.1 shows the framework design of the system, which is consists of two major modules, i.e. indexing and grading.

The model answers (documents) that uploaded by teachers will be indexed before stored in the database. The answer scripts (queries) of students which come from the database will also be indexed. After both documents and queries been indexed, they will be sent to the grading engine.

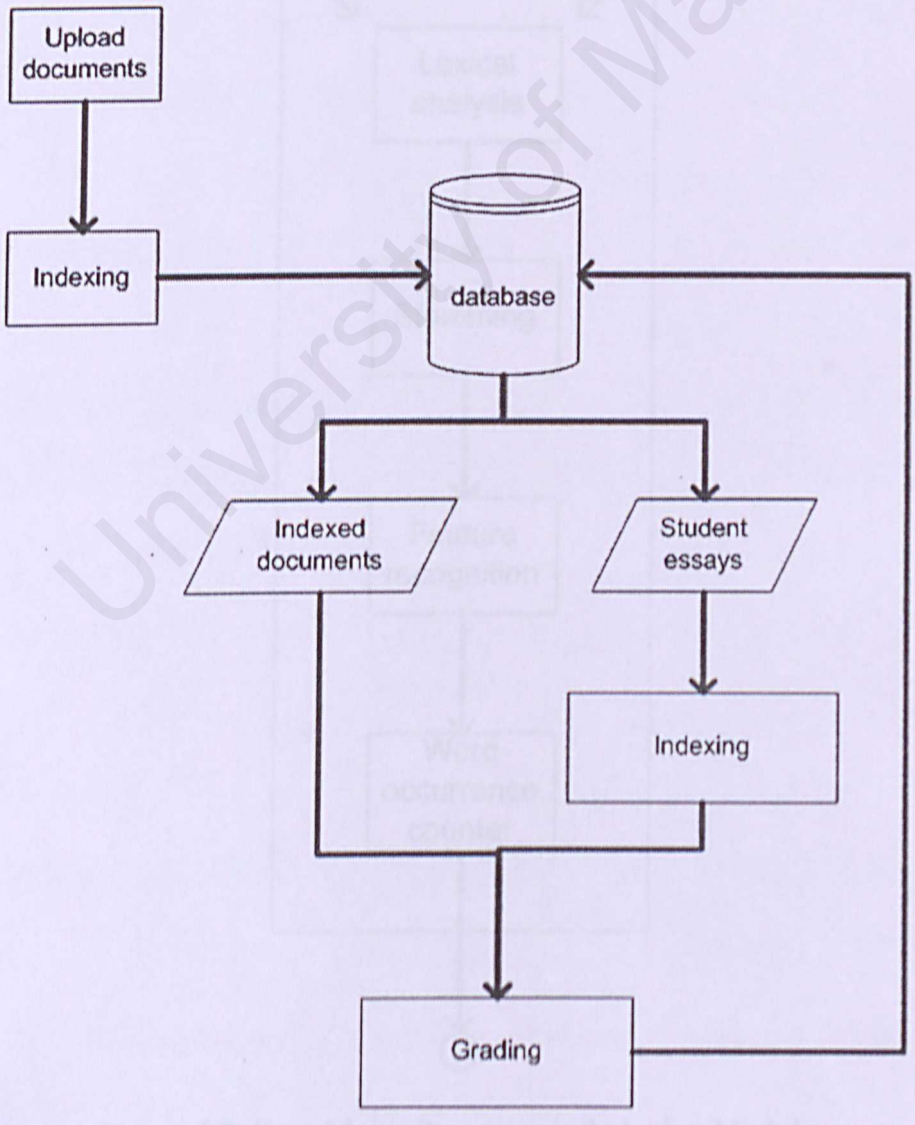


Figure 5.1: The Framework of Automated Essay Grading System

5.3 System Functionality Design

5.3.1 Indexing

This is the earliest part of the Automated Essay Grading System. Both the documents and queries will go through this document preprocessing before they are sent to be graded. In my design of this indexing module, it consists of four major processes, i.e. lexical analysis, stemming, feature recognition and word occurrence counting as you can see in Figure 5.2.

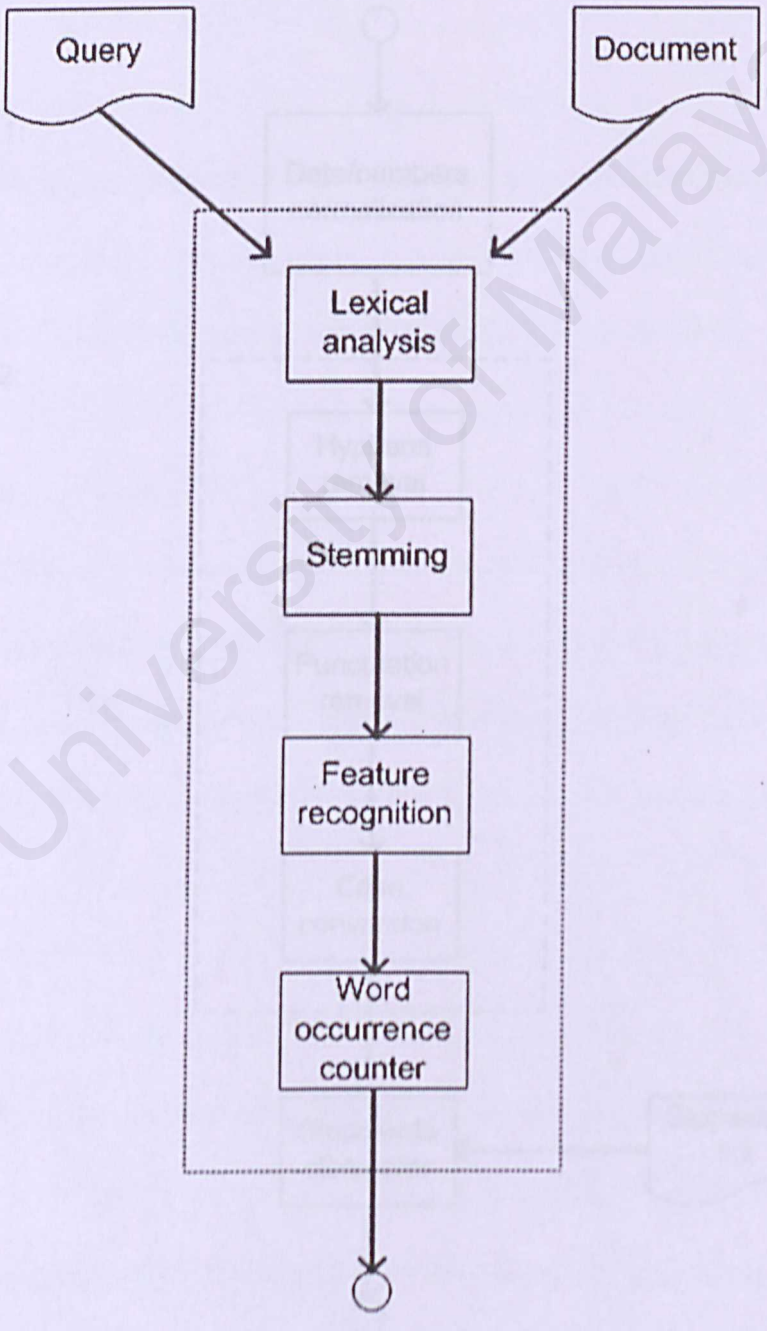


Figure 5.2: Four Major Processes in Indexing Module

5.3.1.1 Lexical Analysis

This lexical analysis consists of five processes which are divided into three steps. They are Date and Number Removal, Hyphens Removal, Punctuation Removal, Case Conversion, and Stopwords Elimination. As you can see from Figure 5.3, dates and numbers are removed in the first parse, followed by hyphens, punctuation and case conversion in the second parse of text. Finally, stopwords are eliminated from the documents and queries.

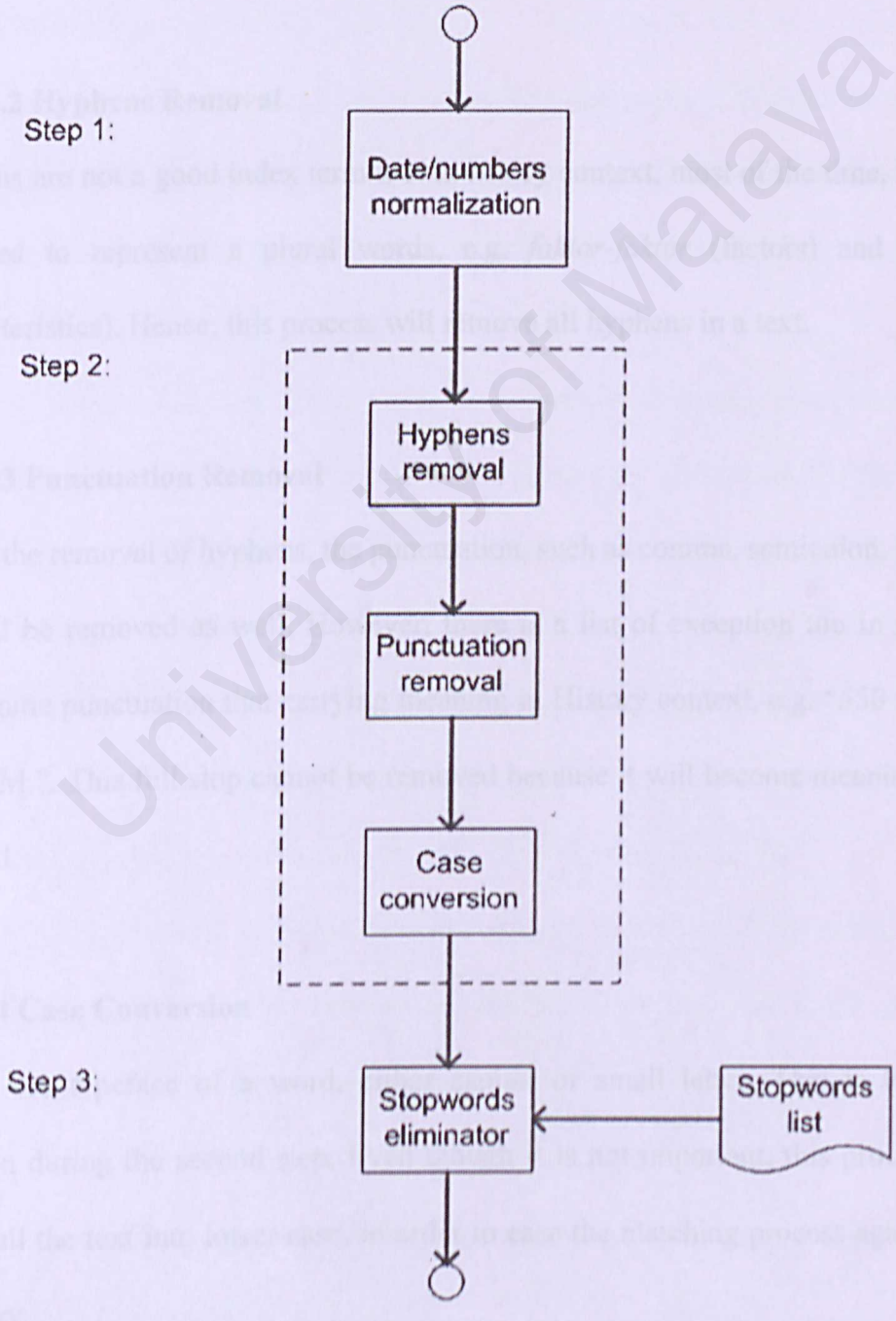


Figure 5.3: Structure of Lexical Analysis

5.3.1.1.1 Date Normalization

Numbers are usually not a good index term. However, sometimes numbers have some exceptions like in our system which designed specifically to History subject. Thus, we do a date and numbers normalization to the documents and queries. This is because some students may write a date 30/9 or 30 September. So, this process will convert all these various format into a standard form of date, such as DD/MM/YYYY, 25/12/1995.

5.3.1.1.2 Hyphens Removal

Hyphens are not a good index term too. In Malay context, most of the time, hyphens are used to represent a plural words, e.g. *faktor-faktor* (factors) and *ciri-ciri* (characteristics). Hence, this process will remove all hyphens in a text.

5.3.1.1.3 Punctuation Removal

During the removal of hyphens, the punctuation, such as comma, semicolon, full stop, etc. will be removed as well. However, there is a list of exception too in order to retain some punctuation that carrying meaning in History context, e.g. "550 S.M" or "550 S.M.". This full stop cannot be removed because it will become meaningless if removed.

5.3.1.1.4 Case Conversion

Case is the typeface of a word, either capital or small letter. This is the third condition during the second step. Even though it is not important, this process will change all the text into lower case, in order to ease the matching process against our dictionary.

5.3.1.1.5 Stopwords Eliminator

Stopwords like articles, preposition, conjunctions are always not a discriminator in a text. They do not carry any meaningful term in an essay. For example, *itu* (that), *di* (at), *mana* (where) and *kerana* (because). In addition, we will extend the list of stopwords to include some common words in History, e.g. *Sultan* (ruler in a state) and *Pembesar* (Minister). So, all the stopwords will be eliminated.

5.3.1.2 Stemming

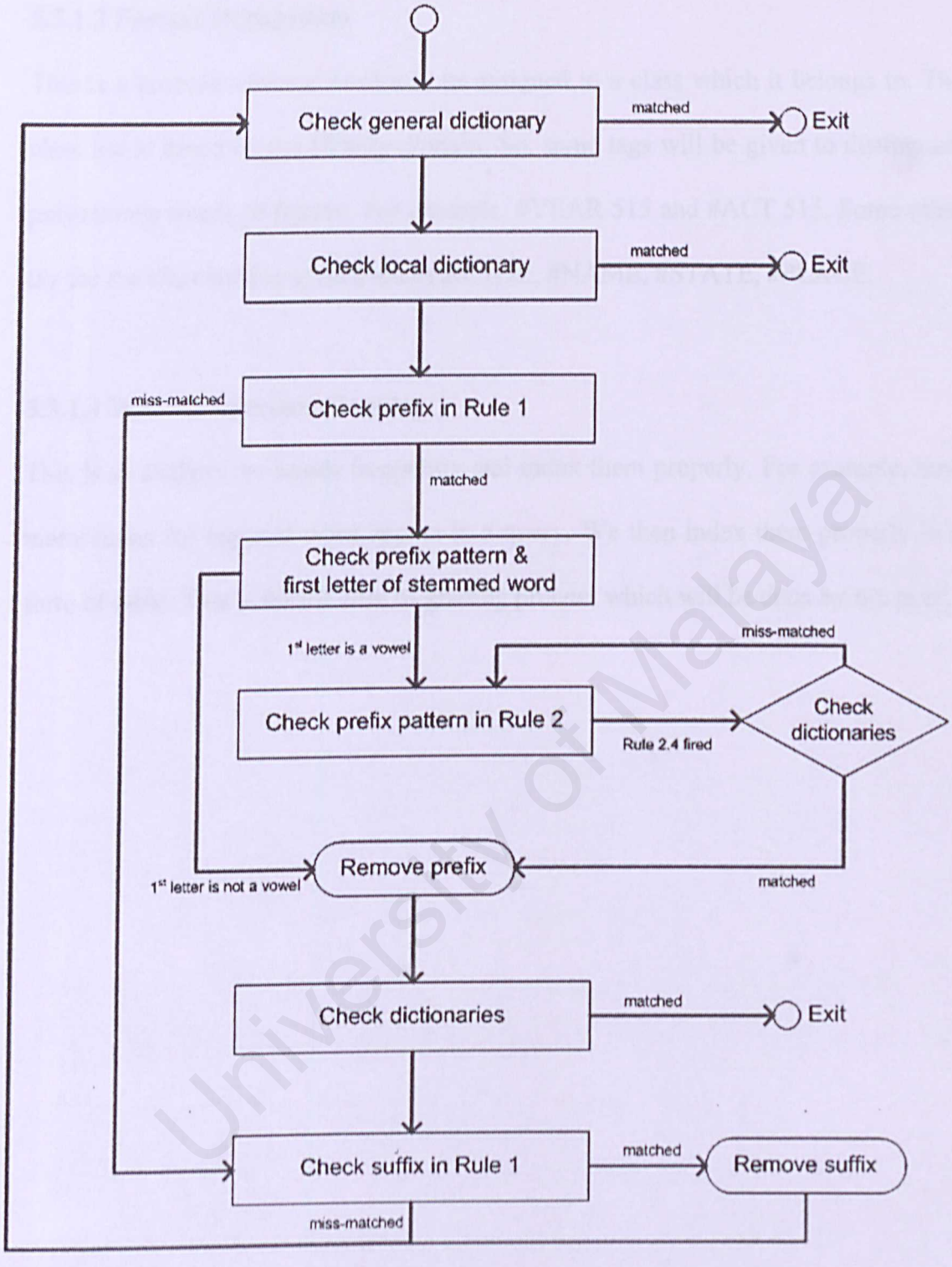
Stemming plays a very important role in this indexing process. Without stemming, it will prevent the perfect match between the query and the document. Besides, stemming helps reduces the variants of words, e.g. *berperang*, *berperangan* into *perang* (war).

As other Malay stemmer, we have a general dictionary also. Besides the general dictionary, we will have another local dictionary which contains the History terms, e.g. *pemerintah* (ruler). If stem it, became *perintah* (order), this already changed its context meaning. So, local dictionary is a subset of the general dictionary. All root words in general dictionary will be a root word in the local dictionary too. However, not every root words in local dictionary will appear in general dictionary.

Basically, our stemming algorithm as below (Norisma, 2002):

1. Step 1: Check the word against a general dictionary. If the word is found in the dictionary, then accept the word as the root word and exit; otherwise, proceed to the next step.
2. Step 2: Check the word against the local dictionary. If the word is found in the dictionary, accept the word as the root word and exit; otherwise, proceed to the next step.

3. Step 3: Check the word against the Prefix in Rule 1. If the word matches the Prefix rules, Check the pattern of the prefix and the first letter of the stemmed word; otherwise, go to Step 8.
4. Step 4: If the pattern of the prefix matches the prefix pattern in Rule 2 and the first letter of the stemmed word is a vowel, then, apply the Rule 2 to the words; otherwise, remove the prefix and go to Step 7.
5. Step 5: Check the prefix of the word against the pattern of the Rule 2. If it matches the fourth rule (Rule 2.4), then check the new stemmed word against the dictionary and proceed to the next step; otherwise, remove the prefix and go to Step 7.
6. Step 6: If the word is not found in the dictionary, then go back to Step 5; otherwise remove the prefix and proceed to the next step.
7. Step 7: Check the word in the dictionary. If the word is found in the dictionary, then accept the word as a root word and exit; otherwise, proceed to the next step.
8. Step 8: Check the word against the Suffix in Rule 1. If it matches with the Suffix rules, then remove the suffix and go to Step 1; otherwise just go to the Step 1.



Rule 1
Prefix: meng, meny, men, mem, me, peng, peny, pen, pem, di, ter, ke, ber, bel, be, per, pel, pe, se, sub, pra, pro, dwi
Suffix: an, al, at, kan, kah, lah, isme, i, nya, pun, wan, wati

Rule 2
2.1: replace 'men' or 'pen' with 't'
2.2: replace 'meng' or 'peng' with 'k'
2.3: replace 'meny' or 'peny' with 's'
2.4: replace 'mem' or 'pem' with 'f' or 'p'

Figure 5.4: The Algorithm of Malay Stemming

5.3.1.3 Feature Recognition

This is a process where a word will be assigned to a class which it belongs to. The class list is based on the History domain. So, some tags will be given to distinguish polysemous words or figures. For example, #YEAR 515 and #ACT 515. Some other tag for the class list being used are #FACTOR, #NAME, #STATE, #PLACE.

5.3.1.4 Word Occurrence Counter

This is to analyze the words frequently and index them properly. For example, how many times for *inggeris* word appear in a query. We then index them properly in a form of table. This is for the ease of grading process which will be done by my peer.

5.4 Database Design

Data storage is considered as the heart of an information system. It is a central source of data meant to be shared by many users for a variety of applications. The relational database model is used in database implementation for automated essay grading system. The database is constructed using Microsoft SQL Server 2000. Figure 5.5 shows the database logical model.

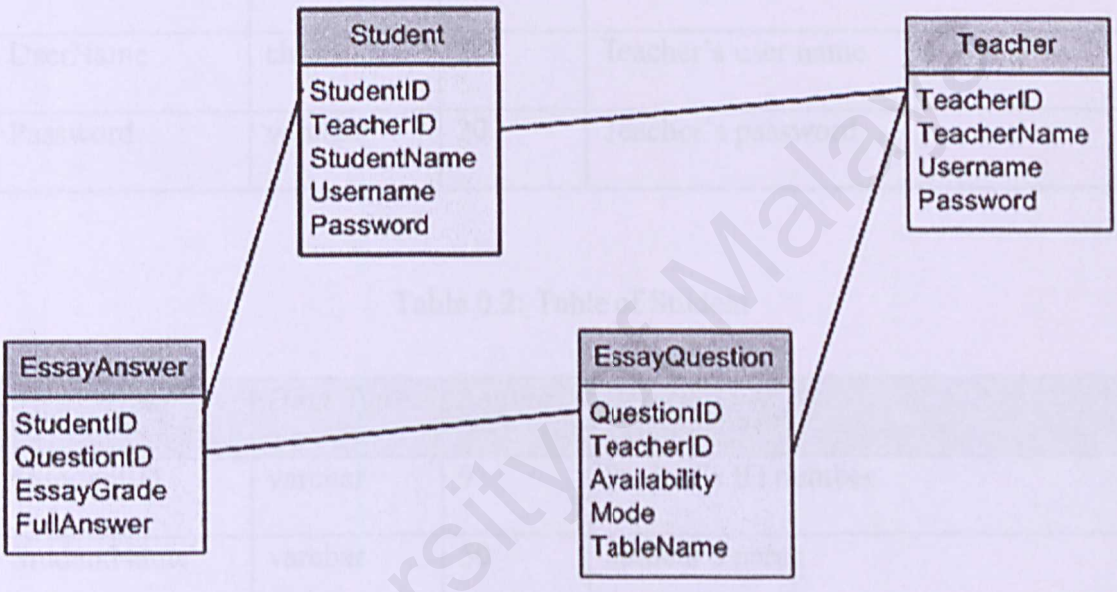


Figure 5.5: The Database Design for the Automated Essay Grading System

5.4.1 Tables of Database

Below are the tables of automated essay grading system

*** : primary key**

Table 0.1: Table of Teacher

Field Name	*Data Type*	*Length*	*Description*
*TeacherID	varchar	9	Teacher's ID number
TeacherName	varchar	50	Teacher's name
UserName	char	20	Teacher's user name
Password	varchar	20	Teacher's password

Table 0.2: Table of Student

Field Name	*Data Type*	*Length*	*Description*
*StudentID	varchar	9	Student's ID number
StudentName	varchar	50	Student's name
UserName	varcahr	20	Student's user name
Password	varchar	20	Student's password
TeacherID	varchar	9	Student's teacher

Table 0.3: Table of EssayQuestions

Field Name	*Data Type*	*Length*	*Description*
*QuestionID	nvarchar	10	Question ID
TeacherID	char	9	Teacher's ID
Availability	nvarchar	1	Determine whether the question is available to the students or not.
Mode	nvarchar	10	Determine whether it is a test, quiz or exercise.
TableName	nvarchar	100	Table name of the model answer

Table 0.4: Table of EssayAnswers

Field Name	*Data Type*	*Length*	*Description*
*StudentID	nvarchar	9	Student's ID
QuestionID	char	10	Question ID
EssayGrade	nvarchar	10	Grade of the essay
FullAnswer	Text		Full answer of the student essay

5.5 Prototypes of User Interface Design

Some prototype of the Graphical User Interface (GUI) designs had been done in the early stage. Below are the screen shots:

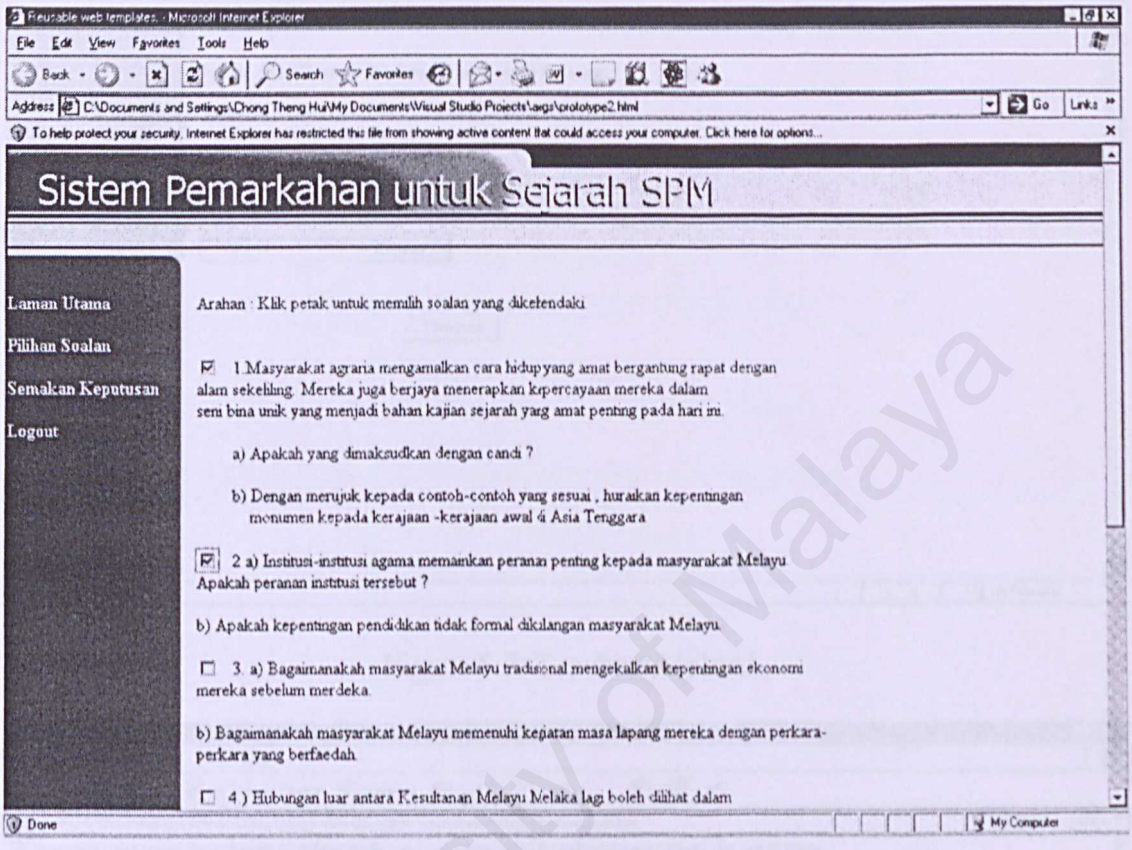


Figure 5.6: The Set Question Screen

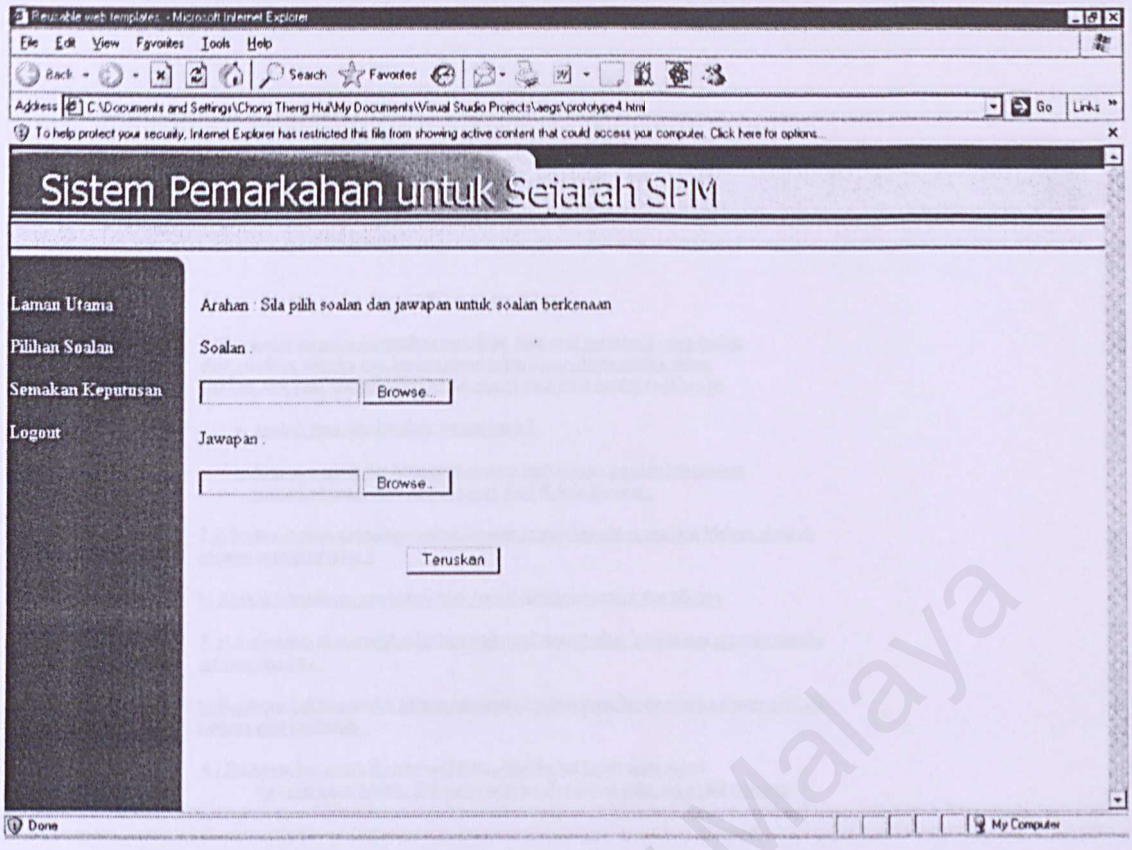


Figure 5.7: Teachers Upload

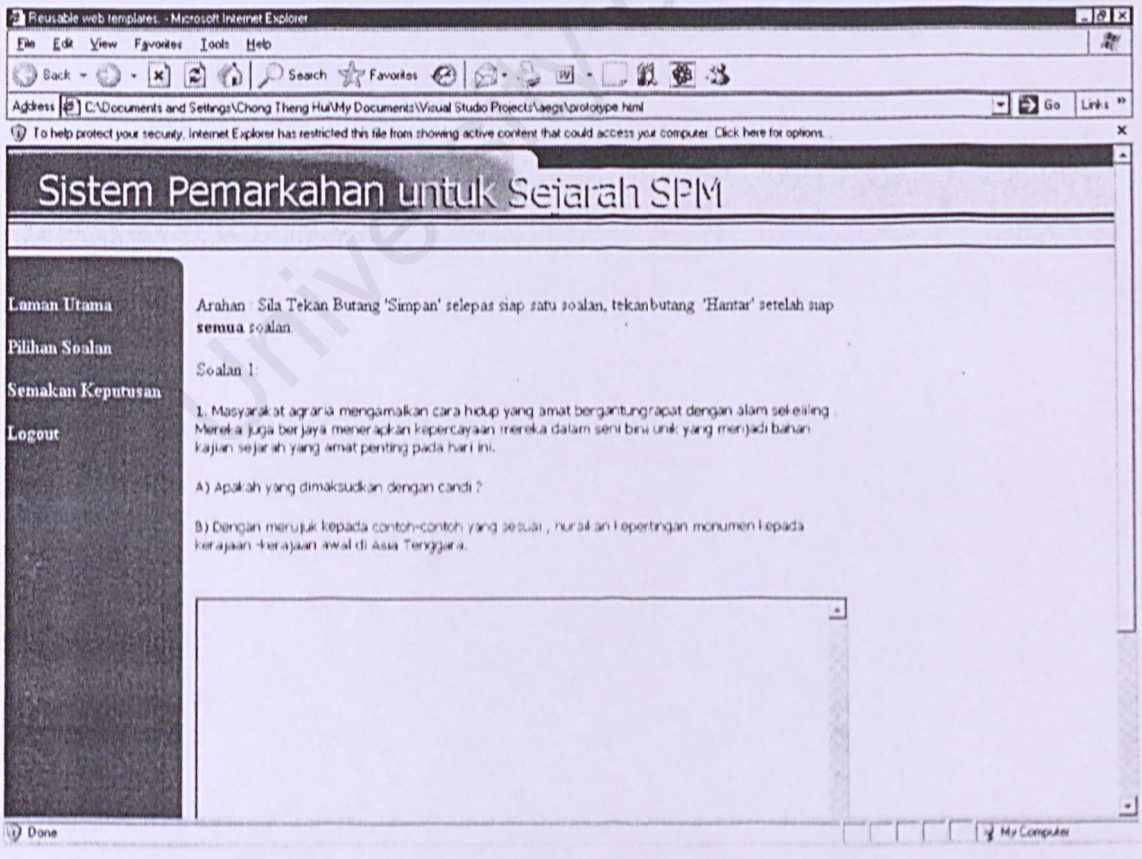


Figure 5.8: Students Essays Composer

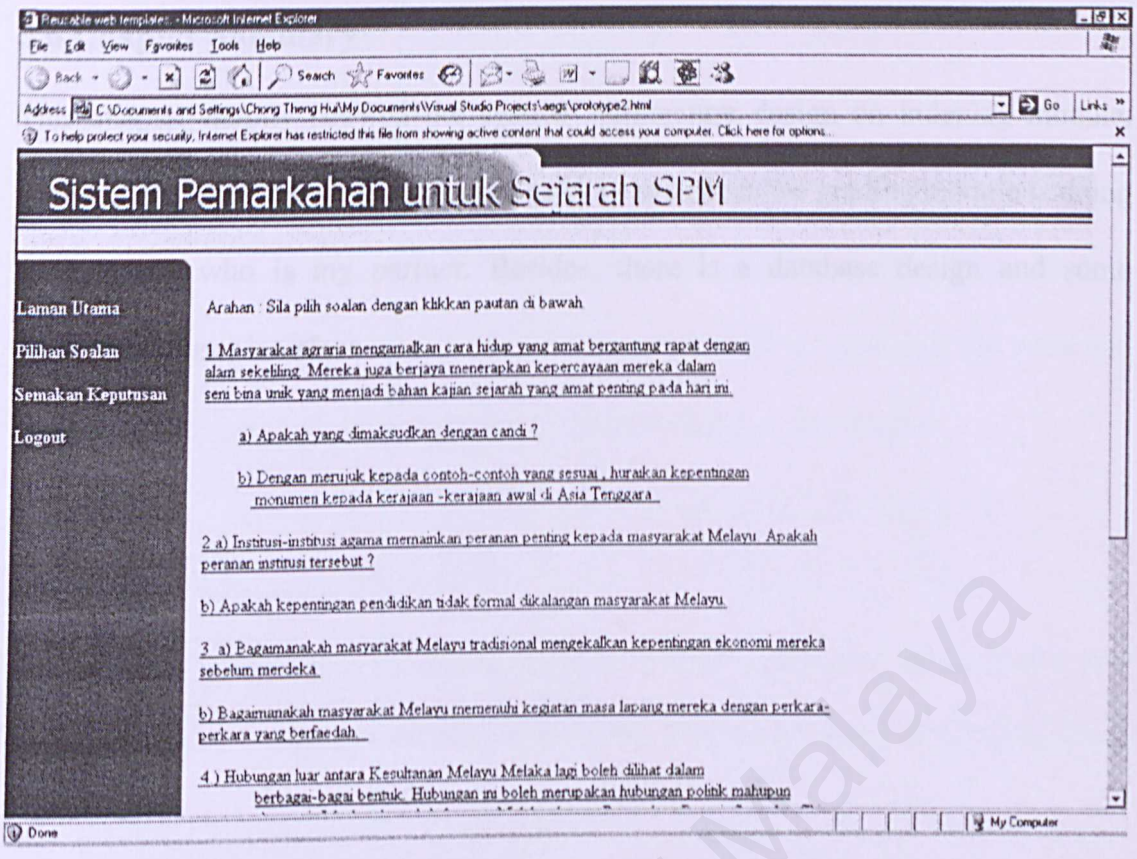


Figure 5.9: Questions Selection Screen

5.6 Chapter Summary

This chapter mainly showed the system architecture design on indexing module, rather than the whole automated essay grading system as the grading module is doing by my peer who is my partner. Besides, there is a database design and some prototype of user interface.

Chapter 6: System Development and Implementation

System development is known as a coding process of transforming the systems specification into an executable or working real-system. The major processes of transforming the Indexing algorithms into working instruction will be described in this chapter. Besides, the development environment, both in hardware and software, as well as the chosen programming languages are stated in this chapter.

6.1 Development Environment

Development environment is the description of the equipment of hardware and software which I was using when developing this Automated Essay Grading System.

6.1.1 Hardware

Below is the hardware configuration that used to develop this system:

- Intel Pentium III M 1GHz
- 384 MB RAM
- 40GB hard disk with approximately 10GB of fee space
- 14.1" LCD
- Other standard peripherals

6.1.2 Software

The software tools that were used to develop this system are as below:

- Microsoft Windows XP Professional with Service Pack 2
- Microsoft Internet Information System 5.1
- Microsoft Internet Explorer 6.0
- Microsoft .Net Framework 1.1

- Microsoft Visual Studio.Net 2003
- Microsoft SQL Server 2000
- Adobe Photoshop CS
- Macromedia Flash MX 2004

6.2 System Implementation

Since my major module is Indexing of this Automated Essay Grading System, so it is emphasized in the following part. However, this doesn't mean that I ignore the other modules. The full code will be attached in the appendix behind this dissertation.

6.2.1 Programming Languages and Approach

This Automated Essay Grading System is a web-based real-time system. So, we had decided to choose ASP.NET with Visual Basic.NET as code-behind as our main stream language. This is a strong and new stylish fourth generation programming language. We chose this language due to its powerful integration of web-based feature which suited to our system requirement. Also, the Transact-Structured Query Language (T-SQL) which been used is one of the back-end languages that play an important rule in the system backbone. Apart from that, Javascript had been chosen to run some function at client-side, such as windows pop-up message box or validation checking.

I strongly adapt myself to having a good programming practice, e.g. the use of consistent and meaningful variable names. Besides, comments are remarked so that other programmer (especially my partner) could understand them easily. The code layout for increased readability is not a matter in our Integrated Development Environment kit, i.e. Microsoft Visual Studio.NET because it has neatened the code

layout automatically for programmer. In addition, we have a name convention standard for all the variables used. Please refer to Appendix A.

6.2.2 System Modules and Functionalities

The Indexing module consists of four major processes, i.e. Lexical Analysis, Stemming, Feature Recognition and Word Occurrence Counter. All these processes had been separated into sub-functions or sub-classes (engines), so that easy for implementing into our coding. Moreover, this approach could ease the maintenance work on this system in the future and fulfill the code reuse objective.

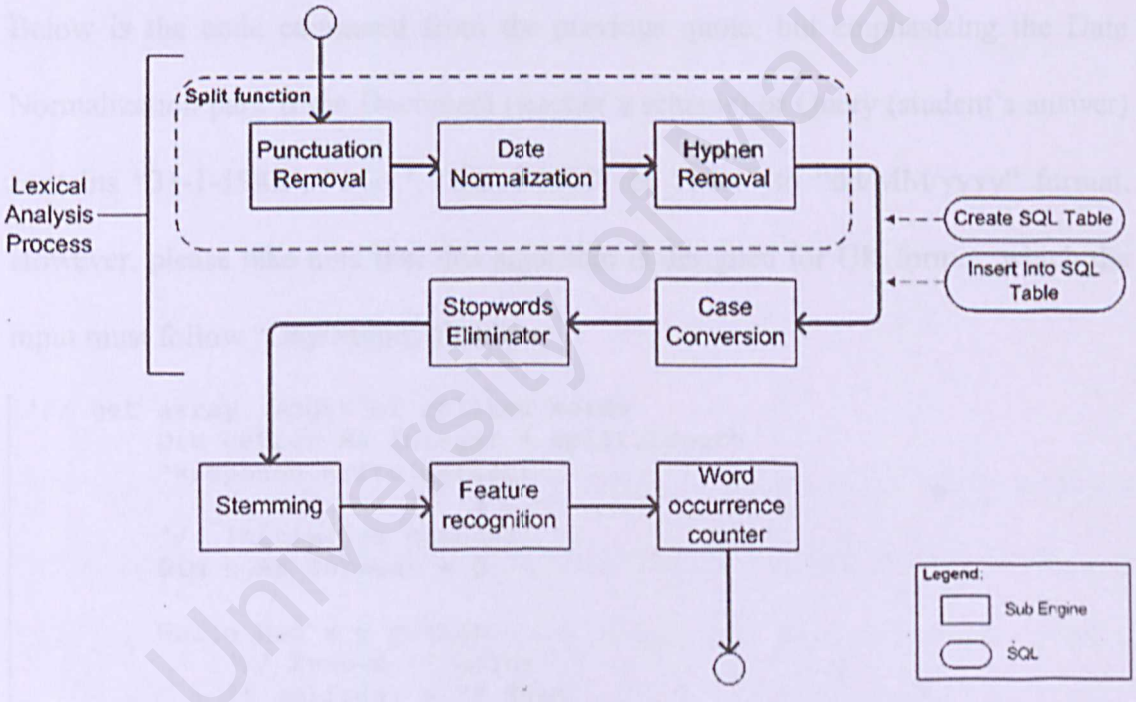


Figure 6.1: Design of Engines and Function in Indexing Module

6.2.2.1 Lexical Analysis Processes

The Lexical Analysis process had been broken into five sub engines, i.e. Punctuation Removal, Date Normalization, Hyphen Removal, Case Conversion and Stopwords Eliminator. Meanwhile, the first three sub engines are belonged to Split function which splits the sentences into words.

6.2.2.1.1 Punctuation Removal

Below shows the continuous code block of Split function. At the beginning of the code shows how the punctuation been removed.

```
Private Sub split(ByVal txtScheme As String, ByVal tableName As _
String)
        Dim delimStr As String = " ,'.:()""@ "
        Dim delimiter As Char() = delimStr.ToCharArray()

        '// Declare array of splited Words
        Dim split As String() = Nothing
        Dim updatedSplit As String() = Nothing
        split = txtScheme.Split(delimiter)
```

6.2.2.1.2 Date Normalization

Below is the code continued from the previous quote, but emphasizing the Date Normalization part. If the Document (teacher's scheme) or Query (student's answer) contains "31-1-1982", "11-1", "20/12/1908" or "18/4" to "dd/MM/yyyy" format. However, please take note that this algorithm is designed for UK format, which the input must follow "Day/Month/Year".

```
'// get array length of splited words
        Dim getLen As Integer = split.Length
        'Response.Write(getLen)

        '// Initialize counter
        Dim x As Integer = 0

        While Not x = getLen
            '// Remove "" words
            If split(x) = "" Then
                x += 1

            Else
                '// Date Normalization
                If split(x) Like "##-##-####" Then
                    split(x) = Format(Cdate(split(x)), _
"dd/MM/yyyy")
                ElseIf split(x) Like "#-##-####" Then
                    split(x) = Format(Cdate(split(x)), _
"dd/MM/yyyy")
                ElseIf split(x) Like "##-#-####" Then
                    split(x) = Format(Cdate(split(x)), _
"dd/MM/yyyy")
                ElseIf split(x) Like "#-#-####" Then
                    split(x) = Format(Cdate(split(x)), _
"dd/MM/yyyy")
```

```
                ElseIf split(x) Like "##/##/####" Then
                    split(x) = Format(CDate(split(x)), _
"dd/MM/yyyy")
                ElseIf split(x) Like "#/#/####" Then
                    split(x) = Format(CDate(split(x)), _
"dd/MM/yyyy")
                ElseIf split(x) Like "##/#/####" Then
                    split(x) = Format(CDate(split(x)), _
"dd/MM/yyyy")
                ElseIf split(x) Like "#/#/####" Then
                    split(x) = Format(CDate(split(x)), _
"dd/MM/yyyy")

                ElseIf split(x) Like "##-##" Then
                    split(x) = Format(CDate(split(x)), "dd/MM")
                ElseIf split(x) Like "#-##" Then
                    split(x) = Format(CDate(split(x)), "dd/MM")
                ElseIf split(x) Like "##-#" Then
                    split(x) = Format(CDate(split(x)), "dd/MM")
                ElseIf split(x) Like "#-#" Then
                    split(x) = Format(CDate(split(x)), "dd/MM")

                ElseIf split(x) Like "##/##" Then
                    split(x) = Format(CDate(split(x)), "dd/MM")
                ElseIf split(x) Like "#/##" Then
                    split(x) = Format(CDate(split(x)), "dd/MM")
                ElseIf split(x) Like "##/#" Then
                    split(x) = Format(CDate(split(x)), "dd/MM")
                ElseIf split(x) Like "#/#" Then
                    split(x) = Format(CDate(split(x)), "dd/MM")
                End If
```

6.2.2.1.3 Hyphen Removal

This algorithm changes the double word which represents plural words in Malay context into single word by removing the hyphen in between the two words, e.g. *faktor-faktor* (factors) dan *syarat-syarat* (conditions). Finally, the split word is entered into queue which is storing in computer memory. The split function ends here.

```
                'Hyphens Removal
                If split(x) Like "[A-z]*-[a-z]*" Then
                    split(x) = split(x).Substring(0, _
split(x).IndexOf("-"))
                End If

                wordQueue.Enqueue(split(x).Trim)
                x += 1
            End If

        End While
    End Sub
```

6.2.2.1.4 Case Conversion

Every single split word is changed into small case letters when inserting into a SQL table in order to ease the process of matching later on. Below is the Create SQL Table sub engine.

```
Namespace Engine
    Public Class clsCreateSQLTable
        '//declare strConn
        Private strConn As String =
System.Configuration.ConfigurationSettings.AppSettings.Get("Connect
ionString")
        Private objSqlHelper As SqlHelper

        '// Create a temp table for storing words
        Public Sub CreateSQLTable(ByVal tableName)
            Dim myConn As New SqlClient.SqlConnection(strConn)
            Dim myCmd As New SqlClient.SqlCommand
            'Dim tableName As String = "ANS_020031_T001_Q03"

            myConn.Open() 'MUST open
            'Assign connection to cmd1.
            myCmd.Connection = myConn

            'Execute query to drop prior version of table.
            myCmd.CommandText = "IF EXISTS (" & _
            "select * from dbo.sysobjects " & _
            "where id = object_id(N'[dbo].[" + tableName + "]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)" & _
            "DROP TABLE dbo." + tableName
            myCmd.ExecuteNonQuery()

            'Execute query to create new version of table
            myCmd.CommandText = "CREATE TABLE [dbo].[" + tableName
+ "]  " & _
            "(" & _
            "[ID] [int] IDENTITY (1, 1) NOT NULL , " & _
            "[ansWord] [varchar] (50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL , " & _
            "[classType] [varchar] (50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL ,  " & _
            "[frequency] [float] NULL ,  " & _
            "[weightVSM] [float] NULL ,  " & _
            "[weightNN] [float] NULL ,  " & _
            "[att1] [float] NULL ,  " & _
            "[att2] [float] NULL ,  " & _
            "[att3] [float] NULL ,  " & _
            "[att4] [float] NULL ,  " & _
            "[att5] [float] NULL   " & _
            ")"
            myCmd.ExecuteNonQuery()

            myConn.Close()
        End Sub
    End Class
End Namespace
```

This is followed by the sub engine of SQL Table Insertion. The word queue is passed into the class for manipulation.

```
Namespace Engine
    Public Class clsInsertWords
        '//declare strConn
        Private strConn As String =
System.Configuration.ConfigurationSettings.AppSettings.Get("Connect
ionString")
        Private objSqlHelper As SqlHelper

        Public Sub insertWords(ByRef lstQ As Queue, ByVal tableName
As String)
            Dim myConn As New SqlClient.SqlConnection(strConn)
            Dim myCmd As New SqlClient.SqlCommand
            myConn.Open()
            myCmd.Connection = myConn

            While Not lstQ.Count = 0
                'Dequeue and Insert words into Table
                myCmd.CommandText = _
                "INSERT INTO " + tableName + " " & _
                "(ansWord) " & _
                'Change to LOWER CASE
                "VALUES ( LOWER( '" + lstQ.Dequeue() + "' ) )"
                myCmd.ExecuteNonQuery()
            End While

            myConn.Close()
        End Sub

    End Class
End Namespace
```


6.2.2.1.5 Stopwords Eliminator

The words inside the table are then matched to a list of Stopwords. The word is eliminated from the table if it matches with the stopword list. Below is the Stopword elimination engine and the stored procedure.

```
Namespace Engine
    Public Class clsElimStopWords
        '//declare strConn
        Private strConn As String =
System.Configuration.ConfigurationSettings.AppSettings.Get("Connect
ionString")
        Private objSqlHelper As SqlHelper

        Public Sub elimStopWords(ByVal tableName As String)
            Dim objcls As New clsSqlHelper
            Dim objParams As SqlParameter()
            objcls.getParameterSet(objParams,
"usp_INDEXING_Stopword_Eliminator")
            objParams(0).Value = tableName.Trim
            objcls.execSP(objParams,
"usp_INDEXING_Stopword_Eliminator")
        End Sub

    End Class
End Namespace
```

```
ALTER        proc usp_INDEXING_Stopword_Eliminator
    @tableName varChar(50)
as
SET NOCOUNT ON
Declare @SQL varChar(1000)
SET @SQL = '


DELETE FROM '+@TableName+'
WHERE ansWord IN (
SELECT stopWord FROM List_Of_Stopword )

'
EXEC(@SQL)

SET NOCOUNT OFF
```


6.2.2.2 Stemming Process

This process is to get the root words of each entered words. It truncates the prefixes, suffixes of a word, according to my designed algorithm. Due to the long coding of Stemming process, below is just the pseudocode for easy understanding. The full codes of it could be found in the appendix.

```
stemming()
{
 while X <= maximum word count
     get word(X);
prefix:
     Check_general_dictionary();
     Check_local_dictionary();
     if not match both dictionary {

         check_prefix_rule_1();
         if match List_Of_Prefix
             word(X) = @prefix + @stemmedWord;

         check_prefix_rule_2();
         if @prefix = 'men' or 'pen'
             if @firstLetter like '[aeiou]'
                 word(X) = 't' + @stemmedWord;
         else if @prefix = 'meng' or 'peng'
             if @firstLetter = 'e'
                 set @sTemp = word(X);
                 check_suffix();
                 check_general_dictionary();
                 if match
                     goto 'endRule2';
                 else
                     restore Word(X);

             if @firstLetter like '[aeiou]'
                 word(X) = 'k' + @stemmedWord;
         else if @prefix = 'meny' or 'peny'
             if @firstLetter like '[aeiou]'
                 word(X) = 's' + @stemmedWord;
         else if @prefix = 'mem' or 'pem'
             if @firstLetter like '[aeiou]'
                 word(X) = 'f' + @stemmedWord;
                 check_suffix();
                 if match
                     truncate suffix;
                 check_general_dictionary();
                 if not match
                     word(X) = 'p' + @stemmedWord;

endRule2:
         check_general_dictionary();
         if match
             do nothing;
         check_suffix();
         if match
             truncate suffix;

         check_double_prefix();
         if @prefix = 'per' or 'ter or 'se'
             goto 'prefix'
     }
 end while
}
```

6.2.2.3 Feature Recognition Process

Each word is assigned to a class according to the class list, which is stored in the List of Local Dictionary table. For unknown class case, the word is automatically assigned to #UNKNOWN class. Below is the engine in VB.NET and stored procedure codes.

```
Namespace Engine
    Public Class clsFeatureRecognition
        '//declare strConn
        Private strConn As String =
System.Configuration.ConfigurationSettings.AppSettings.Get("Connect
ionString")
        Private objSqlHelper As SqlHelper

        Public Sub startFeatureRecognition(ByVal tableName As
String)
            Dim objcls As New clsSqlHelper
            Dim objParams As SqlParameter()
            objcls.getParameterSet(objParams,
"usp_INDEXING_Feature_Recognition")
            objParams(0).Value = tableName.Trim
            objcls.execSP(objParams,
"usp_INDEXING_Feature_Recognition")
        End Sub

    End Class
End Namespace
```

```
ALTER       proc usp_INDEXING_Feature_Recognition
    @tableName varChar(30)
as

SET NOCOUNT ON
DECLARE @SQL varChar(8000)
SET @SQL = '
DECLARE @i int
DECLARE @strWord varChar(50)
DECLARE @boolLocalDict int
DECLARE @class varChar(50)

    SET @i = 1

    WHILE @i <= ( SELECT MAX(ID) FROM '+ @tableName +' )
    BEGIN
        SET @strWord = ( SELECT ansWord FROM '+ @tableName +' WHERE
ID = @i )
        set @boolLocalDict =
dbo.func_Check_Local_Dictionary(@strWord)
        if ( @boolLocalDict = 1 )
        begin
            set @class = ( SELECT classType
                           FROM dbo.List_Of_Local_Dictionary
                           WHERE localWord = @strWord
                         )
```

```
            UPDATE '+ @tableName +'
            SET classType = @class
            WHERE ID = @i
        end
        else
        begin
            UPDATE '+ @tableName +'
            SET classType = ''#UNKOWN''
            WHERE ID = @i
        end

        SET @i = @i + 1
    END -- end WHILE

'
EXEC (@SQL)

SET NOCOUNT OFF
```

6.2.2.4 Word Occurrence Counter Process

This process is created for the use of grading part which was done by my partner. It counts the frequency of a word and calculates by using the Nearest Neighbour and Vector Space Model method. The engine code and stored procedure are shown below.

```
Namespace Engine
    Public Class clsWordOccur
        '//declare strConn
        Private strConn As String =
System.Configuration.ConfigurationSettings.AppSettings.Get("Connect
ionString")
        Private objSqlHelper As SqlHelper

        Public Sub startWordOccur(ByVal tableName As String)
            Dim objcls As New clsSqlHelper
            Dim objParams As SqlParameter()
            objcls.getParameterSet(objParams,
"usp_INDEXING_Word_Occurrence")
            objParams(0).Value = tableName.Trim
            objcls.execSP(objParams,
"usp_INDEXING_Word_Occurrence")
        End Sub

    End Class
End Namespace
```

```
ALTER   proc usp_INDEXING_Word_Occurrence
    @tableName varChar(30)
as
DECLARE @SQL varChar(8000)
SET @SQL = '

DECLARE @i int
DECLARE @count int
DECLARE @strWord varChar(50)

    SET @i = 1
    WHILE @i <= ( SELECT MAX(ID) FROM '+ @tableName +' )
    BEGIN
        --// get word fr Students answer
        SET @strWord = ( SELECT ansWord FROM '+ @tableName +' WHERE
ID = @i )

        SET @count = ( SELECT COUNT (ansWord) FROM '+ @tableName +'
WHERE ansWord = @strWord)

        --// BEGIN VSM and NN weight
        UPDATE '+ @tableName +'
        SET frequency = @count
,           weightVSM = square(@count)
,           weightNN = (@count)
        WHERE ansWord = @strWord
        --// END VSM and NN weight

        DELETE
        FROM '+ @tableName +'
        WHERE ansWord = @strWord AND ID <> @i

        SET @i = @i + 1
    END
'
EXEC (@SQL)
```

6.3 Chapter Summary

The development of Automated Essay Grading System has been carefully planned and it does take a period of precious time. A lot of the functionalities as described in the previous chapter have been implemented orderly in time. Some use cases have been added in order to improve system logical performance. In short, AEGS has fully utilized the Microsoft technologies to build up the system. The next chapter would draw the system testing procedures on AEGS.

Chapter 7: System Testing

Testing step is a critical process in discovering logical error and data flow of the program besides determine the system reliability. In short, the main purpose of doing system testing is to ensure the system quality, form either aspect of specification, algorithm design or coding. In fact, software or system defect could only be found out through testing.

In developing a system, testing had been carried out throughout during the whole process. Usually, a testing consists of several steps, as shown in the figure below.

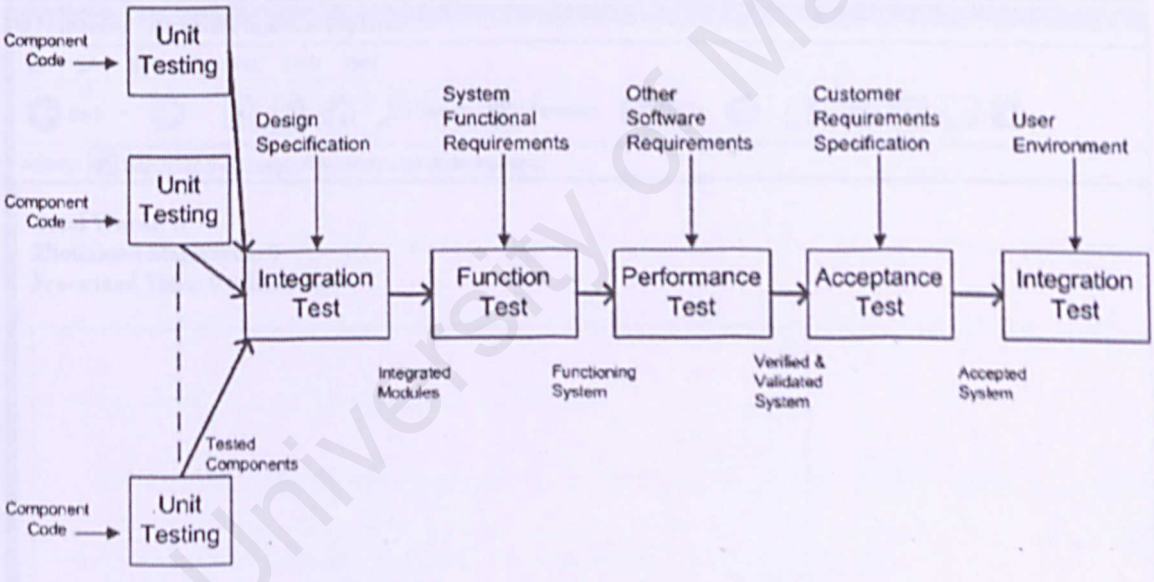


Figure 7.1: System Testing Flow Example

Generally, there are three testing stages altogether, i.e. Unit Testing, Integration Testing and System Testing, which have been practiced in our AEGS development period indeed. In the ending of the each testing description below, a sample of my testing is attached for supporting the points.

7.1 Unit Testing

This was the first stage of testing process. Each of the artifacts has been tested independently and isolated from other components. The component function has been verified in this stage by testing its input and seeing the output. Every single unit of component was ensured to function properly as required so that we could integrate them in the next step and test again.

There are quite a number of approaches of Unit Testing, e.g. Ad hoc tests, Black box tests and White box (Glass box) tests. Below is the figure of my testing platform.

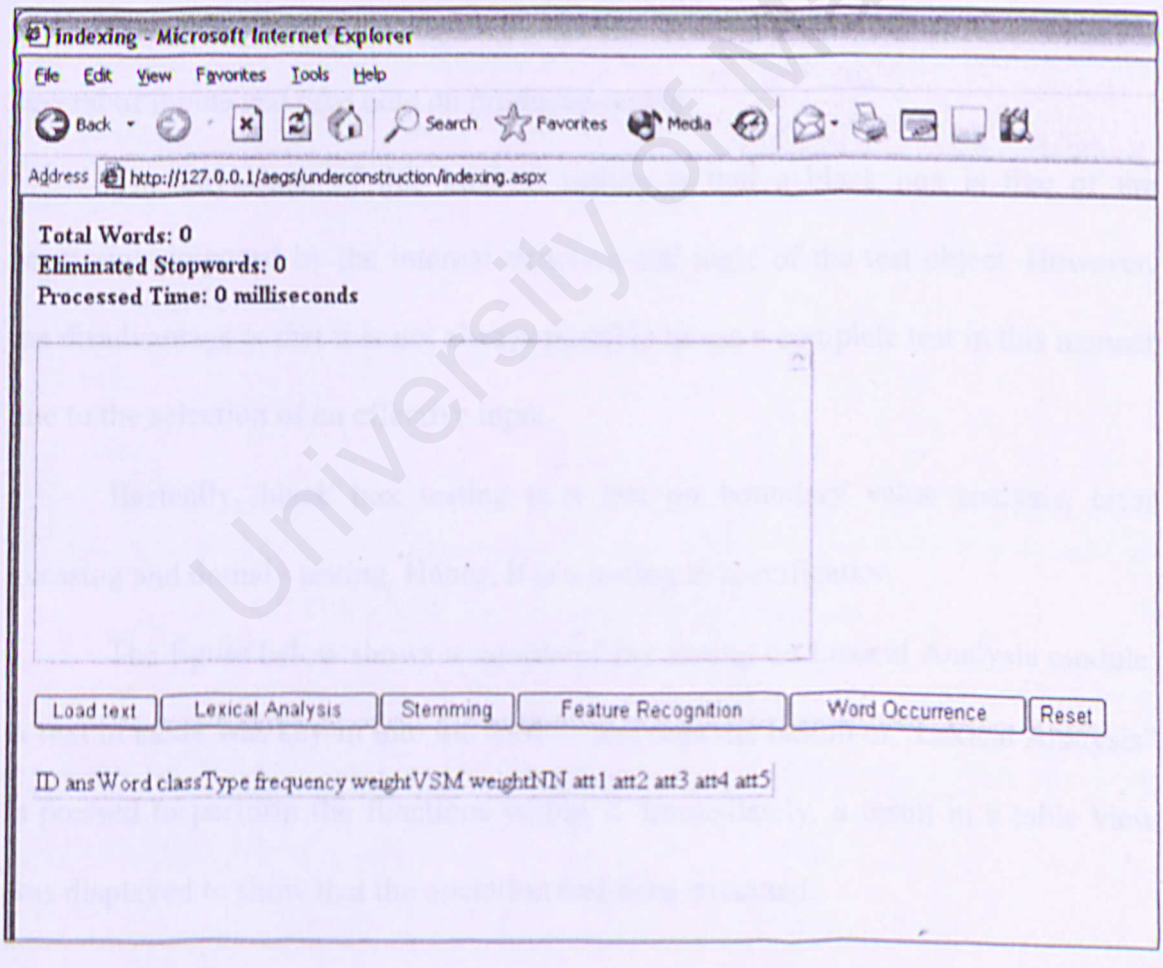


Figure 7.2: Testing Platform of AEGS

7.1.1 Ad hoc Test

Ad hoc (also known as Ad Lib test) approach is simply play with the functioning unit, trying whatever comes to the mind, in attempt to make it fail. This type of testing approach is the fast and efficient way in debugging in the early development stage.

The disadvantage of this method is many errors or bugs will be found through this method until we never be sure what should or should not be tested.

7.1.2 Black Box Test

Black box test, also known as a functional testing approach, is focus on the functionality of code without concern on its logic structure. It verifies how well a modular code meets its requirement. Defect of code was uncovered by feeding several of inputs and take note on produced output.

The advantage of this kind of testing is that a black box is free of the constraints imposed by the internal structure and logic of the test object. However, the disadvantage is that it is not always possible to run a complete test in this manner due to the selection of an effective input.

Basically, black box testing is a test on boundary value analysis, error guessing and domain testing. Hence, it is a testing to specification.

The figure below shows a sample of my testing on Lexical Analysis module. A text of essay was key-in into the textbox and then the button of "Lexical Analysis" is pressed to perform the functions within it. Immediately, a result in a table view was displayed to show that the operation had been executed.

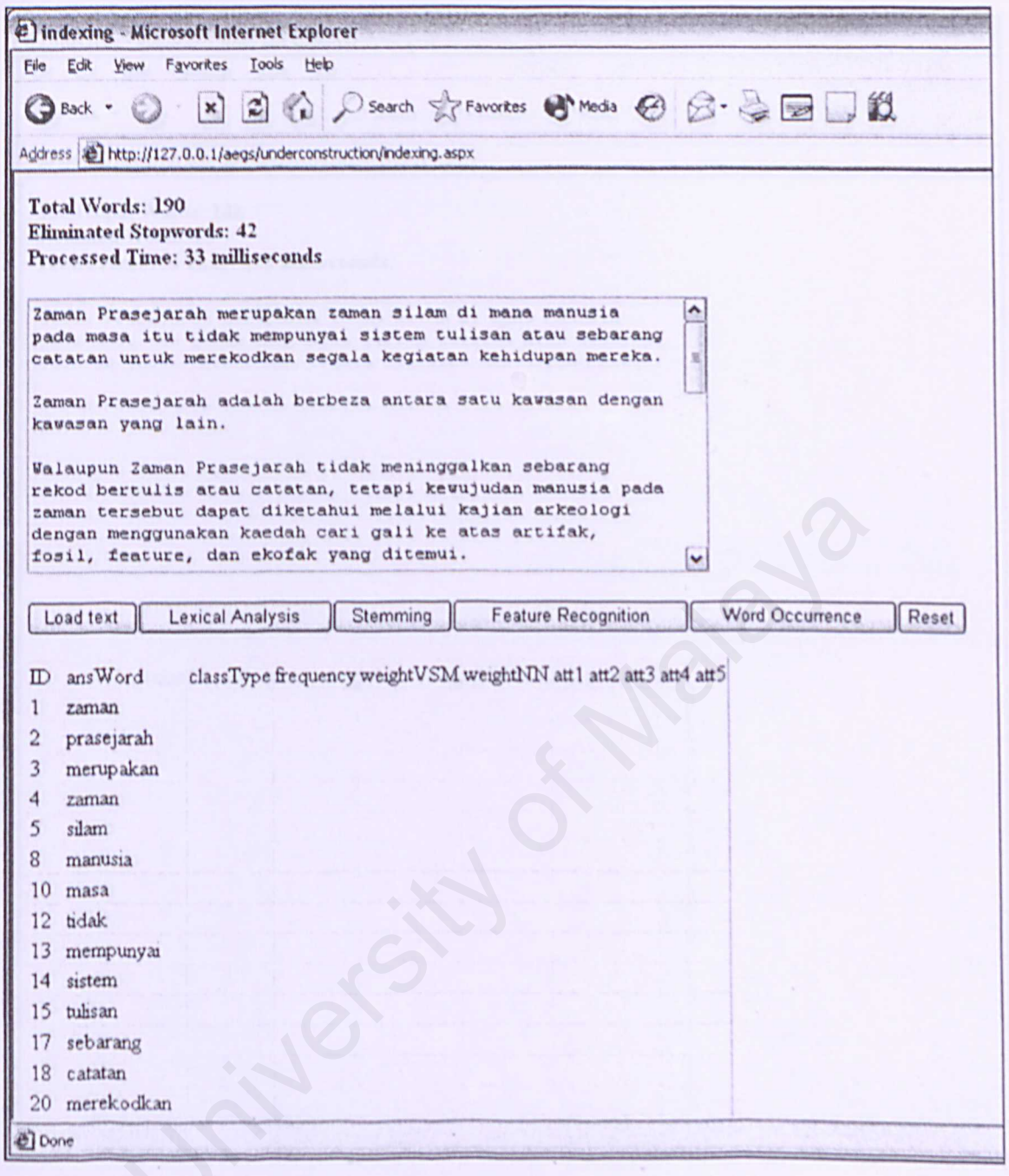


Figure 7.3: Lexical Analysis Unit Testing

As we could see from the figure above, we know how many words had been input and how many stopwords had been eliminated from the original text. Next, we proceed to the white box test.

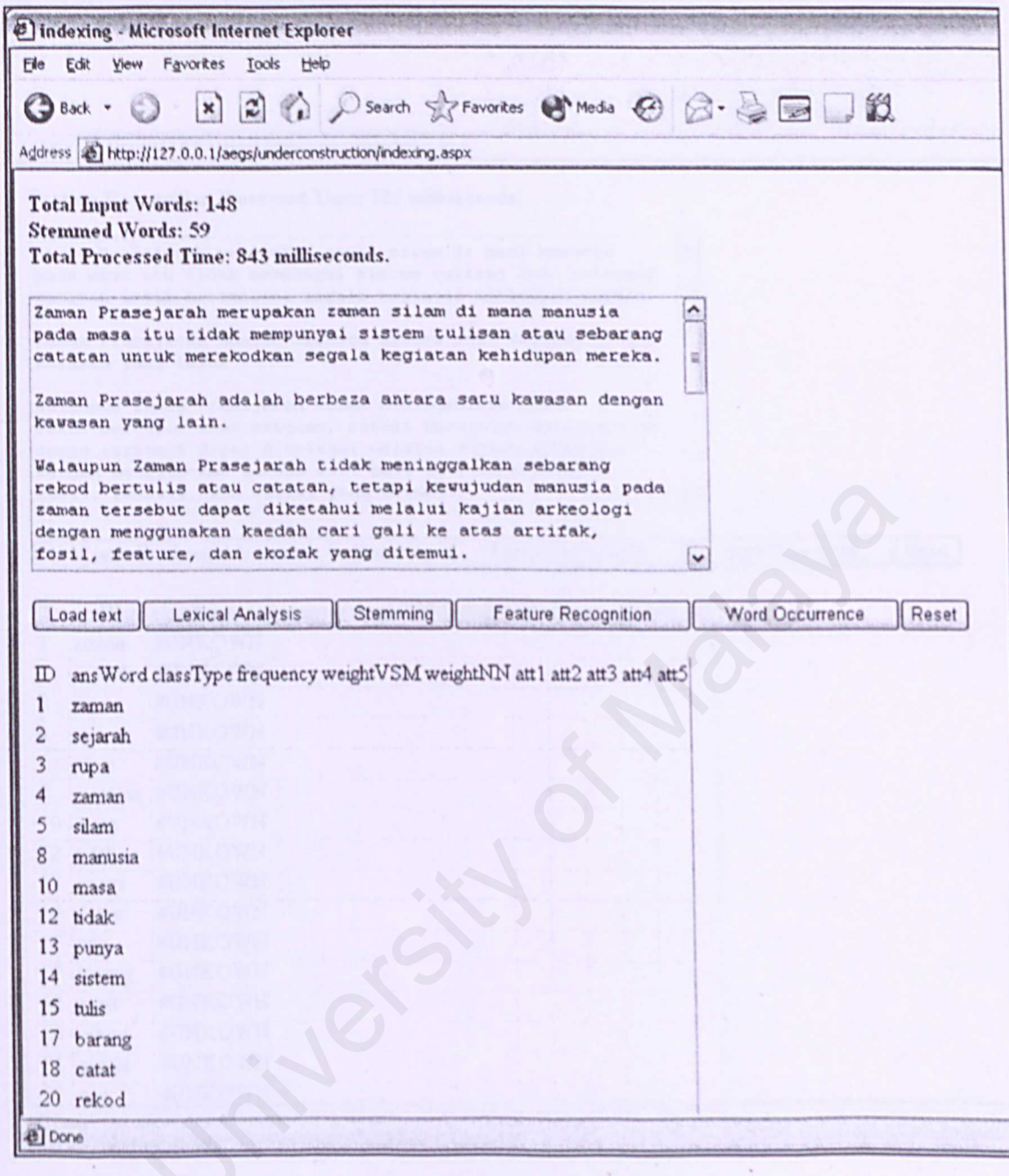


Figure 7.4: Stemming Unit Testing

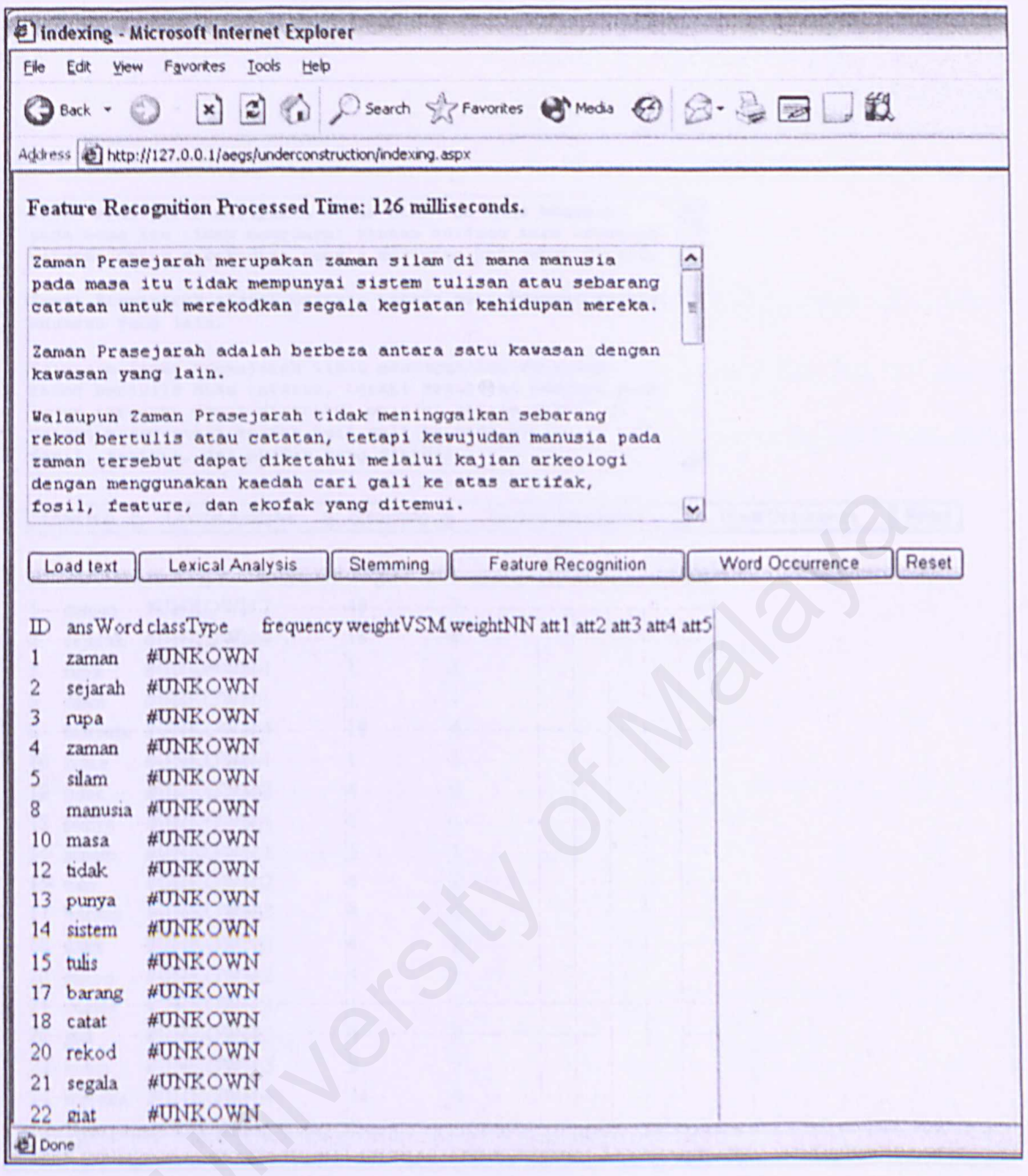


Figure 7.5: Feature Recognition Unit Testing

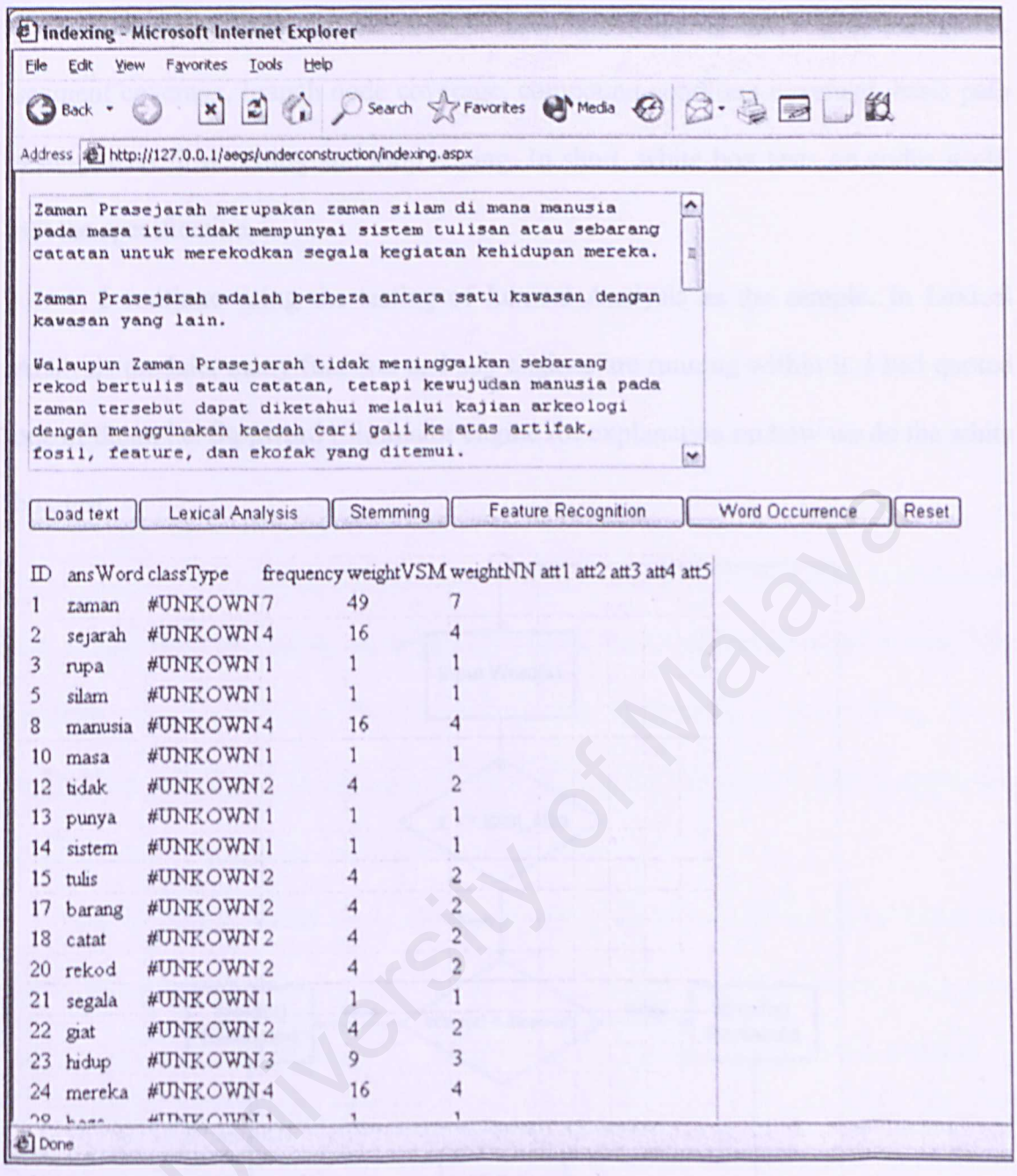


Figure 7.6: Word Occurrence Counter Unit Testing

7.1.3 White Box Test

White box test, also known as structural or logical-driven test, is the type of testing that deals directly with the stricture of the code within a module or a code segment.

The advantage of white box testing is that an analysis of the code can be used to find out how many test cases are needed to guarantee a given level of test coverage. In short, the white box testing methodology is focused in term of coverage.

Basically, there are six type of code coverage in white box testing, i.e. segment coverage, branch node coverage, compound condition coverage, basis path testing, data flow testing and loop testing. In short, white box tests on codes itself, not the specification.

I continue using the testing of Lexical Analysis as the sample. In Lexical Analysis module, many function and sub engines are running within it. I had quoted one of them, i.e. Stopword Eliminator engine for explanation on how we do the white box test.

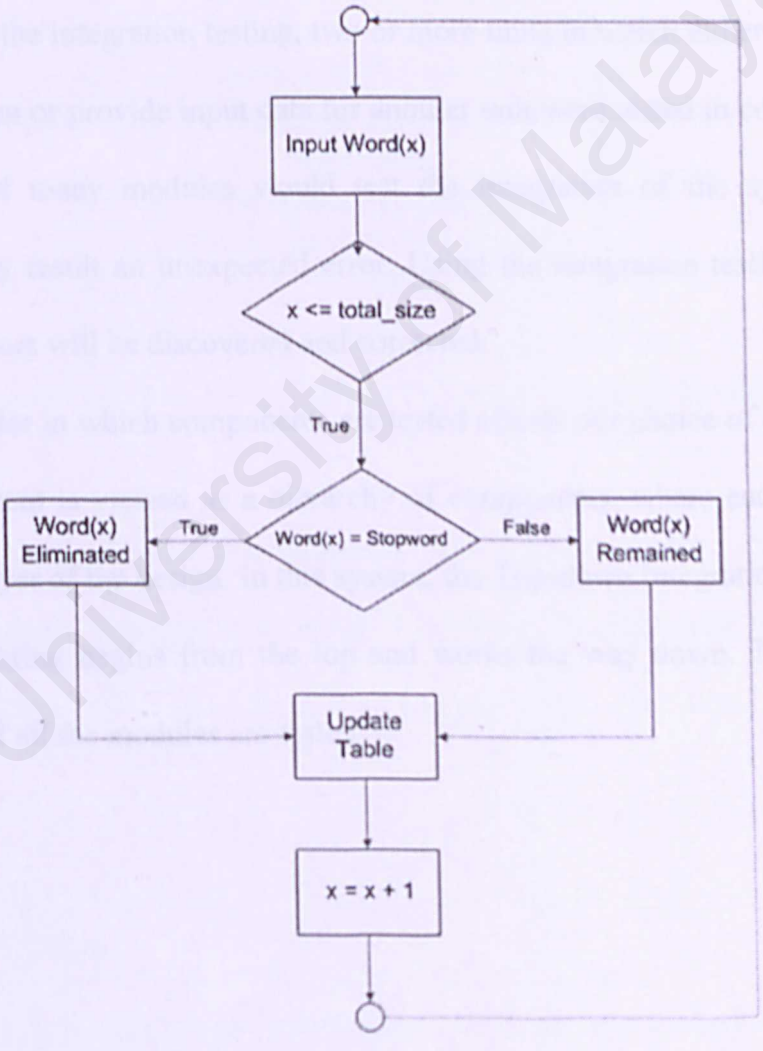


Figure 7.7: Data Flow of Stopword Eliminator Engine

This Stopword Eliminator engine was tested by input bigger values of x until it exceeded its boundary condition to see whether it will still proceed to check with

the stopword list table. Apart from that, I changed some stopwords of the list so that the decision branches were both validated through the test.

7.2 Integration Testing

After all the unit tests, the next step is to verify that the interface between the components are well-defined and handled properly. Through this test, the component of modules of AEGS was tested to be well-functional when they were integrated together.

During the integration testing, two or more units in which either unit that use output data from or provide input data for another unit were tested in collection. The combination of many modules would test the integration of the system. Some integration may result an unexpected error. Using the integration testing approach, this kind of errors will be discovered and corrected.

The order in which components are tested affects our choice of test cases and tools. The system is viewed as a hierarchy of components, where each component belongs to a layer of the design. In this system, the Top-down Integration approach is used where testing begins from the top and works the way down. The process is continued until all the modules are tested.

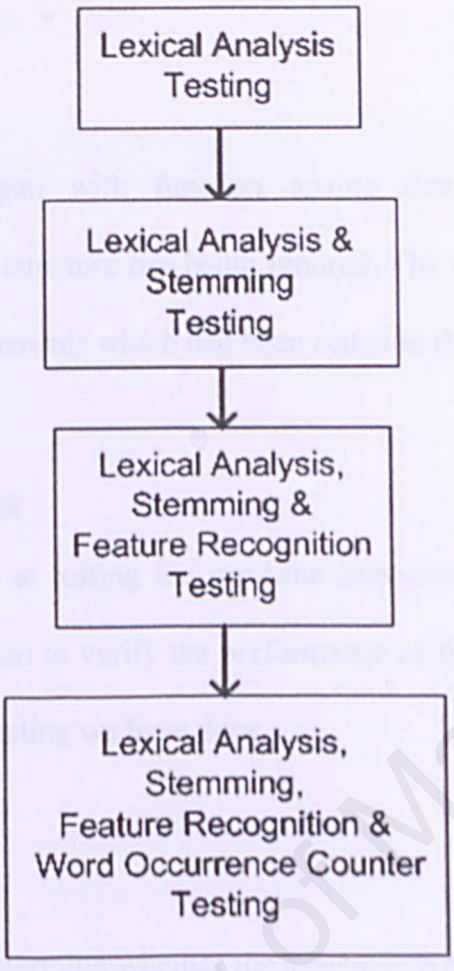


Figure 7.8: Top-Down Testing Process of Indexing Module

Figures above shows my integrated testing process, from Lexical Analysis, Stemming, Feature Recognition to Word Occurrence Counter. Each testing on them was independently. After all of them are integrated, they were tested again with the interface.

7.3 System Testing

System testing is the last but not least testing procedure. It is performed to ensure the entire system meets the user's requirement by uncovering its limitation and capabilities. Normally, users could join the developers in this stage of testing.

Several steps were taken in testing our AEGS, i.e. functions testing and performance testing.

7.3.1 Functions Testing

System testing was begun with function testing that focus completely on functionality. The system structure has being ignored. The testing was based on the system's functional requirements which had been stated in the early chapter.

7.3.2 Performance Testing

Performance testing aims at testing the run-time execution. Response time of the event triggered was checked to verify the performance of the system too. Below are some descriptions of the testing we have done.

7.3.2.1 Stress Testing

The main purpose is to determine whether the system can handle, as it should, large and varies workload at one time. It subject system to high loads over a short period of time.

7.3.2.2 Human Factor Testing

In this testing, interface and message are being evaluated by many user to get the best interaction effects. It was concentrated at the appearance and the interaction of the system. All aspect that may be related to ease of use, such as display screen, has been examined.

7.3.2.3 Response Time Testing

A few sample of document with various numbers of words had been chosen to run the response time testing. The data had been taken down as in the table below.

Table 7.1: Response Time VS Words

Total Words	Response Time (ms)				
	Lexical Analysis	Stemming	Feature Recognition	Word Occurrence	Total
123	0	530	60	0	590
204	0	890	173	0	1063
253	0	1170	216	0	1386
260	0	1220	220	0	1440
344	0	1720	373	0	2093
418	0	2173	513	0	2686
659	0	3830	1140	0	4970

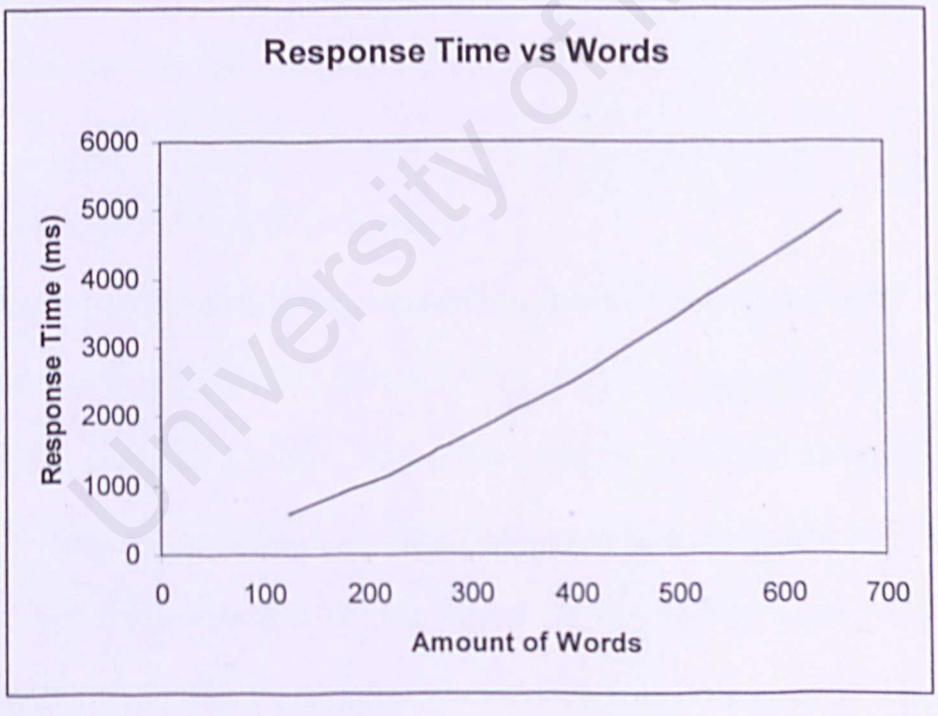


Figure 7.9: Graph of Response Time VS Amount of Words

A graph with response time VS amount of words had been plotted. Clearly, that is a linear graph. The slope of the graph is Response Time / Amount of Words,

which denoted as our Relative Response Time, RRT = 8.2895ms. Consequently, smaller RRT value gives faster system performance.

7.4 Chapter Summary

Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Unit, integration and system testing has been carried out for AEGS. Consequently, all functions beyond AEGS would likely to be performed with error-free. The following chapter would deals the system evaluation on Indexing.

Chapter 8: System Evaluation and Conclusion

System evaluation is the ultimate phase of developing a system and an important phase before delivery the system to the end user. Evaluation phase was to determine the extent to which the system the expected outcomes have been realized, and the prescriptive value of the process where extraneous factors were taken consideration. In fact, my system evaluation was emphasized on my stemming algorithm, which has played an important part in my Indexing module of AEGS. Lastly, a conclusion would be made according to my system strength and limitation.

8.1 System Evaluation

This section consists of my evaluation of stemming algorithm, system strength, system limitation and future enhancement.

8.1.1 Evaluation of Stemming Algorithm

My designed stemmer is a linguistically-motivated (dictionary-based) stemmer, instead of a purely rule-based stemmer. This stemming algorithm was developed using affixes stripping approach with a dictionary look-up. The dictionary is very simple; it consists of a list of vocabulary. The stemming process is done by stripping the shortest possible match of affixes. The dictionary look-up is performed before each stripping step and the stripping process itself is implemented recursively.

Ideally, a good stemmer will all word from the same semantic group to the same stem. But due to the irregularities in all human natural languages, a stemmer is unavoidable to make mistake, though it is provided by a list of vocabulary (dictionary). In other words, we could say that no stemmers could be functioned perfectly.

There are always have existed error cases where some words are merged to different stem while they are from the same conceptual group (understemming); or the words are merged to the same stem while their conceptual group are different (overstemming). Hence, a good stemmer should produce as few understemming or overstemming as possible.

8.1.1.1 Overstemming Errors

Consider the eight words shown in the second column of the table below, which fall into two distinct conceptual groups, and the third column is their stemmed words. We will look at the effect of applying stemming algorithm to these words.

Table 8.1: Overstemming Word Pairs with Stemmed Words

Group	Full Words	Stemmed Words
1	seribu (a thousand)	ribu
	menyeribu (thousand days after death)	ribu
	beribu (to have thousand of)	ibu
	meribu (thousand of)	ribu
	ribuan (thousand of)	ribu
2	beribu (to have a mother)	ibu
	beribukan (with a mother)	ibu
	keibuan (femina)	ibu

Coming up next will show how overstemming indices (OI) and understemming indices (UI) can be computed. In principle, the idea is to compare every pair of words in the sample -- though if fact, to avoid wasting lots of time, only

certain pairs are actually considered. For each comparison, we will know whether the pair of words belongs to the same conceptual group, and whether they were in fact converted to the same stem.

Table 8.2: Indication of Overstemming Word Pairs

	seribu	menyeribu	beribu	meribu	ribuan	beribu	beribukan	keibuan
seribu		X	X	X	X	1	1	1
menyeribu	X		X	X	X	1	1	1
beribu	X	X		X	X	0	0	0
meribu	X	X	X		X	1	1	1
ribuan	X	X	X	X		1	1	1
beribu	1	1	0	1	1		X	X
beribukan	1	1	0	1	1	X		X
keibuan	1	1	0	1	1	X	X	

1 = A pair of non-groupable words with non-identical stems. (successful stemming).
0 = A pair of non-groupable words with identical stems. (Overstemming error).
X = A pair of groupable words that not considered when counting Overstemming errors.

In this case, the proportion of word pairs which were correctly not merged to the same stem is 24 out of 30, hence OI = 1 - (24/30), OI = 0.20.

8.1.1.2 Understemming Errors

After that, the words from group 1 are taken for undergoing understemming error analysis.

Table 8.3: Indication of Understemming Word Pairs

	seribu	menyeribu	beribu	meribu	ribuan	beribu	beribukan	keibuan
seribu		1	0	1	1	X	X	X
menyeribu	1		0	1	1	X	X	X
beribu	0	0		0	0	X	X	X
meribu	1	1	0		1	X	X	X
ribuan	1	1	0	1		X	X	X
beribu	X	X	X	X	X		1	1
beribukan	X	X	X	X	X	1		1
keibuan	X	X	X	X	X	1	1	

1 = A pair of groupable words with identical stems (successful stemming).
0 = A pair of groupable words with non-identical stems (Understemming error).
X = A pair of non-groupable words that not considered when counting Understemming errors.

In this case, the proportion of word pairs successfully merged is 18 out of 28, hence UI= 1 - (18/28) = 0.357.

8.1.1.3 Discussion

Actually, there are some problems with this approach. Firstly, the manual construction of the grouped word collection is time-consuming, which limits the size of the word collections which can be used. Secondly, it is sometimes unclear whether or not two words should be grouped together, for instance, whether *berperang* (to

have war / brown coloured) and *peperangan* (war) should be merged depends very much on the topic of the original text. Moreover, the question of whether or not a group will contain all of the words it should and none of the words that it should not is highly relative to the document being inspected.

The OI and UI actually indicate the level of overstemming and understemming errors respectively. The smaller values show the fewer errors of overstemming or understemming.

8.1.2 Evaluation of Stopword List

The stopword list was produced after doing the word frequency analysis. Word frequency analysis was conducted by performing experiment on Malay language news articles. This experiment used online Malaysian newspapers as text source. One week editions are collected from Berita Harian Online, (*http://www.bharian.com.my*, 6/3/2005), one of the most widely read newspaper in Malaysia.

These editions are taken in a consecutive of every day in a week (started from 28 February 2005 until 6 March 2005) with the total of 100 documents. These articles are only the daily local news of the newspaper.

This articles which was used in this experiment consists of 4553 unique words, after removing the name of people, organization, cities and countries. I then attach only the top 500 of the list of frequently used in Malay news articles (Appendix C). From these top 500 frequently used words, I suggest a list of stopwords (Appendix D) which was inherited from the words frequency analysis list.

8.1.3: Encountered Problems and Solutions

Problems encountering is a normal thing when developing a system. Troubleshooting is then carrying out to overcome the obstacles. The following are the major problems encountered from the beginning of the project through the end of the system development process.

8.1.3.1: Difficulties in Designing User Interface

Problems that faced during the early stage of development are lack of knowledge and experience of the real system flow and layout of standard user interface. Therefore, it is difficult in designing the most appropriate logic and user interface. By referring to many web-based systems interface, it did help us to come out a design in a more presentable and attractive style yet professional look.

8.1.3.2: Timer Architecture

Problem that faced during designing was the time constraint of essay test for students. We at first thought of putting a stop-watch as time keeper but we then realized that when the line was dropped and how could we continue the last remained time? Hence, we changed the design to a better solution that read from the server time by setting a period of test by particular teachers.

8.1.4: System Strengths

Absolutely, AEGS has some powerful features that considered as its strength, comparing to other similar existing system. All the strength that analyzed by us were well-described as below.

8.1.4.1: In Specific Term

a) Efficient Response Time

The relative response time of the indexing procedures in AEGS is short. In our benchmark testing on a total of seven documents with different words amount, we get as low as 8.2895ms. In another words, the system spends only 0.0083 second on a single word.

8.1.4.2: In General Term

a) Users Authentication

Generally, the security feature in the login procedure of the system ensures only authenticated users to perform their operation legally to this system. Besides, the password configuration that consists of combination of numbers and letters and must has at least six characters in total.

b) Online Test / Exercise

This AEGS provides online essay test of SPM History subject. It is a totally web-based examination which could be accessed through the internet at anywhere in the world. Besides, student could take it as online exercise (configurable by teacher). They could easily access to their exercise questions at home or school.

c) Real-Time Grading

Apart from the web-based feature stated above, the system is also a real-time grader. Student could get to know their results in a short moment after they submit their answers. This has overcome the time-consuming problem facing by a human grader.

d) Export Results

The results obtained could be export and store in Microsoft Excel spreadsheet format. By this method, a teacher could be manipulate his/her student's results easily and perform any charting analysis on the results data.

8.1.5: System Limitation

Despite of the strengths, a system sure has some limitation and so does AEGS. Below are the weakness encountered when evaluating this system.

8.1.5.1: In Specific Term

a) Aspect of Lexical Analysis

The operation is only performed on a word, instead of a phrase or a sentence. So, it couldn't recognize "*Sultan Iskandar Syah*" (name of a ruler of state) as a unit of entity yet. However, it separated it into "*Sultan*", "*Iskandar*", "*Syah*", three words. Meanwhile, the hyphen removal algorithm which has been using will truncate some of the phrase, e.g. "*al-Quran*" became "*al*"; "*soal-selidik*" (questionnaire) became "*soal*" (to ask).

8.1.5.2: In General Term

b) Unable to Recognize Handwritten

This AEGS is designed for online test/exercise that accept computer-based text essay, instead of handwritten query. So, a user must have a computer with a standard keyboard and mouse attached when using this system.

8.1.6: Future Enhancement

Due to the limitation of this system, there are a few suggestions that may be useful for future enhancement of the AEGS. The suggestions are as below:

8.1.6.1: In Specific Term

a) Aspect of Lexical Analysis

The Lexical Analysis process should be upgraded to phrase-level or sentence-level, so that more linguistically meaningful words could be obtained. This could ease the feature recognition process as well.

b) Aspect of Stemming

Since every stemmer has its own overstemming and understemming errors, the linguistically-motivated method which is now using should be upgraded into rule-based stemmer, which is get rid of using list of vocabulary (dictionary), in order to reduce the text parsing burden in current algorithm. This may need supportive research in natural language processing area.

8.1.6.2: In General Term

a) Handwritten Recognition

Handwritten recognition would be a better choice since most of the public examination is still carrying on traditional style in Malaysia. Research and development in this kind of area should be invested so that a recognition device could be came out and hence integrated with AEGS to perform a higher level of query input process.

8.2: System Conclusion

In conclusion, an automated essay grading system has been created to assess the writing essays of SPM History subject. This should be good news for teacher as it will help to reduce the time of preparing test question and their grading process. Much time is saved by fully utilize this system. Besides, they could easily organize their students' results.

Secondly, for indexing part of this system, a less-error stemmer for Malay language has been created too. It has enhanced the other indexing process which consists of lexical analysis, feature recognition and word occurrence counter.

Clearly, the objectives that had been declared in this early stage of project (proposal) are achieved.

8.3: Chapter Summary

This is the last chapter in this project. It evaluates the system and analysis the stemming process. Also, the strength and limitation of the system had been discovered in this chapter. Developing future enhancement is suggested in this chapter as well. A system conclusion on whether the objectives were achieved ends this chapter. Lastly, you may proceed to read the grading part of this AEGS in my project partner's report.

Appendix A

Table A.1: ASP.NET Object Definition Standard

ASP.NET Server Input Controls	Instances
Button	**btn**ObjectName
Calendar	**cal**ObjectName
Check Box	**chk**ObjectName
Check Box List	**chl**ObjectName
Data Grid	**dtg**ObjectName
Data List	**dtl**ObjectName
Drop Down List	**lst**ObjectName
Hyper Link	**lnk**ObjectName
Image Button	**but**ObjectName
Image	**img**ObjectName
Label	**lbl**ObjectName
Link Button	**lnk**ObjectName
Panel	**pnl**ObjectName
Radio Button	**rdb**ObjectName
Radio Button List	**rdl**ObjectName
Repeater	**rpt**ObjectName
Table	**tbl**ObjectName
Table Row	**row**ObjectName
Table Cell	**cel**ObjectName
Text Box	**txt**ObjectName

A.1 Stored Procedure Standard

1. Stored procedure name:
   usp_Type_StoredProcName
   e.g. usp_Mst_Student
        usp_Stud_Result
        usp_Teacher_Soalan

2. Function name:
   func_FunctionName
   e.g. func_GetDocketNum

3. SQL statements:

   if exists (Condition)
   BEGIN
   <4 spaces>SELECT Attribute1
   <4 spaces>FROM TableName
   <4 spaces>WHERE Condition
   END

   .:. use 4 spaces, instead of tab, for each inner statements.

A.2 SQL Server Standard

1. Table name

   a) Master table:
      Mst_TableName
      e.g. Mst_Customer

   b) View table:
      View_TableName
      e.g. View_Warehouse

   c) Module table:
      Module_TableName
      e.g. Stud_Result
           Teacher_Soalan

2. **FndMaster**
   - for dropdownlist control

3. For each created Table, please add **10 attributes** for future maintenance.

Appendix B

B.1 Codes of Stemming Algorithm

```
ALTER      proc usp_INDEXING_Stemming
    @tableName varChar(30)
as
SET NOCOUNT ON

DECLARE @SQL varChar(8000)
set @SQL = '
DECLARE @i int
DECLARE @strWord varChar(50)
DECLARE @lstLetter char
DECLARE @boolDict int
DECLARE @boolLocalDict int
DECLARE @prefix varChar(10)
DECLARE @class varChar(50)
DECLARE @timeBegin datetime
DECLARE @timeEnd datetime
SET @timeBegin = ( select getdate() )
    SET @i = 1
    WHILE @i <= ( SELECT MAX(ID) FROM '+ @tableName +' )
    BEGIN
        SET @strWord = ( SELECT ansWord FROM '+ @tableName +' WHERE
ID = @i )

/*        if ( @boolLocalDict = 1 )
        begin
            set @class = ( SELECT classType
                           FROM dbo.List_Of_Local_Dictionary
                           WHERE localWord = @strWord
                         )
            UPDATE '+ @tableName +'
            SET classType = @class
            WHERE ID = @i
        end*/

prefix: set @boolDict = dbo.func_Check_Dictionary(@strWord)
        set @boolLocalDict =
dbo.func_Check_Local_Dictionary(@strWord)
        if ( @boolDict = 0 AND @boolLocalDict = 0 )
        BEGIN
            set @strWord = dbo.func_Check_Prefix(@strWord)
            set @prefix = left(@strWord, (charindex(''-
'',@strWord)-1))
            set @lstLetter = SUBSTRING(@strWord, CHARINDEX(''-'',
@strWord)+1, 1)
            if ( @prefix = ''men'' OR @prefix = ''pen'' )
            begin
                if ( @lstLetter LIKE ''[aeiou]'' )
                begin
                    set @strWord = substring(@strWord,
charindex(''-'', @strWord)+1, len(@strWord)-charindex(''-'',
@strWord) )
                    set @strWord = ''t'' + @strWord
                end
            end
            else if ( @prefix = ''meng'' OR @prefix = ''peng'' )
            begin
```

```
                if ( @lstLetter LIKE ''[aeiou]'' )
                begin
                    set @strWord = substring(@strWord,
charindex(''-'', @strWord)+1, len(@strWord)-charindex(''-'',
@strWord) )
                    set @strWord = ''k'' + @strWord
                end
            end
            else if ( @prefix = ''meny'' OR @prefix = ''peny'' )
            begin
                if ( @lstLetter LIKE ''[aeiou]'' )
                begin
                    set @strWord = substring(@strWord,
charindex(''-'', @strWord)+1, len(@strWord)-charindex(''-'',
@strWord) )
                    set @strWord = ''s'' + @strWord
                end
            end
            else if ( @prefix = ''mem'' OR @prefix = ''pem'' )
            begin
                if ( @lstLetter LIKE ''[aeiou]'' )
                begin
                    set @strWord = substring(@strWord,
charindex(''-'', @strWord)+1, len(@strWord)-charindex(''-'',
@strWord) )
                    set @strWord = ''f'' + @strWord
                    set @strWord = dbo.func_Check_Suffix(@strWord)
                    set @boolDict =
dbo.func_Check_Dictionary(@strWord)
                    if ( @boolDict = 0 )
                    begin
                        set @strWord = ''p'' + substring(@strWord,
2, len(@strWord))
                        set @strWord =
dbo.func_Check_Suffix(@strWord)
                        set @boolDict =
dbo.func_Check_Dictionary(@strWord)
                    end
                end
            end
            set @strWord = substring(@strWord, charindex(''-'',
@strWord)+1, len(@strWord)-charindex(''-'', @strWord) )
            set @boolDict = dbo.func_Check_Dictionary(@strWord)
            if @boolDict = 0
            begin
                set @strWord = dbo.func_Check_Suffix(@strWord)
            end
            if ( substring(@strWord, 1, 3) = ''per'' )
            begin
                goto prefix
            end
            else if ( substring(@strWord, 1, 3) = ''ter'' )
                goto prefix
            else if ( substring(@strWord, 1, 2) = ''se'' )
                goto prefix
        END
        UPDATE '+ @tableName +'
        SET ansWord = @strWord
        WHERE ID = @i
        SET @i = @i + 1
    END
```

```
SET @timeEnd =(select getdate())
DECLARE @timeUsed varChar(30)
SET @timeUsed = datediff(millisecond, @timeBegin, @timeEnd)
SELECT @timeUsed timeUsed
'

--print (@SQL)
EXEC (@SQL)


SET NOCOUNT OFF
```

Appendix C

Table C.1: Word Frequency Analysis

No.	Word	Frequency	No.	Word	Frequency
1	dan	682	41	boleh	69
2	di	613	42	sebelum	67
3	yang	606	43	seorang	67
4	itu	509	44	sebagai	66
5	berkata	264	45	dua	65
6	katanya	237	46	tanpa	63
7	untuk	202	47	kawasan	60
8	tidak	194	48	operasi	59
9	pada	180	49	terbabit	59
10	ini	177	50	pendatang	58
11	akan	176	51	ada	56
12	dalam	175	52	kejadian	56
13	kepada	175	53	pihak	54
14	dengan	174	54	atau	54
15	ke	145	55	tetapi	52
16	semalam	135	56	lima	52
17	mereka	134	57	rumah	52
18	bagi	129	58	lebih	50
19	sini	113	59	antara	50
20	lalu	111	60	sekolah	50
21	berkenaan	110	61	jabatan	50
22	kerana	109	62	termasuk	49
23	beliau	107	63	dia	49
24	juga	106	64	mahkamah	49
25	negara	102	65	orang	48
26	kerajaan	97	66	hari	48
27	tahun	93	67	memberi	48
28	selepas	92	68	jalan	48
29	dari	92	69	oleh	48
30	ketika	91	70	ditahan	47
31	mangsa	83	71	mengenai	46
32	polis	81	72	program	46
33	sudah	77	73	taman	45
34	serta	76	74	kes	45
35	menteri	74	75	perdana	44
36	jam	73	76	semua	44
37	kita	71	77	izin	44
38	negeri	71	78	beberapa	44
39	daripada	70	79	ketua	44
40	kira	69	80	syarikat	44

No.	Word	Frequency	No.	Word	Frequency
81	perlu	43	121	guru	33
82	masalah	43	122	manakala	33
83	kelmarin	43	123	lain	33
84	saya	42	124	menerima	33
85	ia	42	125	tempoh	32
86	terhadap	42	126	kini	32
87	lelaki	42	127	tinggi	32
88	kementerian	42	128	pagi	32
89	pelajar	42	129	pegawai	32
90	hanya	41	130	ramai	31
91	sehingga	41	131	petang	31
92	sebuah	41	132	kuala	31
93	sementara	41	13	mahu	31
94	industri	41	134	kerja	30
95	turut	40	135	masih	30
96	siasatan	40	136	jika	30
97	laporan	40	137	kesalahan	30
98	anggota	39	138	baru	30
99	adalah	39	139	berusia	30
100	seperti	39	140	sebarang	29
101	sama	39	141	harga	29
102	malam	38	142	menjadi	29
103	mengikut	38	143	warga	29
104	dapat	38	144	bersama	28
105	didakwa	38	145	membawa	28
106	berlaku	37	146	iaitu	28
107	membuat	37	147	ops	28
108	pusat	37	148	satu	28
109	mengambil	36	149	ibu	28
110	tiga	36	150	mendapat	27
111	pembangunan	36	151	membabitkan	27
112	rakyat	35	152	sidang	27
113	secara	35	153	laman	27
114	tempat	35	154	web	27
115	tertuduh	35	155	bekerja	27
116	bagaimanapun	34	156	sekitar	27
117	hingga	34	157	sejak	26
118	pulang	34	158	selain	26
119	mempunyai	34	159	sen	26
120	besar	34	160	media	26

No.	Word	Frequency	No.	Word	Frequency
161	pekerja	26	201	februari	22
162	anak	26	202	menjalankan	22
163	seksyen	26	203	bukan	22
164	bahagian	26	204	membantu	22
165	menggunakan	25	205	keluarga	22
166	bulan	25	206	kampung	22
167	masa	25	207	kecil	22
168	melakukan	25	208	kereta	22
169	pertanian	25	209	pelatih	21
170	menyebabkan	25	210	setiap	21
171	asing	25	211	kebangsaan	21
172	kedua	25	212	perkara	21
173	hukuman	25	213	raja	21
174	universiti	25	214	wanita	21
175	keadaan	24	215	alam	21
176	lagi	24	216	dipercayai	21
177	berikutan	24	217	hakim	21
178	subsidi	24	218	semula	21
179	enam	24	219	pengarah	21
180	majlis	24	220	hutan	21
181	pihaknya	24	221	raya	20
182	tiada	24	222	walaupun	20
183	penjara	24	223	terpaksa	20
184	latihan	24	224	bot	20
185	diri	24	225	tengah	20
186	awam	23	226	tindakan	20
187	tegas	23	227	meminta	20
188	rela	23	228	kakitangan	20
189	kraf	23	229	akibat	20
190	supaya	23	230	akhbar	20
191	banyak	23	231	sumbangan	20
192	apabila	23	232	kumpulan	20
193	dilakukan	23	233	tiba	19
194	timbalan	23	234	ahli	19
195	mendapati	23	235	bermula	19
196	air	23	236	melalui	19
197	dibuat	22	237	diberikan	19
198	terus	22	238	pejabat	19
199	memastikan	22	239	maklumat	19
200	diesel	22	240	hospital	19

No.	Word	Frequency	No.	Word	Frequency
241	bukit	19	281	terutama	16
242	wang	19	282	pasangan	16
243	kursus	19	283	berita	16
244	bantuan	19	284	langkah	16
245	ubat	19	285	sumber	16
246	tanah	19	286	pengusaha	16
247	dibawa	18	287	langkawi	16
248	kabinet	18	288	pasti	16
249	keputusan	18	289	pembinaan	16
250	atas	18	290	membayar	16
251	pelbagai	18	291	lapan	15
252	harian	18	292	keterangan	15
253	rayuan	18	293	gaji	15
254	empat	18	294	pengangkutan	15
255	pulau	18	295	seluruh	15
256	mati	18	296	mampu	15
257	projek	18	297	seliter	15
258	mendapatkan	18	298	menyediakan	15
259	koperasi	17	299	kenderaan	15
260	perkhidmatan	17	300	menahan	15
261	sektor	17	301	enggan	15
262	kuasa	17	302	demikian	15
263	stesen	17	303	lanjut	15
264	perdagangan	17	304	mesyuarat	15
265	hasil	17	305	tentera	15
266	masing	17	306	bandar	15
267	daerah	17	307	pinjaman	15
268	gagal	17	308	sekali	14
269	kami	17	309	menghadapi	14
270	kos	17	310	sepatutnya	14
271	imigresen	17	311	sambil	14
272	cedera	17	312	sebanyak	14
273	puteri	16	313	bilion	14
274	sabah	16	314	baik	14
275	datang	16	315	januari	14
276	minyak	16	316	sungai	14
277	usaha	16	317	lucah	14
278	oktober	16	318	memberikan	14
279	berada	16	319	sultan	14
280	menerusi	16	320	barangan	14

No.	Word	Frequency	No.	Word	Frequency
321	muda	14	361	pertama	12
322	saja	14	362	pemandu	12
323	kebakaran	14	363	perjalanan	12
324	isu	14	364	tamat	12
325	kanak	14	365	kalangan	12
326	dekat	14	366	depan	12
327	makanan	14	367	cuba	12
328	bayaran	14	368	aduan	12
329	akta	14	369	bas	12
330	unit	13	370	dijalankan	12
331	sembilan	13	371	pengampunan	12
332	kota	13	372	berhubung	12
333	sistem	13	373	mengeluarkan	12
334	kenaikan	13	374	mana	12
335	ialah	13	375	melayu	12
336	jumlah	13	376	serbuan	12
337	kemudian	13	377	digunakan	12
338	mungkin	13	378	malah	12
339	pernah	13	379	membolehkan	12
340	dihubungi	13	380	pula	12
341	penduduk	13	381	bilik	12
342	rawatan	13	382	kembali	12
343	utama	13	383	sempena	12
344	kewangan	13	384	pengerusi	12
345	hotel	13	385	tujuh	12
346	ditemui	13	386	diberi	12
347	diraja	13	387	sebulan	12
348	bekas	13	388	tapak	12
349	selama	13	389	jenayah	12
350	tuduhan	13	390	belum	12
351	kanun	13	391	dikatakan	12
352	keseksaan	13	392	gegaran	12
353	keselamatan	13	393	bapa	12
354	kajian	13	394	dokumen	12
355	bekalan	13	395	khidmat	12
356	belakang	13	396	cadangan	11
357	tarikh	13	397	kapal	11
358	usahawan	13	398	tambang	11
359	pemeriksaan	12	399	suruhanjaya	11
360	pertahanan	12	400	memiliki	11

No.	Word	Frequency	No.	Word	Frequency
401	saksi	11	441	penerbangan	10
402	april	11	442	bergerak	10
403	mendakwa	11	443	gangguan	10
404	serius	11	444	amanah	10
405	mesti	11	445	tunda	10
406	menunjukkan	11	446	peluang	10
407	proses	11	447	dadah	10
408	tetap	11	448	berkuasa	10
409	beban	11	449	superintendan	10
410	kali	11	450	berakhir	10
411	liter	11	451	kata	10
412	juta	11	452	mengkaji	10
413	disember	11	453	terbaik	10
414	tenaga	11	454	mengurangkan	10
415	sabtu	11	45	hal	10
416	rahman	11	456	cukai	10
417	tempatan	11	457	penguat	10
418	tertentu	11	458	mengadakan	10
419	lokasi	11	459	bencana	10
420	maksimum	11	460	tsunami	10
421	menghubungi	11	461	surat	10
422	undang	11	462	berharap	10
423	kilang	11	463	bom	10
424	tugas	11	464	berkahwin	10
425	melawat	11	465	jawatankuasa	10
426	keluar	11	466	bendera	10
427	diadakan	11	467	pengunjung	10
428	antarabangsa	11	468	memang	10
429	jumaat	11	469	peguam	10
430	dilancarkan	11	470	induk	10
431	awal	11	471	tinggal	10
432	berjalan	11	472	sebelah	10
433	dijangka	11	473	jaya	10
434	lori	11	474	kenyataan	10
435	pokok	11	475	kraftangan	10
436	masuk	11	476	asas	10
437	pakaian	11	477	kegiatan	10
438	tabung	11	478	dikenali	10
439	luar	11	479	menyerahkan	10
440	hektar	11	480	klang	10

No.	Word	Frequency	No.	Word	Frequency
481	kesihatan	10	491	sah	10
482	sendiri	10	492	berjaya	10
483	aspek	10	493	peruntukan	9
484	dewan	10	494	menjadikan	9
485	parah	10	495	khas	9
486	suspek	10	496	menemui	9
487	diserahkan	10	497	zon	9
488	laut	10	498	pemaju	9
489	keretakan	10	499	pilihan	9
490	kemudahan	10	500	lawatan	9

Appendix D

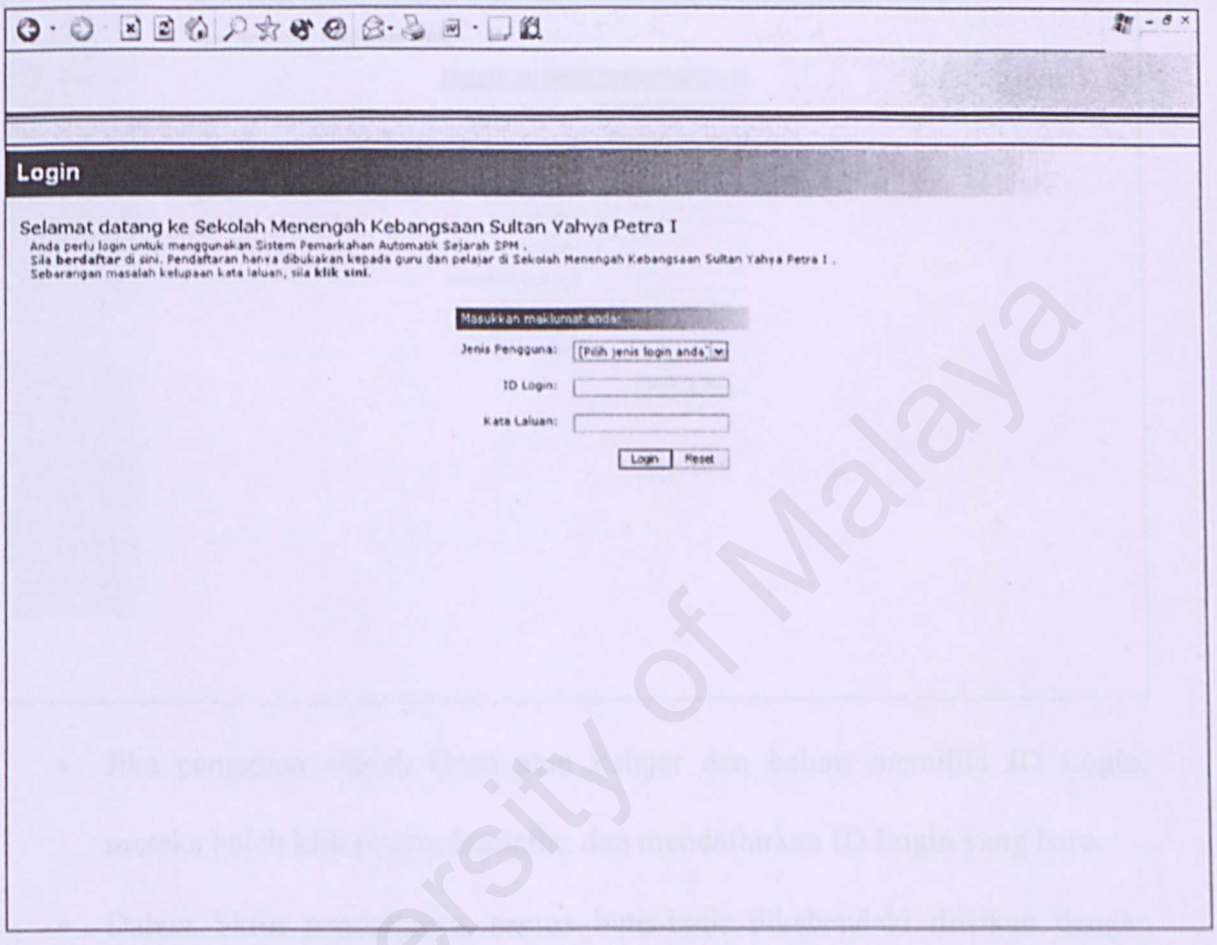
Table D.1: Suggested Stopword List

No.	Word	Root	No.	Word	Root
1	dan	dan	41	terbabit	babit
2	di	di	42	ada	ada
3	yang	yang	43	kejadian	jadi
4	itu	itu	44	pihak	pihak
5	untuk	untuk	45	atau	atau
6	tidak	tidak	46	tetapi	tetapi
7	pada	pada	47	kawasan	kawasan
8	ini	ini	48	operasi	operasi
9	akan	akan	49	lagi	lagi
10	dalam	dalam	50	lebih	lebih
11	kepada	kepada	51	antara	antara
12	dengan	dengan	52	sekolah	sekolah
13	ke	ke	53	jabatan	jabat
14	mereka	mereka	54	termasuk	masuk
15	bagi	bagi	55	dia	dia
16	sini	sini	56	orang	orang
17	lalu	lalu	57	hari	hari
18	berkenaan	kena	58	memberi	beri
19	kerana	kerana	59	jalan	jalan
20	beliau	beliau	60	oleh	oleh
21	juga	juga	61	mengenai	kena
22	tahun	tahun	62	semua	semua
23	selepas	selepas	63	beberapa	berapa
24	dari	dari	64	kelmarin	kelmarin
25	ketika	ketika	65	saya	saya
26	kerajaan	raja	66	ia	ia
27	negara	negara	67	terhadap	hadap
28	sudah	sudah	68	adalah	ada
29	serta	serta	69	seperti	seperti
30	jam	jam	70	sama	sama
31	kita	kita	71	bagaimanapun	bagaimana
32	negeri	negeri	72	hingga	hingga
33	daripada	daripada	73	secara	secara
34	menteri	menteri	74	tempat	tempat
35	boleh	boleh	75	lain	lain
36	sebelum	belum	76	kini	kini
37	seorang	orang	77	masih	masih
38	sebagai	bagai	78	jika	jika
39	tanpa	tanpa	79	iaitu	iaitu
40	kira	kira	80	selain	lain

Appendix E

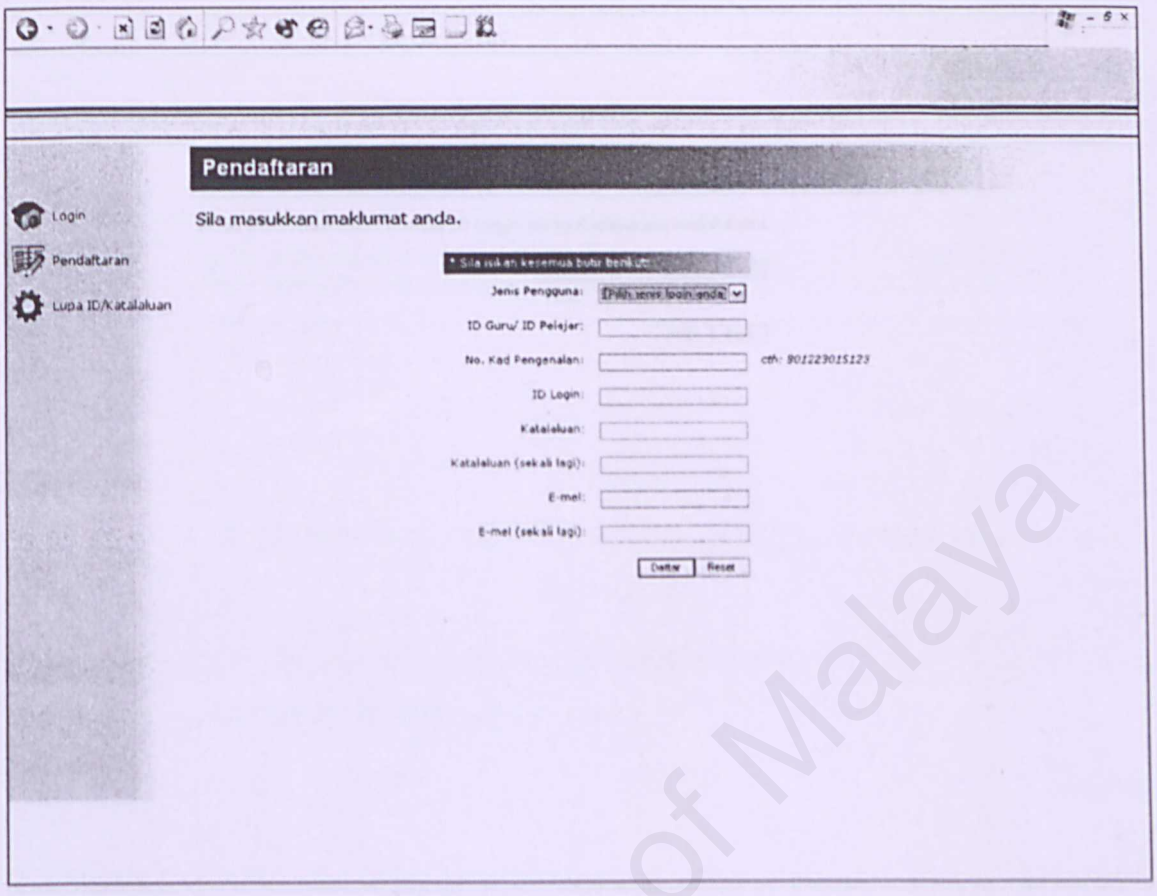
E.1 User Manual (Malay Version)

Pengguna :: Login



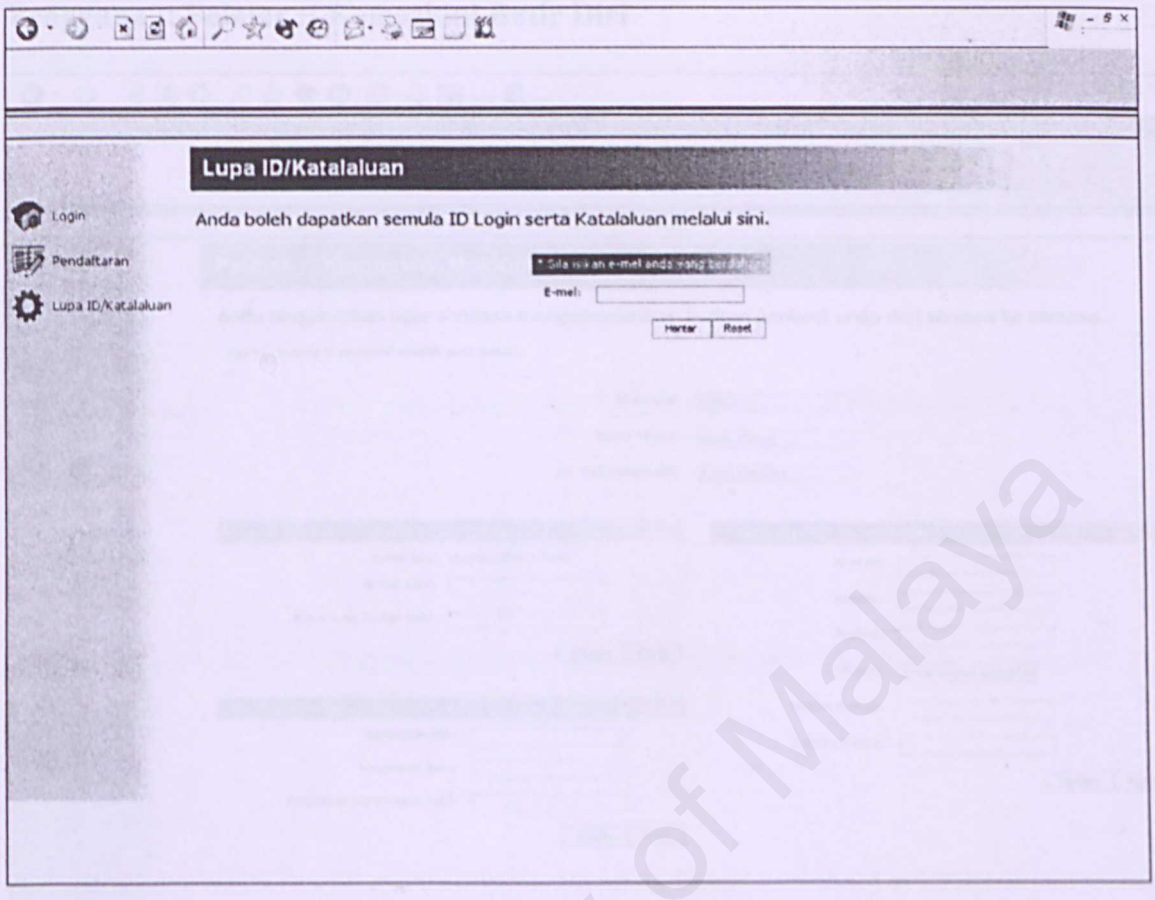
- Pada mukasurat Login, anda dikehendaki memilih jenis login iaitu Guru atau Pelajar. Selepas itu, pasangan ID Login dan katalaluan yang betul hendak dimasukkan pada petak yang telah dilabel.
- Pengguna akan dijemput ke laman utama jika pasangan ID Login dan katalaluan yang betul telah dimasukkan.

Pengguna :: Login :: Pendaftaran



- Jika pengguna adalah Guru atau Pelajar dan belum memiliki ID Login, mereka boleh klik pautan berdaftar dan mendaftarkan ID Login yang baru.
- Dalam Skrin pendaftaran, semua butir-butir dikehendaki diisikan dengan betul. Setelah butir-butir diisikan, pengguna dikehendaki menekan butang 'Daftar' untuk berdaftar.
- Butang 'Reset' adalah untuk mengkosongkan semua petak supaya butiran yang baru dapat diisikan.
- Jika pengguna hendak balik ke skrin Login, pengguna dikehendaki menekan butang Login di sebelah kiri.
- Hanya ID Guru atau Pelajar yang wujud dalam sistem dapat mendaftar. Jika ID anda tidak wujud, sila jumpa system administrator untuk memasukkan ID ke dalam sistem.

Pengguna :: Login :: Lupa ID Login/Katalaluan



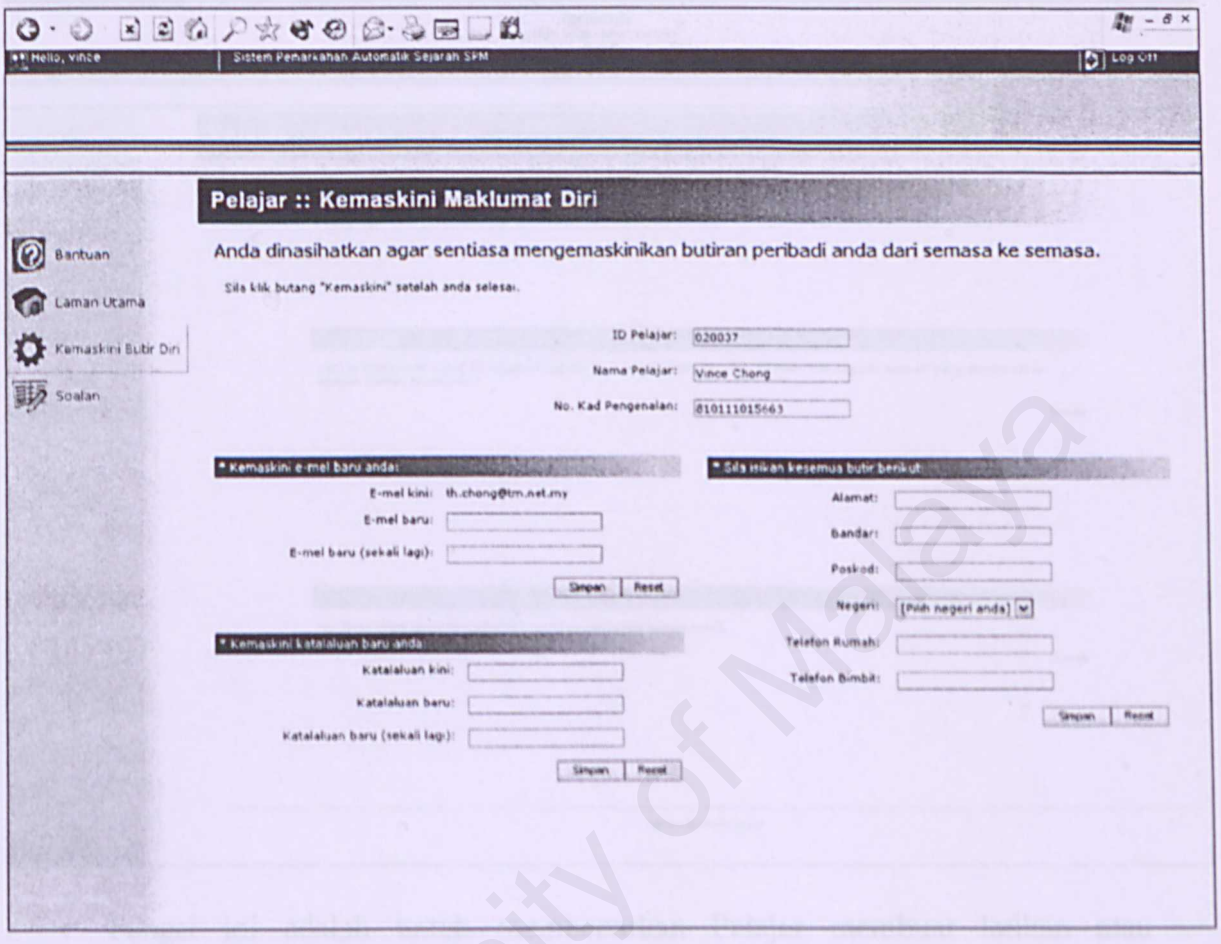
- Jika pengguna lupa ID Login atau katalaluan, pengguna boleh klik pautan *klik sini*. Selepas itu, pengguna dikehendaki memasukkan emel yang telah mereka daftarkan dalam sistem untuk mendapatkan ID Login dan katalaluan mereka. ID Login dan katalaluan akan dihantar ke emel tersebut.
- Emel yang salah tidak dipedulikan.
- Jika pengguna hendak balik ke skrin Login, pengguna dikehendaki menekan butang Login di sebelah kiri.

Pengguna :: Logoff

- Pengguna diingati supaya menekan logoff di sebelah kanan atas setiap kali selepas selesai menggunakan sistem ini.

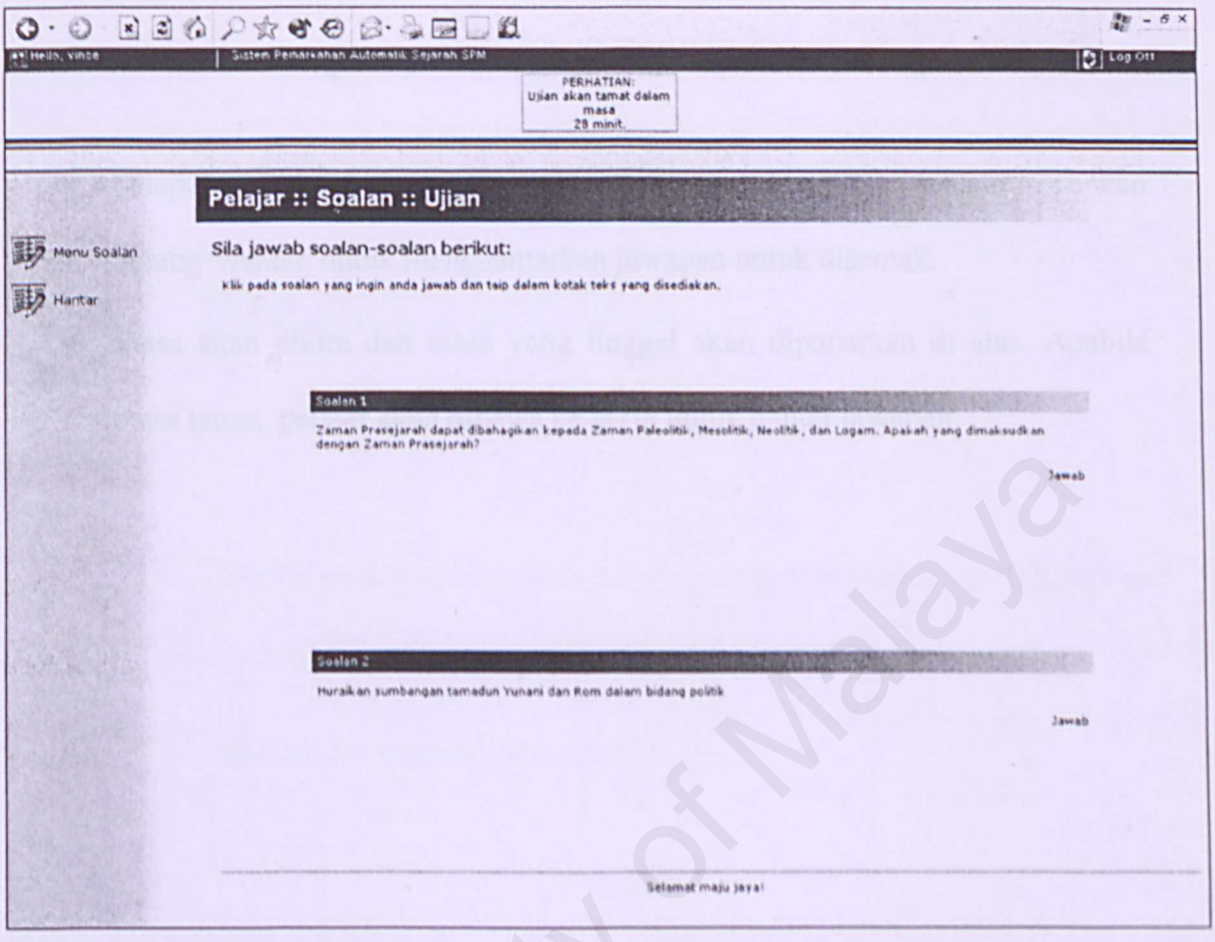
Pengguna :: Pelajar

Pengguna :: Pelajar :: Kemaskini Butir Diri



- Pelajar boleh mengkemaskinikan butir-butir diri dengan fungsi ini. Butir-butir seperti Alamat, Bandar, Poskod, Negeri, Telefon Rumah, Telefon Bimbit, E-mel, dan katalaluan boleh diubah atau dikemaskinikan oleh Pengguna Pelajar
- ID Pelajar, Nama Pelajar dan No. Kad Pengenalan tidak dapat ditukar. Hanya pengguna SysAdmin mempunyai hak untuk menukar butiran-butiran tersebut.

Pengguna :: Pelajar :: Soalan



- Fungsi ini adalah untuk membenarkan Pelajar membuat latihan atau menduduki sesuatu ujian. Selepas memilih fungsi ini, pengguna dikehendaki memilih sama ada pengguna hendak memasuki Latihan atau Ujian.
- Selepas itu, arahan untuk ujian atau latihan akan diberikan kepada pelajar.
- Pelajar dikehendaki menekan butang 'Mula'selepas arahan dibaca.
- Bergantung kepada ujian atau latihan yang disediakan oleh guru, pelajar boleh mimilih soalan yang hendak dijawab dengan menekan pautan *Jawab*.
- Selepas masuk ke dalam skrin menjawab soalan, butang 'Simpan' berfungsi untuk menyimpan jawapan pelajar. Pelajar boleh menekan butang 'Simpan' bila-bila masa. Selepas jawapan disimpan, pelajar boleh teruskan dengan soalan yang lain. Pelajar juga boleh mengembali ke jawapan yang telah disimpan untuk penyemakan dan membuat perubahan.

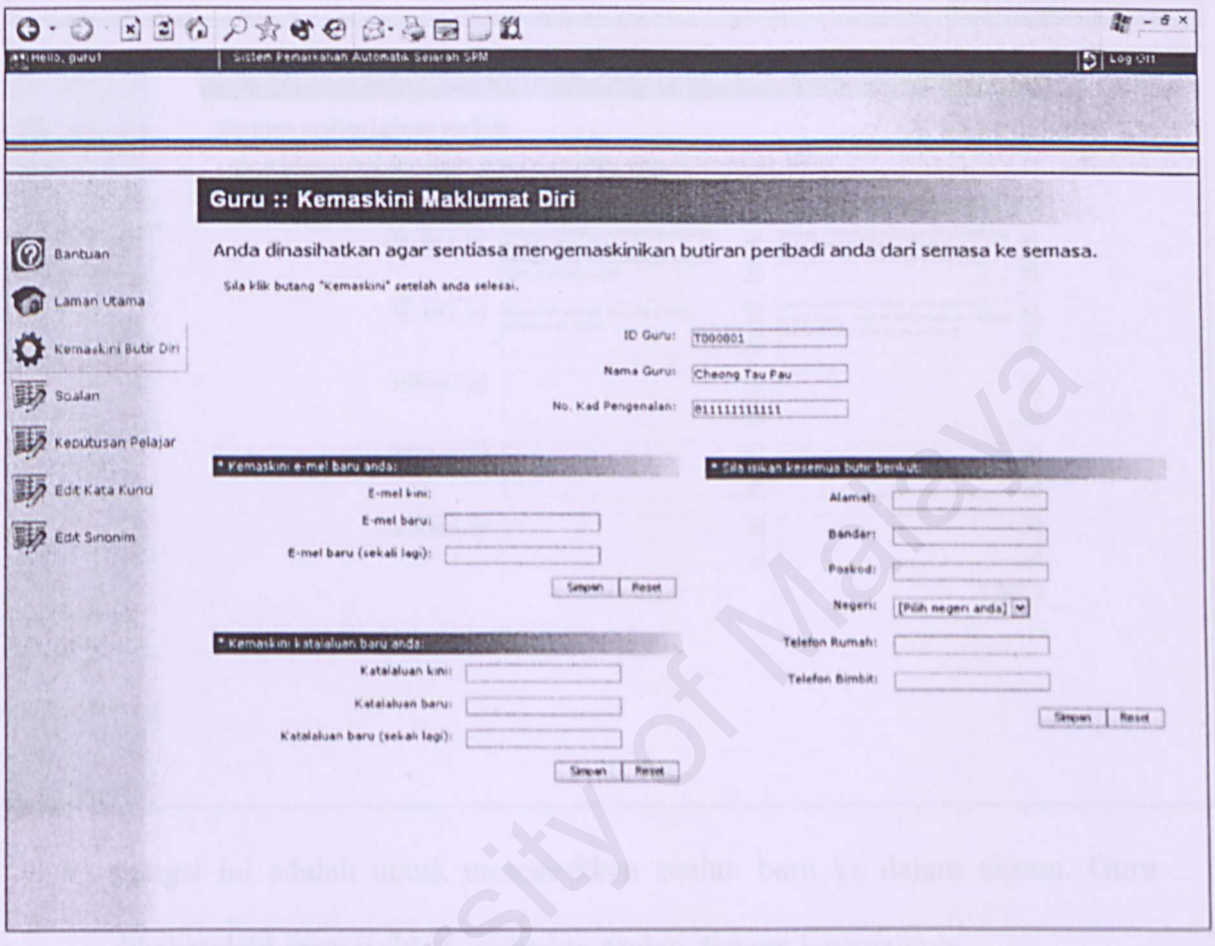
AMARAN : Tekan butang 'Simpan' tidak akan hantar jawapan untuk disemak.

AMARAN : Jika butang 'Simpan' tidak ditekan, jawapan akan hilang.

- Selepas pelajar menyiapkan semua soalan, pelajar dikehendaki menekan butang 'Hantar' untuk menghantarkan jawapan untuk disemak.
- Masa akan dikira dan masa yang tinggal akan dipaparkan di atas. Apabila masa tamat, pelajar akan dibawa ke skrin untuk hantar jawapan.

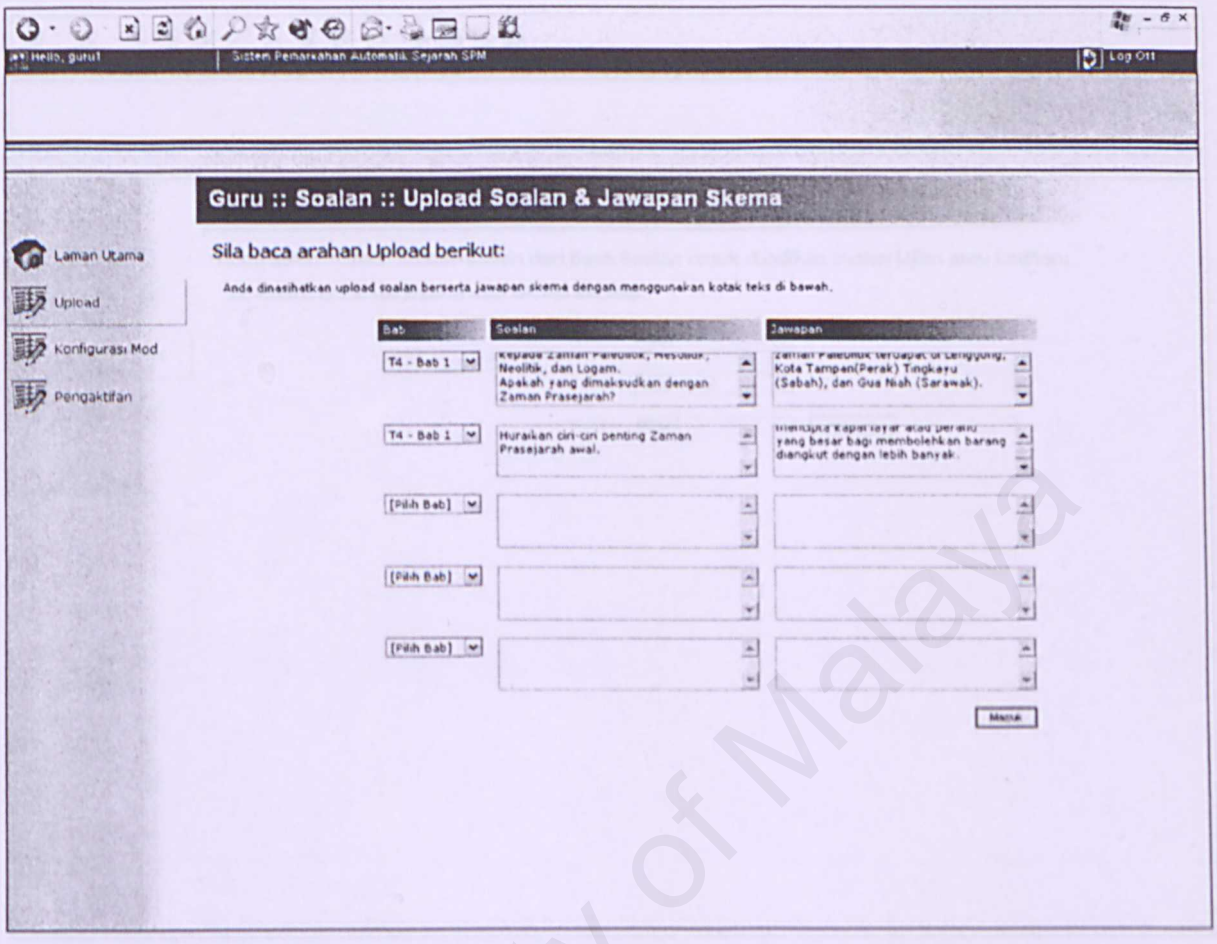
Pengguna :: Guru

Pengguna :: Guru :: Kemaskini Butir Diri



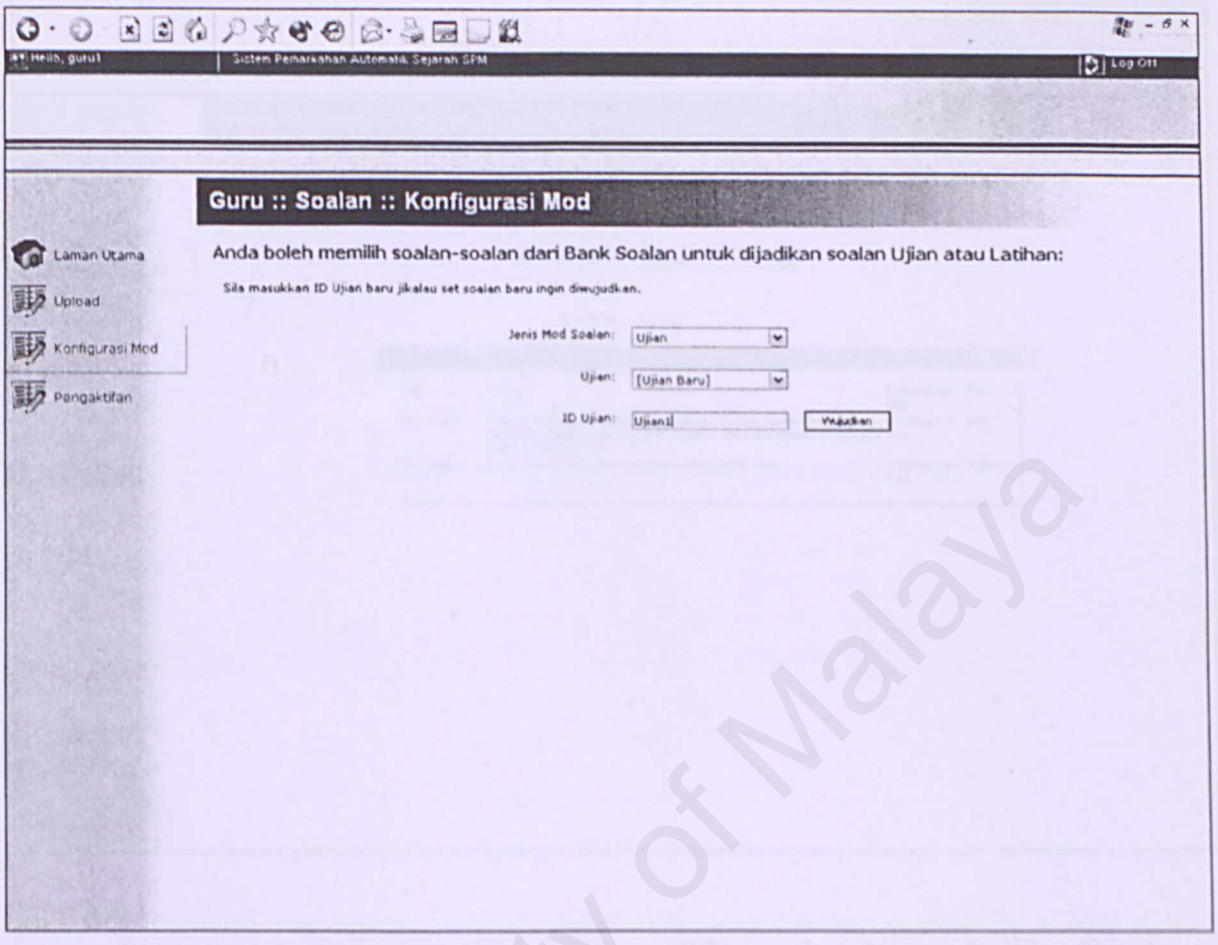
- Guru boleh mengkemaskinikan butir-butir diri dengan fungsi ini. Butir-butir seperti Alamat, Bandar, Poskod, Negeri, Telefon Rumah, Telefon Bimbit, E-mel, dan katalaluan boleh diubah atau dikemaskinikan oleh Pengguna Guru
- ID Pelajar, Nama Pelajar dan No. Kad Pengenalan tidak dapat ditukar. Hanya pengguna SysAdmin mempunyai hak untuk menukar butiran-butiran tersebut.

Pengguna :: Guru :: Soalan :: Upload

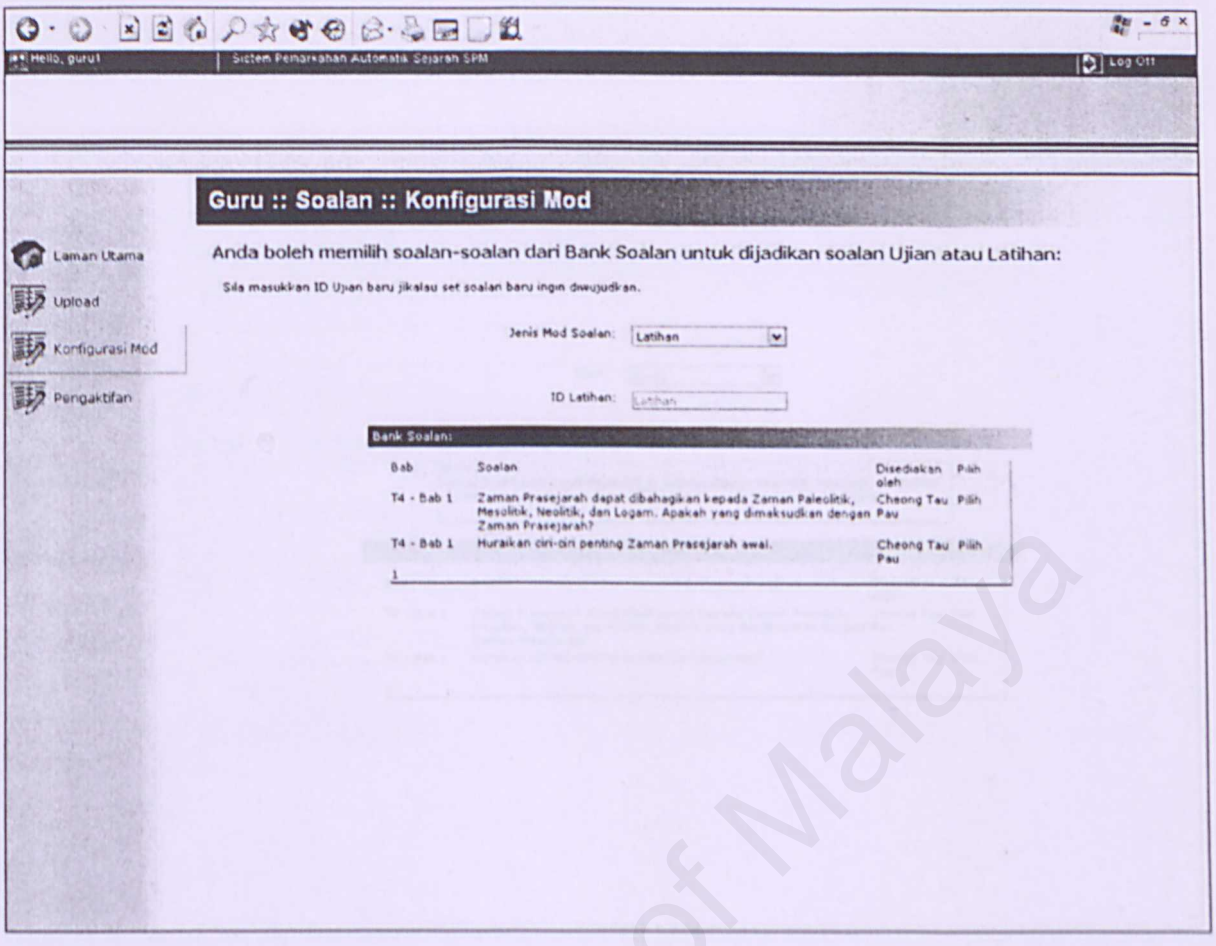


- Fungsi ini adalah untuk memasukkan soalan baru ke dalam sistem. Guru dikehendaki memasukkan pasangan soalan dengan jawapannya.
- Guru harus memilih bab soalan yang hendak dimasukkan.
- Guru boleh menaip soalan dan jawapan di ruang yang disediakan.
- Soalan tidak akan disimpan jika ruangan jawapan adalah kosong.
- Halaman ini membekalkan lima tempat ruangan, iaitu satu hingga lima soalan dan jawapannya boleh disimpan.
- Selepas guru bersedia untuk memasukkan soalan dan jawapan ke dalam sistem, guru dikehendaki menekan butang "Masuk"
- Sistem akan memberi amaran dan proses memasukkan soalan akan gagal jika soalan yang cuba dimasukkan itu telah wujud dalam sistem.

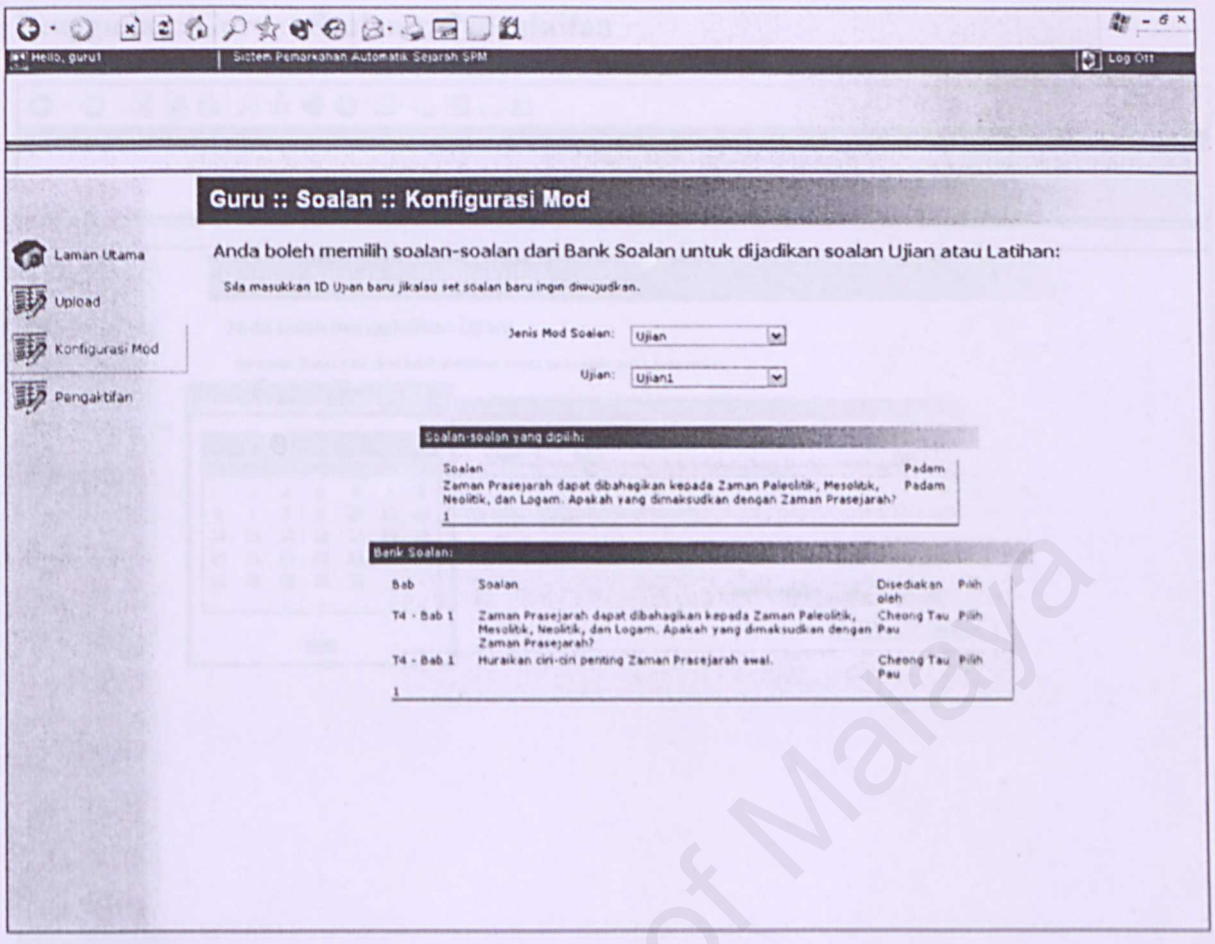
Pengguna :: Guru :: Soalan :: Konfigurasi Mod



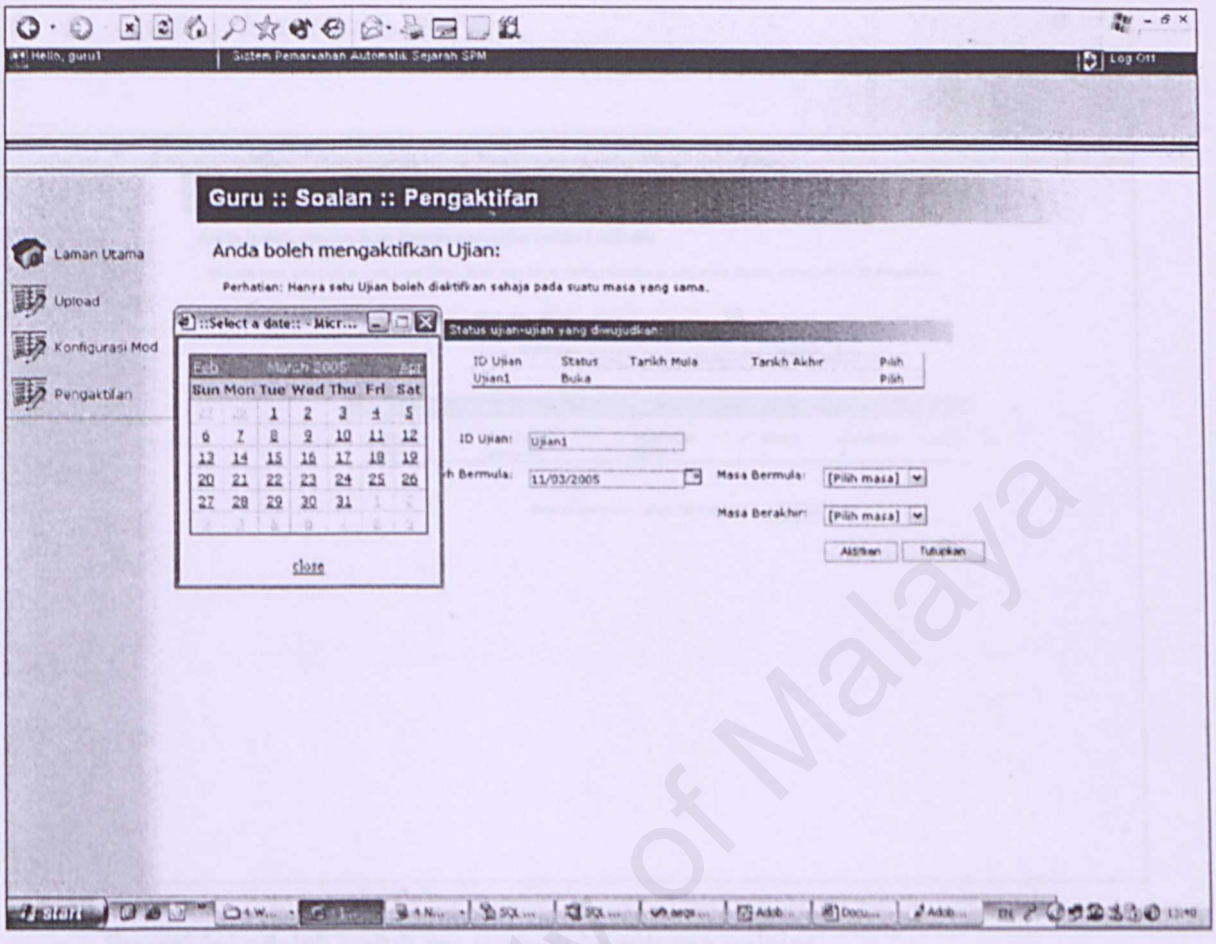
- Fungsi ini adalah untuk memilih soalan untuk ujian dan latihan
- Pilihan mod adalah untuk memilih sama ada menyediakan soalan untuk latihan atau menyediakan soalan ujian.
- Jika guru ingin menyediakan latihan, pilih pilihan latihan. Selepas itu, guru boleh memilih sebanyak lima soalan daripada bank soalan yang telah dimasukkan ke dalam sistem.



- Jika guru ingin menyediakan ujian, pilihan ujian harus dipilih. Untuk menyediakan suatu ujian baru, guru boleh menaip nama ujian baru di dalam petak yang disediakan (Nama ujian ini tidak boleh menganduni aksara tempat kosong). Selepas nama ujian ditaip, butang "Wujudkan" harus ditekan. Selepas itu, nama ujian baru ini akan wujud dalam senarai dan dapat dipilih.
- Untuk ujian yang telah wujud dalam senarai, guru boleh memilih ujian yang hendak disediakan. Selepas itu, guru boleh memilih sebanyak lima soalan daripada bank soalan yang telah dimasukkan ke dalam sistem.
- Setiap soalan yang dipilih akan disimpan dengan automatik.

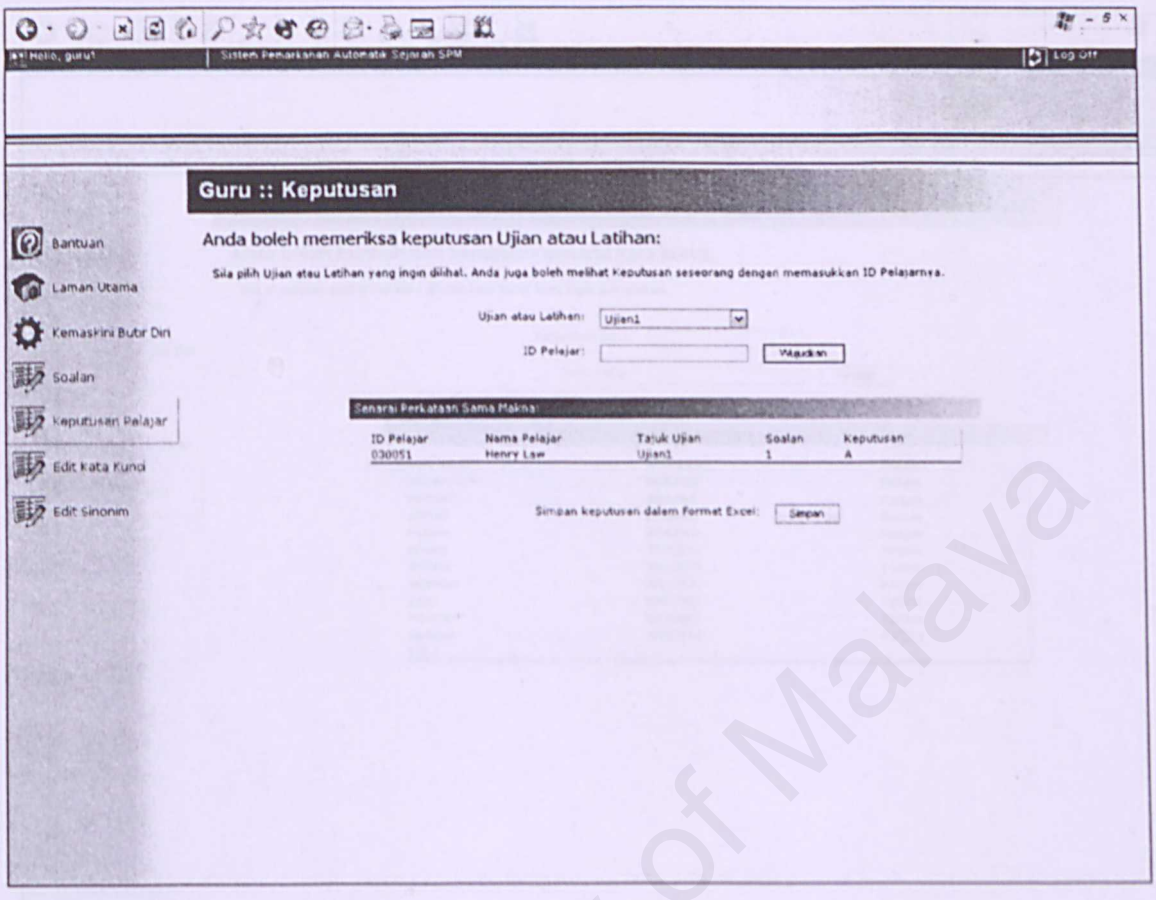


Pengguna :: Guru :: Soalan :: Pengaktifan



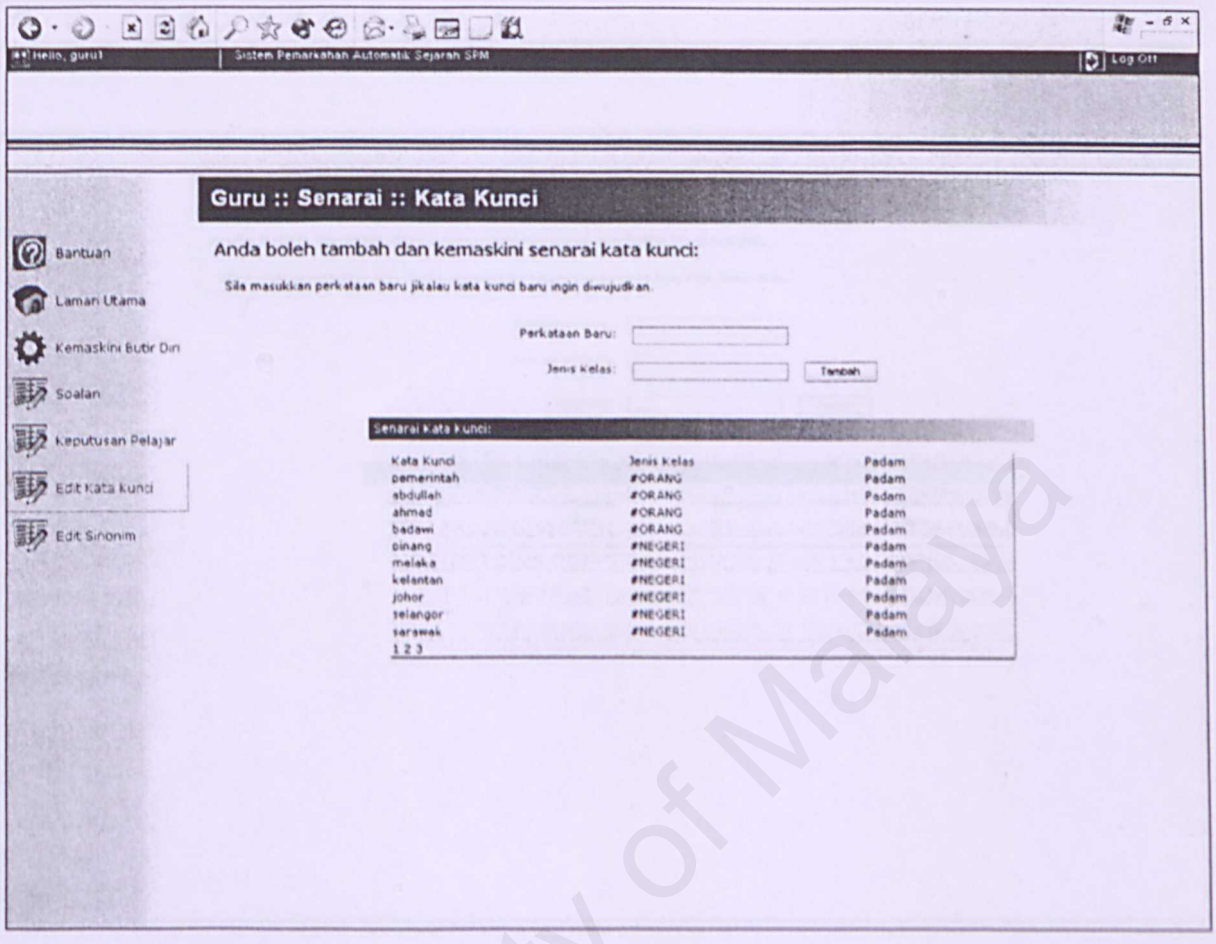
- Fungsi ini adalah untuk mengaktifkan ujian yang telah disediakan
- Semua ujian yang telah disediakan akan ditunjukkan di dalam jadual. Untuk mengaktifkan sesuatu ujian, guru dikehendaki klik pada perkataan pilih pada baris ujian yang hendak diaktifkan.
- Selepas itu, guru boleh klik pada kelender yang disediakan untuk memilih tarikh ujian. Seterusnya, masa mula dan masa tamat peperiksaan juga boleh dipilih dan ditentukan.
- Selepas langkah-langkah tersebut, guru dikehendaki klik butang "Aktifkan" untuk mengaktifkan ujian. Butang "Tutupkan" akan menamatkan ujian itu.

Pengguna :: Guru :: Keputusan Pelajar



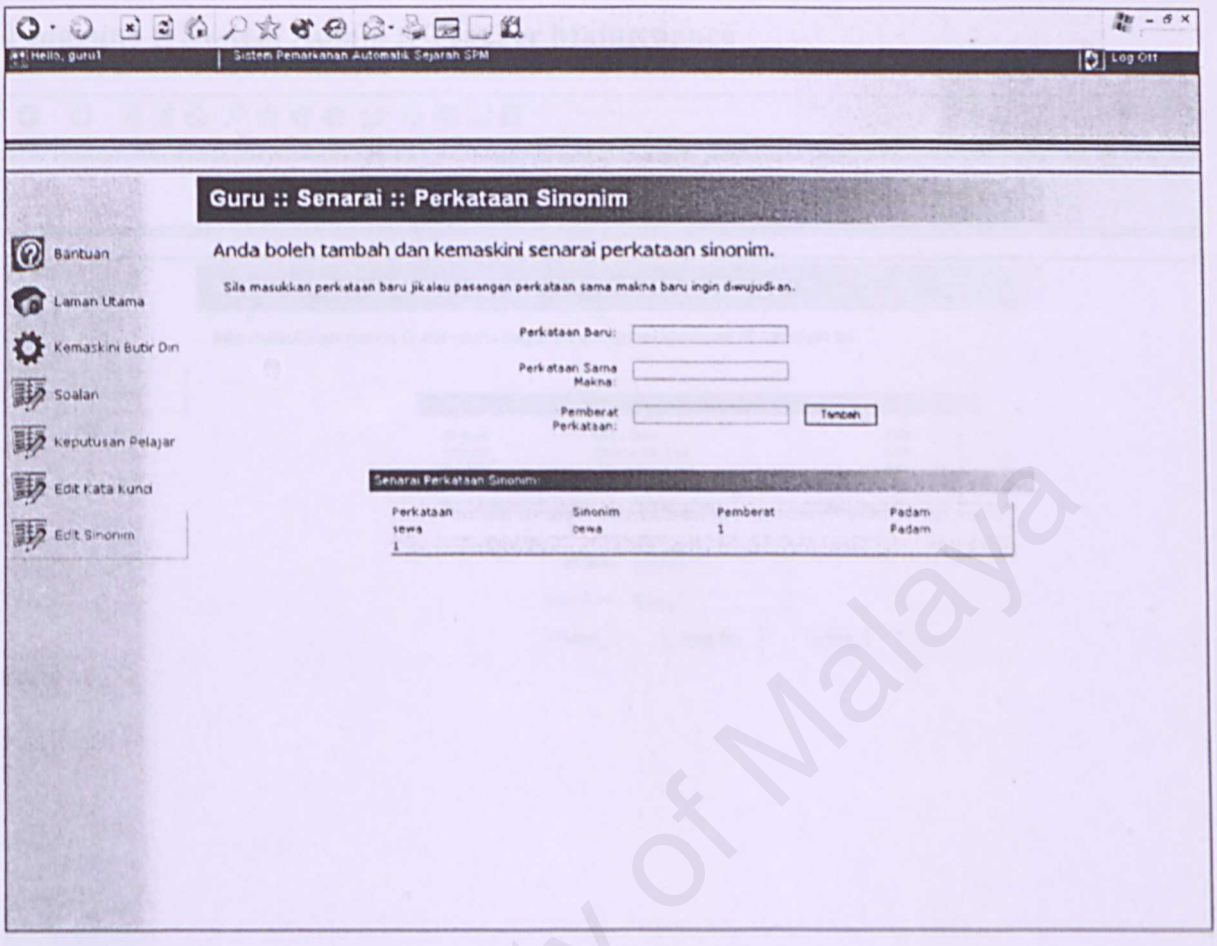
- Fungsi ini adalah untuk menyemak keputusan pelajar
- Apabila guru memasuki halaman ini, semua keputusan pelajar yang diajar akan ditunjukkan.
- Guru boleh memilih untuk melihat keputusan pelajar bagi ujian yang tertentu sahaja dengan memilih ujian pada pilihan ujian.
- Guru juga boleh melihat semua keputusan seseorang pelajar dengan menaip ID pelajar tersebut dalam petak yang disediakan dan menekan butang "Wujudkan"

Pengguna :: Guru :: Edit Kata Kunci



- Fungsi ini adalah untuk menambah kata kunci
- Guru dikehendaki menaip kata kunci yang ingin ditambah di petak yang disediakan. Selepas kata kunci ditaip, butang 'tambah' hendaklah ditekan untuk menambah kata kunci yang telah ditaip.
- Kata kunci yang telah ditambah akan ditunjukkan dalam jadual. Jika guru ingin memadam sesuatu kata kunci, guru boleh klik perkataan padam pada baris kata kunci yang hendak dipadam.
- Jenis kelas adalah pilihan. Jika perkataan tidak dimasukkan dalam ruang ini, kata kunci baru akan ditambah dengan bahagian jenis kelas sebagai "#UNKNOWN"

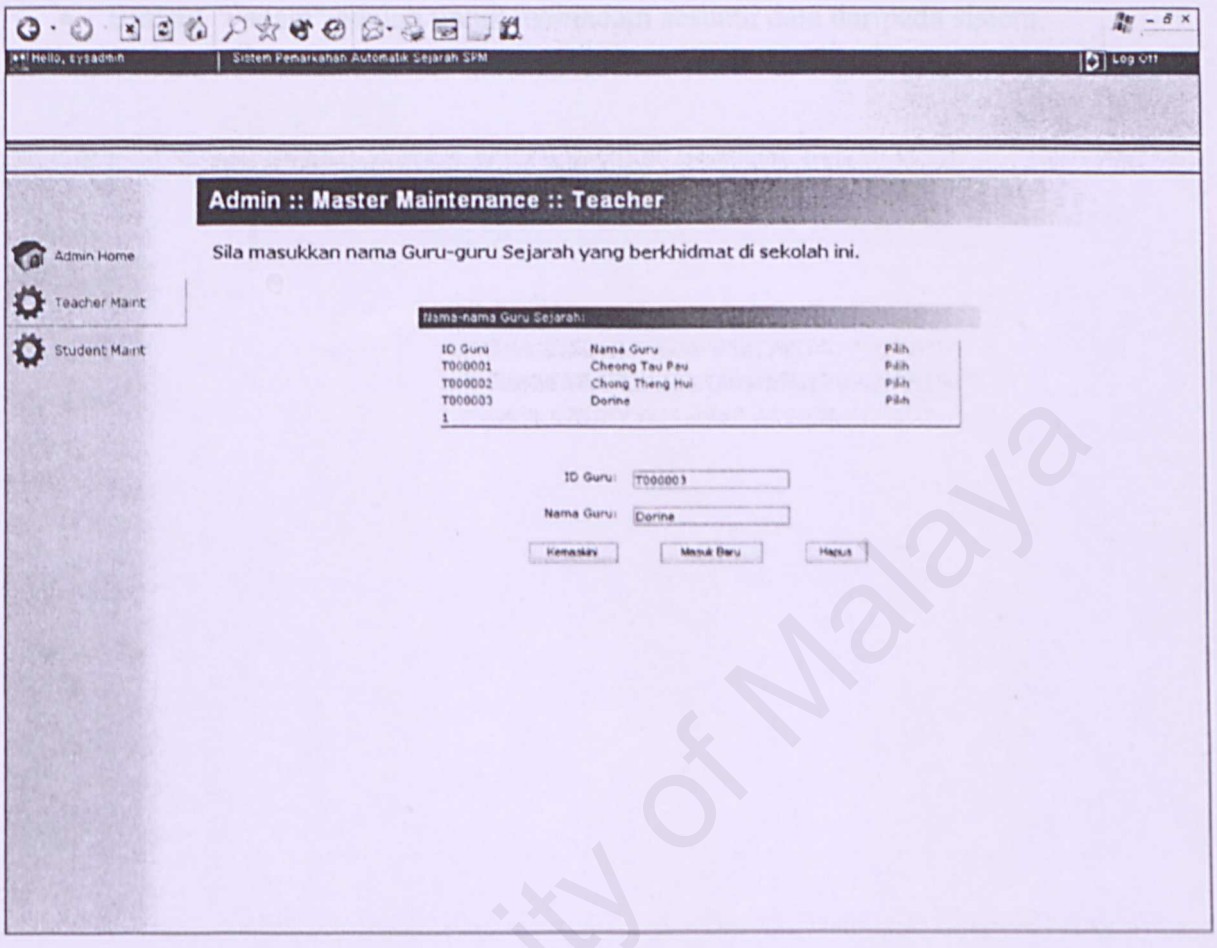
Pengguna :: Guru :: Edit Sinonim



- Fungsi ini adalah untuk menambah perkataan sinonim
- Guru dikehendaki menaip pasangan kata asal dan sinonim yang ingin ditambah di petak yang disediakan. Selepas pasangan kata asal dan sinonim ditaip, butang 'tambah' hendaklah ditekan untuk menambah pasangan kata asal dan sinonim yang telah ditaip.
- Sinonim yang telah ditambah akan ditunjukkan dalam jadual. Jika guru ingin memadam sesuatu sinonim, guru boleh klik perkataan padam pada baris kata kunci yang hendak dipadam.

Pengguna :: System Admin

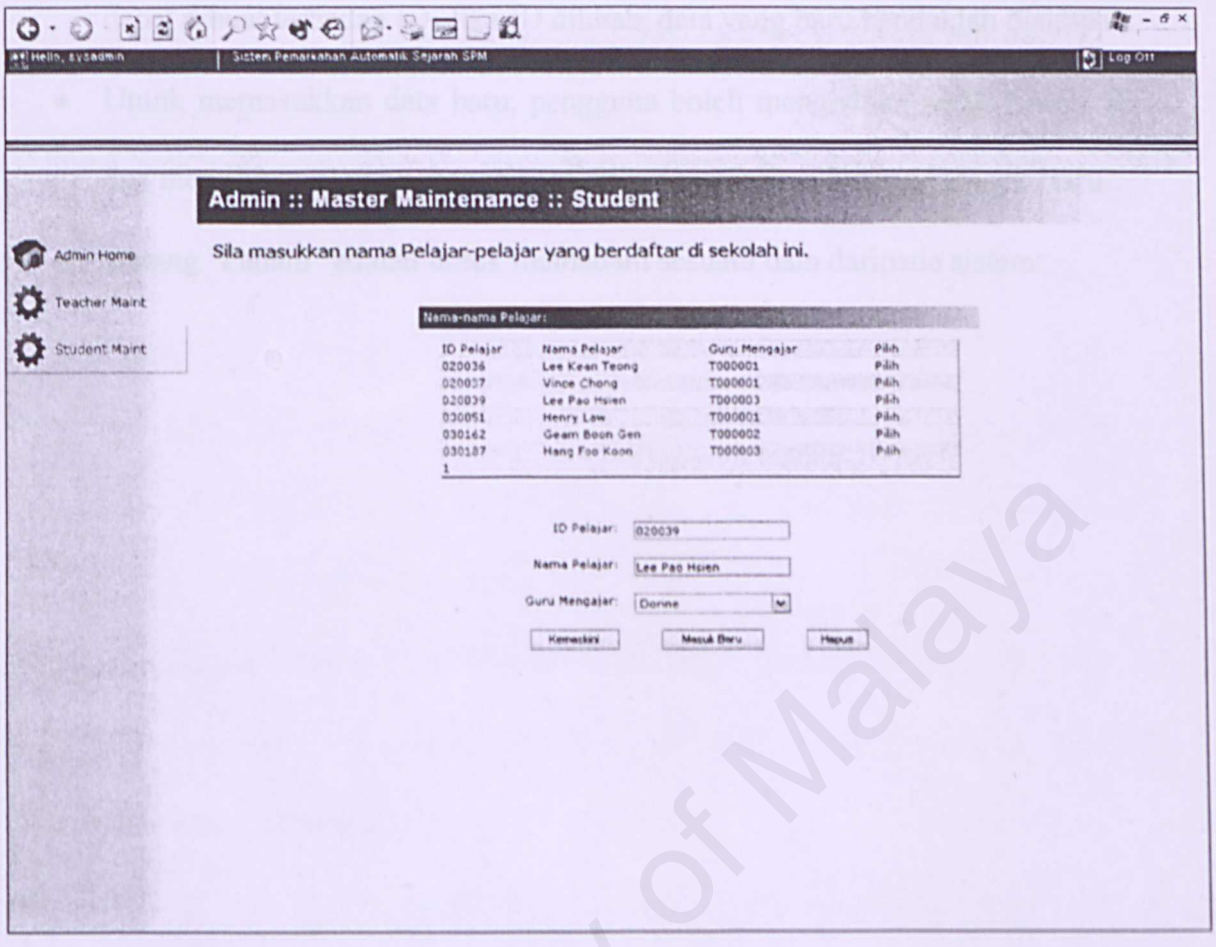
Pengguna :: System Admin :: Teacher Maintenance



- Fungsi ini adalah untuk menambah ID dan Nama Pengguna supaya pengguna itu dapat mendaftar dan menggunakan sistem ini. Pengguna yang dimasukkan ke dalam jadual ini boleh mendaftar sebagai pengguna guru sahaja.
- Jadual akan memaparkan semua data yang telah dimasukkan ke sistem. Untuk mengkemaskini data yang telah wujud, pengguna boleh klik perkataan pilih pada baris data yang ingin dikemaskinikan. Selepas itu, data yang telah wujud akan dipaparkan di dalam petak yang disediakan. Selepas mengubah data itu, pengguna harus menekan butang "Kemaskini" untuk menyimpan data yang telah diubah. Jika butang "Masuk Baru" ditekan, baris yang baru akan ditambah dan data yang wujud itu tidak akan diubah. Ubahan tidak dapat dibuat terhadap ID. Jika ID diubah, data yang baru hendaklah disimpan.

- Untuk memasukkan data baru, pengguna boleh mengisikan petak Nama dan ID seterusnya menekan butang "Masuk Baru".
- Butang "Padam" adalah untuk memadam sesuatu data daripada sistem.

Pengguna :: System Admin :: Student Maintenance



- Fungsi ini adalah untuk menambah ID dan Nama Pengguna supaya pengguna itu dapat mendaftar dan menggunakan sistem ini. Pengguna yang dimasukkan ke dalam jadual ini boleh mendaftar sebagai pengguna pelajar sahaja.
- Fungsi ini adalah untuk menambah ID dan Nama Pengguna supaya pengguna itu dapat mendaftar dan menggunakan sistem ini.
- Jadual akan memaparkan semua data yang telah dimasukkan ke sistem. Untuk mengkemaskini data yang telah wujud, pengguna boleh klik perkataan pilih pada baris data yang ingin dikemaskinikan. Selepas itu, data yang telah wujud akan dipaparkan di dalam petak yang disediakan. Selepas mengubah data itu, pengguna harus menekan butang "Kemaskini" untuk menyimpan data yang telah diubah. Jika butang "Masuk Baru" ditekan, baris yang baru

akan ditambah dan data yang wujud itu tidak akan diubah. Ubahan tidak dapat dibuat terhadap ID. Jika ID diubah, data yang baru hendaklah disimpan.

- Untuk memasukkan data baru, pengguna boleh mengisikan petak Nama, ID dan memilih guru yang mengajar seterusnya menekan butang "Masuk Baru".
- Butang "Padam" adalah untuk memadam sesuatu data daripada sistem.

References

[1] N. Idris. (2003). *Automated Grading Essay for Malay Text using information retrieval and nearest neighbor algorithm.*

[2] N. Idris and S.M.F.D. Syed Mustapha. *Stemming for Term Conflation in Malay Texts*. In International Conference on Artificial Intelligence (IC-AI 2001). Las Vegas, 2001.

[3] F. Ahmad, M. Yusof and T.M. T. Sembok. *Experiments with a Stemming Algorithm for Malay Words*. Journal of the American Society for Information Science, 47(12):909-918, 1996.

[4] N. Idris and S. M. F. D. Syed Mustapha. (2001). *Automated Essay Grading: An Application for Historical Malay Text*. International Conference on Artificial Intelligence and Applications (IC-AI 2001).

[5] S. R. Schach. (2005). *Object-Oriented and Classical Software Engineering*. 6th Edition. McGraw Hill.

[6] Fadillah Z. Tala. (2003) *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*.

[7] Tata Bahasa Melayu. (URL - *http://tatabahasabm.tripod.com*), 16/08/2004.

[8] Three prominent writing assessment programs. (URL - *http://edres.org/betsy/three_prominent.htm*), 15/09/2004.

[9] Knowledge Analysis Technologies. (URL - *http://www.k-a-t.com*), 15/09/2004.

[10] Approaches to the Computerized Assessment of Free Text Responses. (URL - *http://cvu.strath.ac.uk/dave/publications/caa99.html*), 15/09/2004.

[11] R. Baeza, B. Ribeiro. (1999). *Modern Information Retrieval*, Pearson Education Limited.

References

[1] N. Idris. (2003). *Automated Grading Essay for Malay Text using information retrieval and nearest neighbor algorithm.*

[2] N. Idris and S.M.F.D. Syed Mustapha. *Stemming for Term Conflation in Malay Texts*. In International Conference on Artificial Intelligence (IC-AI 2001). Las Vegas, 2001.

[3] F. Ahmad, M. Yusof and T.M. T. Sembok. *Experiments with a Stemming Algorithm for Malay Words*. Journal of the American Society for Information Science, 47(12):909-918, 1996.

[4] N. Idris and S. M. F. D. Syed Mustapha. (2001). *Automated Essay Grading: An Application for Historical Malay Text*. International Conference on Artificial Intelligence and Applications (IC-AI 2001).

[5] S. R. Schach. (2005). *Object-Oriented and Classical Software Engineering*. 6th Edition. McGraw Hill.

[6] Fadillah Z. Tala. (2003) *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia.*

[7] Tata Bahasa Melayu. (URL - *http://tatabahasabm.tripod.com*), 16/08/2004.

[8] Three prominent writing assessment programs. (URL - *http://edres.org/betsy/three_prominent.htm*), 15/09/2004.

[9] Knowledge Analysis Technologies. (URL - *http://www.k-a-t.com*), 15/09/2004.

[10] Approaches to the Computerized Assessment of Free Text Responses. (URL - *http://cvu.strath.ac.uk/dave/publications/caa99.html*), 15/09/2004.

[11] R. Baeza, B. Ribeiro. (1999). *Modern Information Retrieval*, Pearson Education Limited.

[12] E-Rater Demo. (URL -*http://www.etstechnologies.com/html/eraterdemo.html*), 30/08/2004.

[13] Page E.B. (1966). *The Imminence of Grading Essays by Computer*, Phi Delta Kappan, January, 238-243.

[14] Foltz P.W. (1996). *Latent Semantic Analysis for Text-Based Research, Behavior Research Method, Instruments and Computers*, 28, 197-202.

[15] Williams R. (2001). *Automated Essay Grading: An Evaluation of Four Conceptual Models*, Teaching and Learning Forum 2001.

[16] S. Y. Tai, C. S. Ong, N. A. Abdullah. (1990). *On Designing an Automated Malaysian Stemmer for the Malay Language.* Proceedings of the 5th International Workshop Information Retrieval with Asian Languages.

[17] What is Stemming? (URL-*http://www.comp.lancs.ac.uk/computing/research/stemming/general/index.htm*), 30/08/2004.

[18] Snowball. (URL - *http://snowball.tartarus.org/*), 30/08/2004.

[19] Porter Stemming Algorithm. (URL-*http://www.tartarus.org/~martin/PorterStemmer/*), 20/08/2004.

[20] Project Essay Grader Information Centre. (URL-*http://134.68.49.185/pegdemo/*), 28/08/2004.

[21] Berita Harian Online. (URL- *http://www.bharian.com.my*), 06/03/2005.

Any questions or comments, please kindly contact me.
Author: CHONG Theng Hui
Email: thenghui@hotmail.com