

Prevention of Textual Plagiarism Application

by

Nor Atikah binti Mohd Nor

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Technology (Hons)
(Information & Communication Technology)

MAY 2011

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Prevention of Textual Plagiarism Application

by

Nor Atikah Binti Mohd Nor

A project dissertation submitted to the

Information and Communication Technology

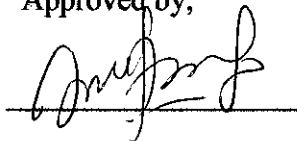
Universiti Teknologi PETRONAS

in partial fulfillment of the requirement for the

BACHELOR OF TECHNOLOGY (Hons)

INFORMATION AND COMMUNICATION TECHNOLOGY

Approved by,



(Emelia Akashah Binti Patah Akhir)

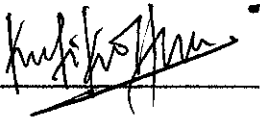
UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

MAY 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



NOR ATIKAH BINTI MOHD NOR

ABSTRACT

This project is mainly about detecting and preventing plagiarism among UTP students. In order to achieve this, a system which named as *Prevention of Textual Plagiarism Application* will be developed which provides the capability to detect similarities between documents submitted by students. Thus, the main focus of this project is to perform a study on how to detect textual plagiarism. Word –for- word plagiarism is the most noticeable and serious form of plagiarism. This form of plagiarism can be categorized as a form of direct stealing without proper acknowledgment and consent of another’s work. Rabin – Karp Algorithm which is a string searching algorithm that uses hashing to compare the strings will be integrated in this project as well. Some fact findings have also been carried out in order to perform the study on plagiarism. As a result, this project is able to compare against each of the files submitted by students in order to find for the similarities among them. The percentage of similarities will then be generated in a text format as the way to ease the lecturers for detecting plagiarism activities among students.

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to Allah S.W.T because of His mercy and blessings had given me the strengths to face challenges in completing this Final Year Project.

I would like to express thousands of appreciations, highest gratitude and sincere thanks to my supervisor, Ms. Emelia Akashah Binti Patah Akhir for all the valuable guidance, poritive and constructive critics and advice that have given to me upon completing this project.

I would also like to express my gratitude and thanks to all lecturers and tutors in Information and Communication Technology (ICT) department who eventually helped me during the project and also in sharing their knowledge and information, which has made this project a very great success. Not to forget, special thanks to all my friends who helped and share their knowledge with me during the development of this project.

Last but not least, I acknowledge with greatest appreciation to other personnel not mentioned above whom gave me such great support in completing this project successfully and to UTP for giving me a chance to gain knowledge and experiences during the Final Year Project development. A sincere apology for all the problems inconvenience caused. All of your kindness and cooperation are highly appreciated and will be fondly remembered.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENT	ii
CHAPTER 1 : INTRODUCTION	
1.1 Background of Study	1
1.2 Problem Statement	3
1.3 Objectives	4
1.4 Scope of Study	4
CHAPTER 2 : LITERATURE REVIEW	
2.1 What is Plagiarism?	5
2.2 Suspecting Plagiarism	6
2.3 Existing Plagiarism Detection Tools	8
CHAPTER 3 : METHODOLOGY	
3.1 System Methodology	
3.1.1 System Development	17
3.1.2 Rabin – Karp Algorithm	19

3.2 Tool Requirement	23
3.3 Project Activities	24
CHAPTER 4 : RESULTS AND DISCUSSIONS	
4.1 Results	25
4.1.1 System Overview	25
4.1.2 Survey Results	28
4.2 System User Interface and Functions	
4.2.1 Description on System User Interface	31
4.2.2 System Functions	32
CHAPTER 5 : RECOMMENDATIONS	33
CHAPTER 6 : CONCLUSION	35
REFERENCES	36
APPENDICES	39

LIST OF FIGURES

- Figure 1: Dutch system Ephorus – Plagiarism Report
- Figure 2: Dupe Free Pro
- Figure 3: WCopyFind
- Figure 4: Essay Verification Engine
- Figure 5: System Development
- Figure 6: The initial fingerprints.
- Figure 7: Updating the fingerprint
- Figure 8: Second Comparison
- Figure 9: Third Comparison
- Figure 10: Fourth Comparison
- Figure 11: Fifth Comparison
- Figure 12: Use Case Diagram for Prevention of Textual Plagiarism Application
- Figure 13: Methods of Plagiarizing
- Figure 14: Frequency of Plagiarizing
- Figure 15: the percentile ranks associate with SUS scores and letter grades
- Figure 16: Timeline for FYP Part 1
- Figure 17: Timeline for FYP Part 2
- Figure 18: Splash Screen
- Figure 19: Main Window Form
- Figure 20: Select a directory
- Figure 21: Generate a report
- Figure 22: About Form

LIST OF TABLES

Table 1: Free-to-use plagiarism detection tools

Table 2: Proprietary Plagiarism Detection Tools

ABBREVIATION

ICT – Information and Communication Technology

FYP – Final Year Project

PDF – Portable Document Format

HTML – Hypertext Markup Language

RTF – Rich Text Format

TXT – Text File

DOC – Document

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF STUDY

Plagiarism can be defined as the copying or paraphrasing of someone else's ideas, information, language, or writing into one's own original work. This act of plagiarizing is done without appropriate acknowledgement of the original source [1]. The definitions of plagiarism is also ranged from common dictionary meanings or single words that all signified some sort of immoral intent which are lying, cheating, stealing, dishonesty and deception. What emerges from these attempts to define plagiarism is that it falls under the umbrella of "academic cheating", which commonly is seen as an academic misconduct offence and deserving punishment [2].

The advent and widespread use of the Internet has made the occurrence of plagiarism flourishing excessively. This is due to the easiness provided by the Internet where students can just simply typing the keyword-driven Internet research using search engines like Google and Yahoo. Then all the related information on a wide range of topics will be viewed on the students' screen of their laptop or desktop. This information can be cut-and-pasted into new documents with minimal effort.

In doing any assignments or paper works, students should generate their own ideas and solutions to produce high quality of paper works. Thus, it is definitely the

responsibilities of academic to ensure the plagiarism activities are able to be detected and prevented among the students. Fortunately, there are numbers of existing mechanisms which can help academicians especially lecturers in solving this problems.

Nevertheless, this problem seems to have no end solution as the numbers of plagiarism keep increasing day-by-days. Thus, the purpose of this project is to provide a new mechanism which will be developed in complementing the previous application. This new mechanism is significant to the lecturers as it will assist them by giving a solution to the plagiarism in a friendly and better manner. Besides, it will also help the organization, Universiti Teknologi PETRONAS (UTP) in producing the high quality graduates as well as well-rounded students.

This paper looks at the increasing amount of academic resources available to aid tutors detect plagiarism. Support for finding similarities in both the (long established) constrained text and the (more recent) free-text areas of writing are examined. The paper ends with a look at the future of anti plagiarism measures.

1.2 PROBLEM STATEMENT

The term “plagiarism” is derived from the Latin term for plundering. In fact, the idea that kidnapping the words of others, as a child is kidnapped from a parent is appropriate to explain the way in which plagiarism is defined. Plagiarism has always existed, but the growth of the Internet in an online world has made it much easier to do. There are lists of reasons as to why students engage in plagiarism due to their attitude themselves instead of blaming the technological advances completely. A typical list might look like the following taken from the University of Alabama in Huntsville (2007):

- i. Lack of research and writing skills
- ii. Misconceptions about terminology
- iii. Problem evaluating Internet sources
- iv. Confusion about how to cite sources
- v. Product-oriented writing assignments
- vi. Poor time management and organizational skills

That is the reason why the number of students produces non-original assignments and plagiarism activities day by days become overpowering. In a 2006 Canadian survey, Birchard says, “more than half of the undergraduates and 35% of the graduate students surveyed admitted to some form of cheating on written course work, such as failing to footnote, turning to someone else’s work, or falsifying a bibliography” [3].

Plagiarism can also contribute to another side impacts such as reducing the qualities of students’ project or assignments where do not meet the education quality standard. This is because lecturers will tend to give wrong evaluation upon their work. Moreover, it will also demoralize the honest students since successful plagiarism will encourages lifelong dishonesty.

1.3 OBJECTIVES

1. To perform a study on how to detect textual plagiarism.
2. To enable lecturers to detect the similarities of students' assignments by using *Prevention of Textual Plagiarism Application*.
3. To identify any plagiarism activities in a better manner by checking all the documents against the materials submitted by students.
4. To reduce the numbers of students committing in plagiarism activities.

1.4 SCOPE OF STUDY

The scopes of study for developing *Prevention of Textual Plagiarism Application* are:

- To perform a study on how to detect textual plagiarism.
- To develop a system that enable to minimize lecturer's effort in detecting plagiarism of text documents.
- The system also will assist lecturer in evaluating the students' work.
- The system is developed to cater for only word-for-word plagiarism.
- The system will cater only .txt and .doc file formats

CHAPTER 2

LITERATURE REVIEW

2.1 What is plagiarism?

The definitions of plagiarism which are from the variant of sources can be deduced into three parts [4–5]. First, that the plagiarist is using someone else’s work, or ideas. Second, that he or she does so without proper acknowledgement and third, that mere paraphrasing or rephrasing of such work or ideas in no way mitigates the crime.

In October 2010, a survey has been conducted by UiTM students via the student portal as part of the anti-plagiarism campaign in UiTM. The result has revealed that almost 48% from the total of 1,884 students participated in the survey admitted that they have committed plagiarism in doing their assignments [6]. Whereas, 66% of 16,000 students from 31 prestigious U.S. universities have cheated at least once, says 1991 Rutgers University study.

UiTM Anti-Plagiarism Survey, October 2010 [6]

**48% from the total of 1,884 students participated
have committed plagiarism**

Rutgers University Study, 1991 [7]

**66% of 16,000 students from 31 prestigious U.S.
universities have cheated at least once**

2.2 Suspecting Plagiarism

There are several ways of suspecting any kind of plagiarism activities which are [7]:

- **Changes of vocabulary or style** – if the vocabulary or writing style changes significantly within the paper, this can indicate possible cut-and-paste plagiarism. Pay particular attention to the style in the introduction and conclusion as they are likely written by the author him/herself.
- **Incoherent text** – if the flow of a text is not consistent or smooth, this could indicate that part of the text is not the author's own work.
- **Amount of similarity between texts** – there may be a certain amount of similarity between texts written about the same topic, but it is unlikely that independently written texts would share large amounts of the same or similar text.
- **Order of similarity between texts** – if the order of matching words or phrases between two texts is the same in both texts, this may indicate plagiarism.
- **Dependence on certain words and phrases** – an author may prefer using particular words or phrases. Consistent use of these words and phrases in a text written by some-one else with different word preferences may indicate plagiarism.
- **Outdated text or sources** – if a paper contains statements which are no longer true or if all the sources cited are several years old, plagiarism may be indicated.

- **Mistakes** – an obvious clue. It is very unlikely that independently written texts would have the same spelling or grammatical mistakes.
- **Text Size & Font** – another obvious clue. If sections of the paper appear in a different font or size, cut-and-paste plagiarism is likely.
- **Citation style** – if more than one citation style is used in the paper, this may indicate a cut-and-paste plagiarism.
- **Dangling references** – if references appear in the text but not in the footnotes or bibliography, this may indicate a cut-and-paste plagiarism

2.3 Existing Plagiarism Detection Tools

There are numbers of plagiarism detection tools which already exist in this millennium year. These existed tools are mainly focusing in detecting and preventing plagiarism among students as the vision to produce high quality graduates. The majority of the plagiarism detection tools have been created for students' essays [8-11]. Besides that, detecting plagiarism for research papers, case studies and review papers are also the reason of why most of the tools are created [12,13].

“A good plagiarism software will compare published work in all sources, magazines, academic journals, books and billions of academic papers,” according to Harrel (2009). Tools for plagiarism detection can be divided regarding their comparison policy into two main categories:

- (i) tools that search in a document database that the user provides
- (ii) tools that conduct an internet-wide searching.

Another more technical is the classification into two main classes, namely those being web based tools, and the other is those consisting of computer based tools requiring some kind of local installation. “Simple” users usually prefer a web based tool that could quickly accomplish their search, while “advanced” or “professional” users may prefer a computer based tool, capable of multi-searching batches of files each time.

Name	Proprietary/ Free	Platform	URL
Plagiarism Detect	Free	Web-based	http://plagiarismdetect.com/
Article Checker	Free	Web-based	http://www.articlechecker.com/
Dupe Free Pro	Free	PC installation	http://www.dudefreepro.com/

DOC Cop	Free	Web-based	http://www.doccop.com/index.html
The Plagiarism Checker	Free	Web-based	http://www.dustball.com/cs/plagiarism.checker/
Viper-the Anti-plagiarism Scanner 1.5	Free	PC installation	http://www.scanmyessay.com
Dupli Checker	Free	Web-based	http://www.duplichecker.com/
Plagium	Free	Web-based	http://www.plagium.com/
SeeSources	Free	Web-based	http://plagscan.com/seesources/
Chimpsky	Free	Web-based	http://chimpsky.uwaterloo.ca/
Copytracker	Free	Web-based	http://copytracker.ec-lille.fr/
Splat	Free	PC installation	http://splat.cs.arizona.edu/
Wcopyfind	Free	PC installation	http://plagiarism.phys.virginia.edu/Wsoftware.html
Copy Tracker	Free	Web-based	http://copytracker.ec-lille.fr
Pairwise	Free	Advanced Sever Installation	http://www.pairwise.cits.ucsb.edu/

Table 1: Free-to-use plagiarism detection tools

Moreover, plagiarism detection tools are also classified in open source and proprietary. For instance, the “simple” user can be fully satisfied by the usability and the effectiveness of open source tools. This is because, that user is just going to check a few of his/her papers per year which not necessarily need to use the proprietary ones. On the other hand, a more advanced user should use a combination of open source and proprietary tools which are as listed in Table 2, the lists of some licensed plagiarism detection tools.

Name	Version	Platform	URL
DupeCop	Proprietary	Web-based	http://www.dupecop.net/index-online.html
Turnitin	Proprietary	Web-based	http://turnitin.com/static/index.php
EVE2	Proprietary	PC Installation	http://www.canexus.com/
Ithenticate	Proprietary	Web-based	https://www.ithenticate.com/
SafeAssign	Blackboard plug in	Web-based	http://www.mydropbox.com/
Copycatch	Proprietary	PC Installation	http://www.cflsoftware.com/
Ephorus	Proprietary	PC Installation	http://fronter.info/downloads/Ephorus.pdf

Table 2: Proprietary Plagiarism Detection Tools

Based on the list of plagiarism tools detection in Table 1 and Table 2, it can be deduced that the computer based tools are much more preferable to use by the lecturers compared to the web based tools. This is because computer based tools providing the capability of multi-searching batches of files each time. Moreover, as the lecturers are supposed to evaluate for bunch of students' paper work or assignment per year.

Thus, lecturers will found that the tools are hard to use and consume much time in order to evaluate the entire students' work if they are relying on the web based tools. For instance, PAIRwise might take up to 45 minutes for turnaround which is the total time between submission of a process and its completion (Board of Regents of the University System of Georgia (BOR), 2007). Thus, there are several acceptable and suitable software systems to be implemented by the lecturers:

1. Dutch system Ephorus:

- It is classified as good software where it took the first place over the seventeen software systems for plagiarism and collusion detection survey. That software was tested in Berlin, Germany, in September 2007 [14].
- All the work handed by the students will automatically be compared into the system with (i) documents on the internet, (ii) other documents handed in by students and (iii) documents from other institutions using Ephorus [15].
- Ephorus Plagiarism control works:
 - *Hand-in folder* - The Fronter hand-in folder is adjusted so that all work submitted is automatically scanned through the plagiarism control system.
 - *Growing database* - Each time the system is used, the database of documents and student papers for plagiarism detection is increased.
 - *File formats* - Plain Text, Word, OpenOffice, PDF and HTML files can all be scanned.

2. Dupe Free Pro[17]:

- This plagiarism detection tool is useful when users need to perform rewrites or for anyone who writes articles intended to be published on the Web. It allows the users to check if the similarity percentages meet users' requirements.
- The similar texts will then be highlighted in both of them to show the similarity. The percentage of duplicate content the program has found will be displayed at the bottom of the screen.
- The numbers of words, sentences and paragraphs contained in each document also will be counted. The highlighted content will then be searched to the Internet such as Google, Yahoo! and MSN, looking for any replication texts.

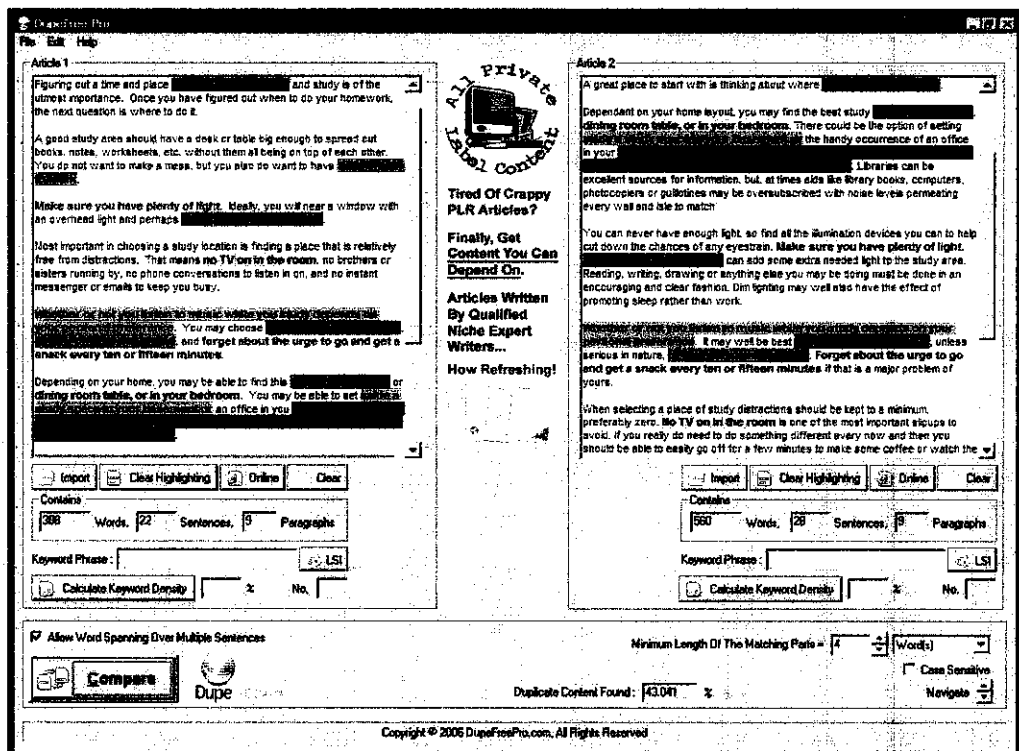


Figure 2: Dupe Free Pro

3. WCopyFind [18]:

- WCopyFind is also similar with the other plagiarism detection tools where it examines a collection of document files. In order to search for the similarities between those files, it extracts the text portions and look through them.
- Advantages :
 - Can support for text, html, and some word processor which just in the .doc format.
 - Can find which words are most common in the news.
 - Can list and count the matching words between two documents.
 - Generate html report files.
- Disadvantages :
 - Cannot search the web or internet to find matching documents for the users.
 - Users must specify which documents it compares
 - Cannot support for pdf files directly.

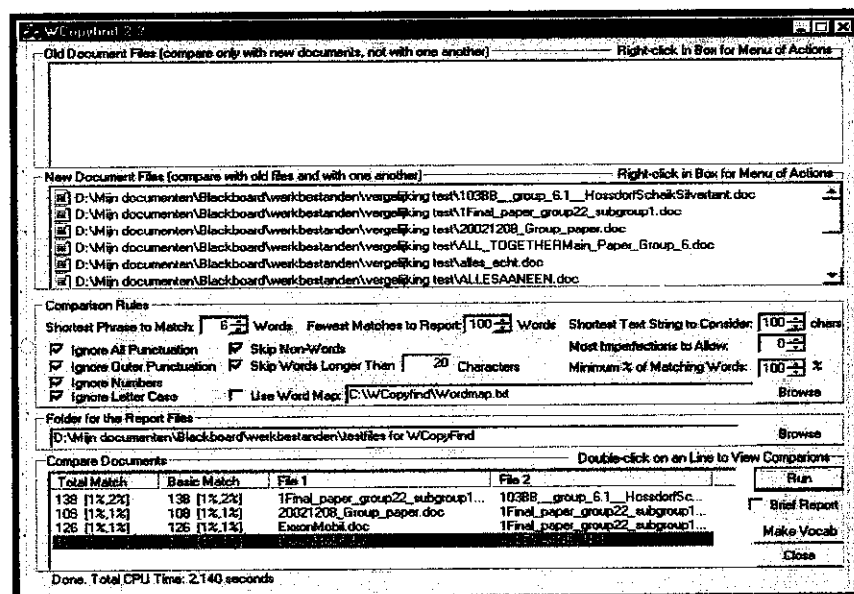


Figure 3: WCopyFind

4. EVE2 (Essay Verification Engine 2.4) [19]:

- EVE2 is a tool that is installed on the user's workstation, but connects to the Internet while searching the document. The tool is very user friendly and can process several documents at once.
- Disadvantages: It requires a strong computer with extensive processing power. It requires enormous amounts of processor capacity, which slows down the user's operating system. Also, the fact that the tool itself is not very stable sometimes leads to runtime errors or reports without an outcome.
- Advantages: Results that are generated lead to good indications, summing up the sources of the (underlined) matches that are found. The report is a RTF-document, which does not provide the opportunity to directly click on the links that are found.

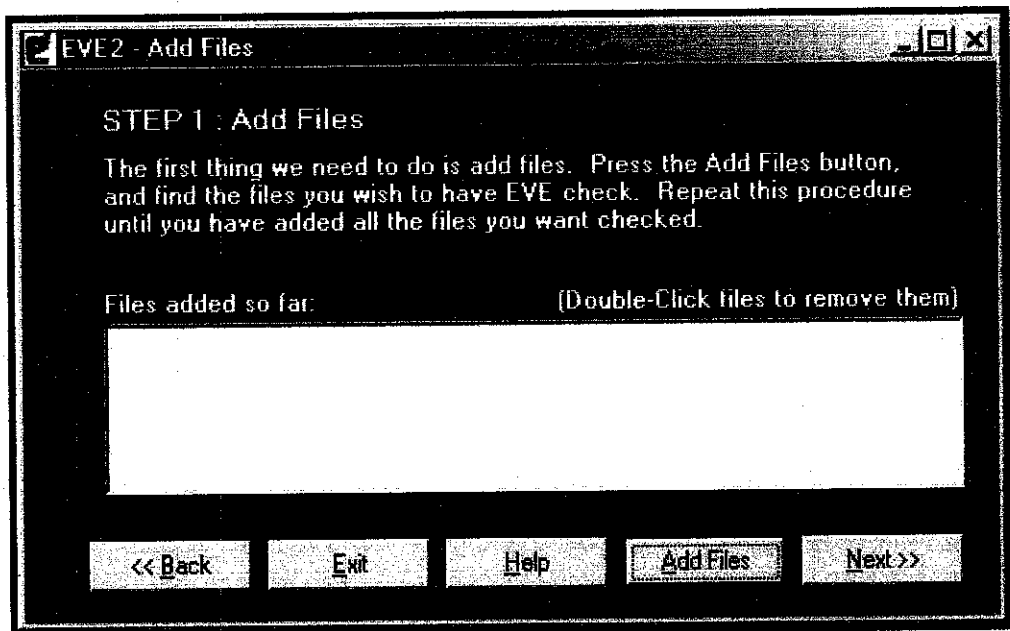


Figure 4: Essay Verification Engine

Based on the four acceptable and suitable software systems which are mentioned above; **(i) Dutch system Ephorus, (ii) Dupe Free Pro, (iii) WCopyFind and (iv) EVE2**, it can be concluded that each of the systems have their own advantages and limitations. The main reason for listing up all the existing plagiarism tools is basically for the sample application only. This is because *Prevention of Textual Plagiarism Application* has differentiated itself from other tools by providing the capability of multi-searching batches of files. This unique feature does not provided by any of the other four plagiarism software. That is why this project is needed to be developed since it will come out with its own way of detecting plagiarism as well as to minimize the lecturers' effort and make the application simple yet efficient.

CHAPTER 3

METHODOLOGY

3.1 System Methodology

3.1.1 System Development

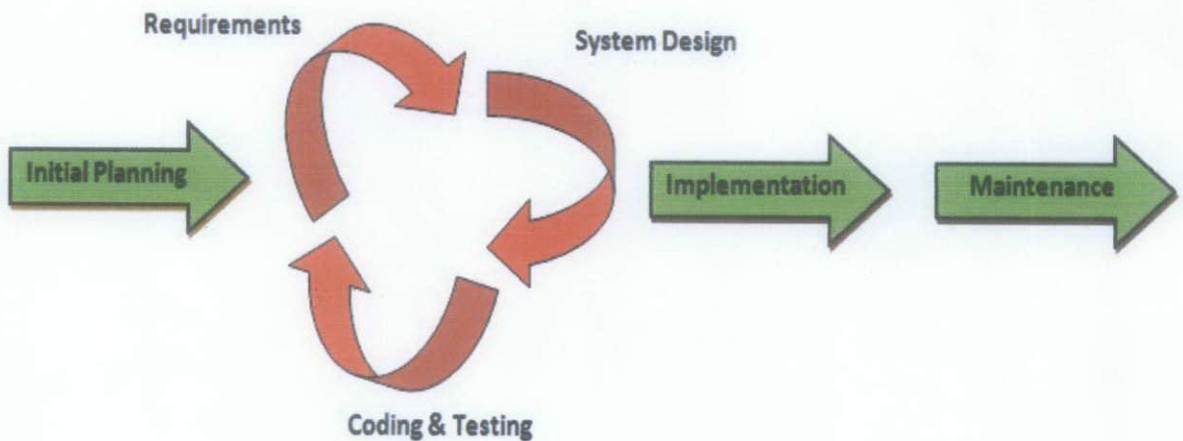


Figure 5: System Development

Iterative model has been implemented in this project as to come out with an efficient and effective application. The first phase which is the initial planning is important to start a system development. At this stage, all information has been gathered by studying the previous research paper, analyzing the significance of the project whether it is worth to be done or not as well as the time frame. All of the needed information is gathered as the reference in order to come out with the problem statement

and as general ideas to develop textual plagiarism detection system. The problem statement is derived from the current background of the advent and widespread use of the Internet which has made the occurrence of plagiarism flourishing excessively.

Instead of that, analyzing or looking into some existing plagiarism that already available in the market is also a very crucial part in this planning or investigation phase. This is as to look into the trend or pattern of the existing system as a sample application.

The next phase is requirements analysis. Requirement analysis is critical to the success of a development project. It encompasses those tasks that go into determining the need or conditions to meet for a new or altered product. Thus, at this phase, the requirement would be the system functions determination itself and also the method that will be used. Based on Figure 5, it shows that the requirement phase can be modified as there are changes.

At the next phase which is system design, it is the process of defining the architecture, components, modules, interfaces and data for a system. It is basically to satisfy specified requirements. In this project, the system design phase is mainly about the system flow processes and the functions such as system interface design are also determined.

Based on the system design, the system will then started to be developed by generating the codes and do some testing to make sure that it is working. The three phases which are requirement analysis, system design as well coding and testing actually can be redone if there is any changes that need to be made as long as it will meet the requirement and ready to be implemented.

Once the system is totally ready to be used by the users, it will move to the last phase which is the implementation. The system is still need to be kept updating as to maximize users' satisfaction and requirements.

3.1.2 Rabin - Karp Algorithm Methodology

Rabin – Karp algorithm is a string searching algorithm that uses hashing to find any one of a set of pattern string in a text. It has been chosen to be implemented in this project as it is a relatively fast string searching algorithm that works in $O(n)$ time on average. It is based on the use of hashing to compare strings.

Hashing can be used as a way of testing whether two strings are the same. If two strings hash the same value, then might be the same string, but if they hash to different values then they are definitely not the same. The Rabin – Karp algorithm compares string's hash values rather than the strings themselves. A hash value is which the numeric value that has been converted from every string. For example, a hash value for a string "hello" is 5. Thus, instead of comparing "hello" with "hello", it will compare with the hash value which is 5. The Rabin – Karp algorithm seeks to speed up the testing of equality of the pattern to substrings in the text by using a hash function. The Rabin – Karp exploits the fact that if two strings are equal, their hash values are also equal. Thus, by implementing this kind of algorithm, what it just has to do is computing the hash value of the substring that is searching for, and then look for a substring the same hash value.

The trick that makes the Rabin – Karp string searching algorithm efficient is the hash function itself. This is due to the easiness of computing for adjacent substring. Once the hash value of the first sub string in the next text string is calculated, the hash value for the next sub string can be calculated in $O(1)$. For example, assume that the pattern is of length 3 and each character in the pattern and the text string represents a number in base 10. The new converted text string now is "234567". According to Rabin – Karp algorithm, the first step would be to calculate the hash functions for the pattern as well as the first sub string. If the hash value is same, then the strings are compared to see, if it is indeed the same. If same, the shift is noted.

The Rabin – Karp algorithm uses a technique called fingerprinting.

1. Given the pattern of length n , and hash it.
2. The first n characters of the text string then will be calculated to get the hash value.
3. The hash value from [1] is compared against the value from [2]. This is to check whether the values are same or not. If not, then it is impossible for the two strings to be the same. However, if the hash value for both are same, then a normal string comparison need to be checked if they are actually the same string or they just hashed to the same value. If they match, means that the comparing is done. But if still not match, the process is still to be continued.
4. Then, shift over a character in the text string. This is a continuing process until the string is either found or it is actually reach to end of the text string.

The initial hashes are called fingerprints. However, there is a way that has been discovered by Rabin and Karp in order to update these fingerprints in constant time. In other words, to go from the hash of a substring in the text string to the next hash value only requires constant time. Below is a simple hash function on how does this works:

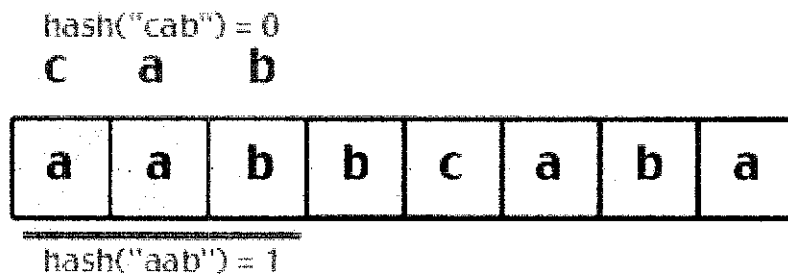


Figure 6: The initial fingerprints.

The hash value for "abc" is determined. $\text{Hash}("abc") = 0$. Then, the first three characters of the text string are hashed, and found that $\text{hash}("aab") = 1$. Both hash values are not same. Thus, continue checking by shifting over a character in the text string.

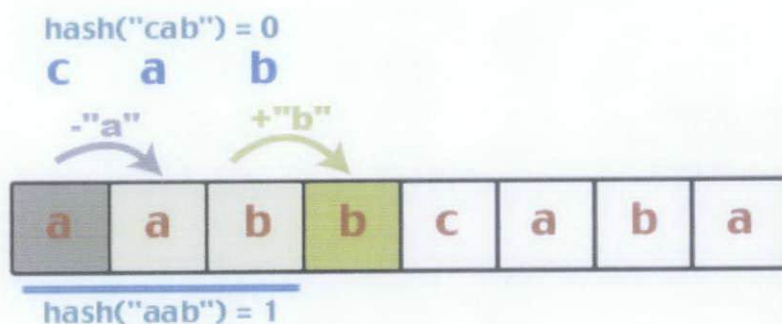


Figure 7: Updating the fingerprint

Hash function has provided some properties that able to update the fingerprint. The previous hash value of "aab" is equal to 1. The next character is "b" where it is added to this sum resulting in $1 + 1 = 2$. Then, the first character in the previous hash is "a" where it will be subtracted "a" from 2; $2 - 0 = 2$. So the modulo is; $2\%3 = 2$. This can be concluded that when sliding the window over, the next character that appears in the text string can just be added and deleted the first character that is now leaving the window.

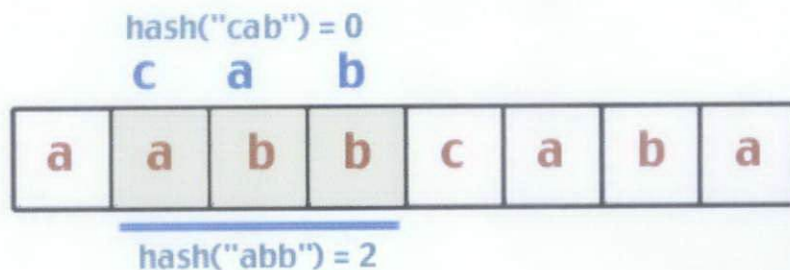


Figure 8: Second Comparison

The update is now completed and the next text to be matched now is "aab". However, the hash values are different; $\text{hash}(\text{"cab"}) \neq \text{hash}(\text{"aab"})$. So continue comparing with the next text.

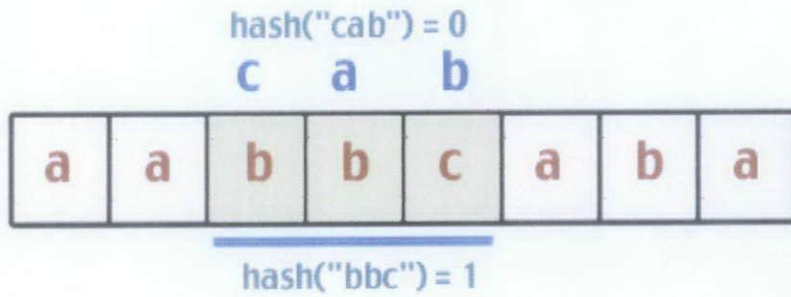


Figure 9: Third Comparison

This third comparison still has the different value. $\text{Hash}(\text{"cab"}) \neq \text{hash}(\text{"bbc"})$. So continue comparing with the next text.

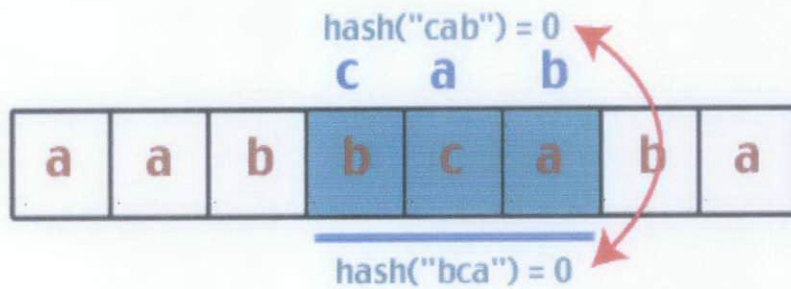


Figure 10: Fourth Comparison

These two hashes values are the same. Then, both string; "bca" and "cab" need to be compared normally whether they are same or not. However, they are not the same. So continue comparing with the next text.

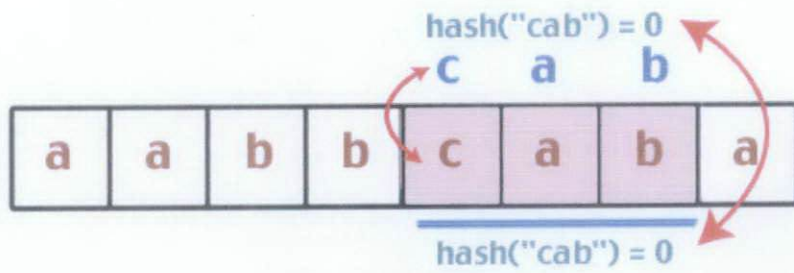


Figure 11: Fifth Comparison

These two hashes values are also the same. Both string are also the same “cab” = “cab”. Thus, this process is stop since the string is found.

3.2 Tools Requirement

The tools requirements are:

3.3.1 Hardware

- Computer

3.3.2 Software

- Netbeans IDE 7.0.1

3.3 Project Activities

- **Stage 1 – System Planning Phase**
 - Gantt chart Planning
 - Supervisor Meeting 1
 - System Interface Planning
 - Progress Update

- **Stage 2 – System Development Phase**
 - Project Requirement Gathering
 - User Interface Prototype
 - User Interface Development
 - Reporting Features
 - Usability Testing
 - Deploy

- **Stage 3 – Deliverable Preparation**
 - Progress Report Preparation
 - Literature Review Enhancement
 - Interface Enhancement
 - Documentation Enhancement
 - Supervisor Meeting
 - Viva Preparation

- **Stage 4 - Submission**
 - Progress Report Submission
 - Pre-EDX
 - Technical report
 - Project Submission

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Results

The first objective of this project which to perform a study on how to detect textual plagiarism has been met. Fact finding has been carried out to investigate related matters pertaining textual plagiarism.

4.1.1 System Overview

UML which is the acronym for Unified Modeling Language is a graphical visualization language. It is a most useful method of visualization and documenting software systems design. It can be used to create process diagrams as it consists of a series of symbols to represent graphically the various component and connectors within the systems. UML can be used for business processing modeling and requirements modeling. A use case diagram which is one types of the UML has been chosen for developing this application which named as *Prevention of Textual Plagiarism Application*.

4.1.1.1 Use Case Diagram

A use case is used to identify, clarify, and organize system requirements. The use case is basically made up of a set of possible sequences of interactions between system and users in a particular environment and related to particular goal. The use case is chosen as the system design for this project because it provides numbers of benefits.

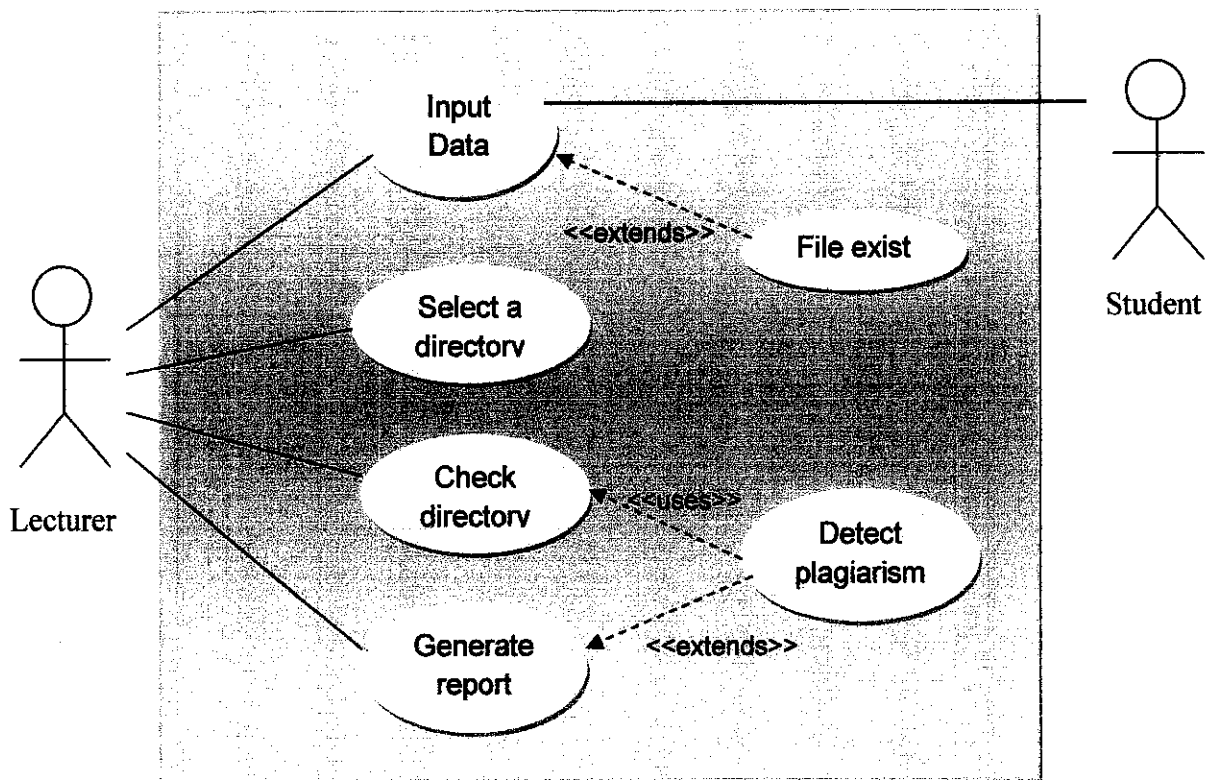


Figure 12: Use Case Diagram for Prevention of Textual Plagiarism Application

The use case diagram shown above is for *Prevention of Textual Plagiarism Application*. This use case involves Lecturer and Student actor. Lecturers use this system to detect plagiarism activity among their students. This use case diagram also involves extends *Input Data* and *Generate Report*. The *Input Data* is extended by the file exist because it provides some optional to the lecturers whether want to input the files or do nothing from the file submitted by the students.

The lecturer will then select a directory that he or she stores all the text documents submitted by students. This will minimize the lecturer's effort since he or she does not need to select and check for the documents one by one. All the documents in a directory will be checked for the similarities just at once. Then a report will be generated in a text form. This report will state the percentage of similarities against all the text documents in the directory. The benchmark for the percentage of similarities would be less than 50% for 'Not suspected as plagiarism' status whereas, more than 50% for 'Suspected as plagiarism' status.

4.1.2 Survey Result

In response to these requirements, two surveys have been conducted which are the first survey has been circulated previously over 40 of UTP students while the second survey is conducted among lecturers of UTP. The first survey is basically to gain the students' feedback about plagiarism and also to know the numbers of students committing in plagiarism activities.

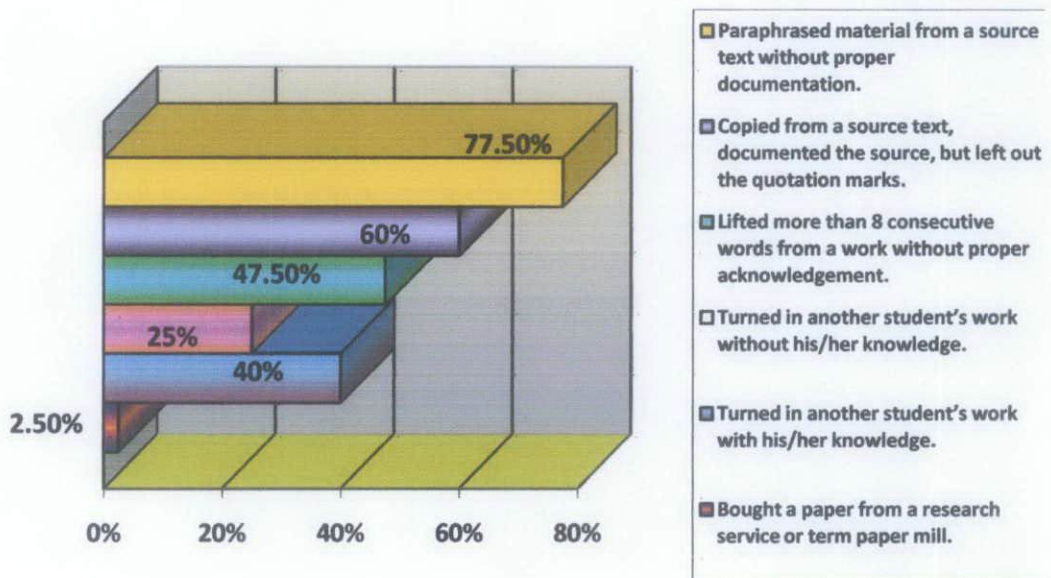


Figure 13: Methods of Plagiarizing

Based on the Figure 13 shown above, it tells that the highest way of plagiarizing been applied by the UTP students is through *paraphrasing material from a source text without proper documentation* with the percentage of 77.50%. On the other hand, *bought a paper from a research service or term paper mill* has recorded the lowest percentage which is 2.50%.

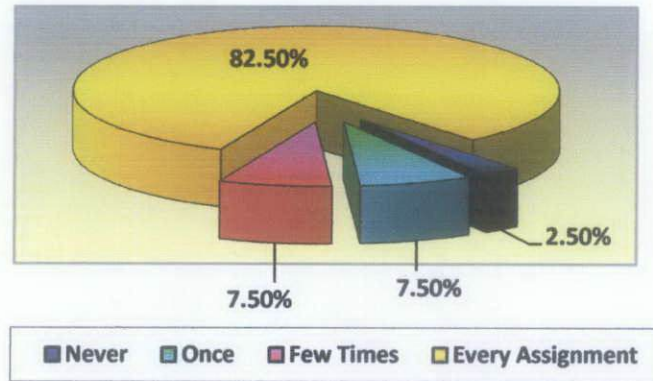


Figure 14: Frequency of Plagiarizing

Figure 14 has clearly shown that most of the UTP students have committed doing plagiarism a few times a semester with the percentage of 82.50%. This meant that even there is a lot of plagiarism detection tool available but students still keep cheating in completing their assignments or projects. Thus, this project will be developing as providing the new mechanism with the aim to reduce the numbers of students plagiarizing.

The second survey is implementing the System Usability Scale (SUS) which is a simple, ten-item scale giving a global view of subjective assessments of usability.

The average SUS score from all 10 lecturers of Universiti Teknologi PETRONAS is an 81. Based on the Graph 1 below that shows how the percentile ranks associate with SUS scores and letter grades, the SUS score of 81 can be interpreted as a grade of an -A. This meant that 'Prevention of Textual Plagiarism Application' is quite accepted by the lecturers of UTP and they said this system is usable then they would consider using it in the future after further enhancement.

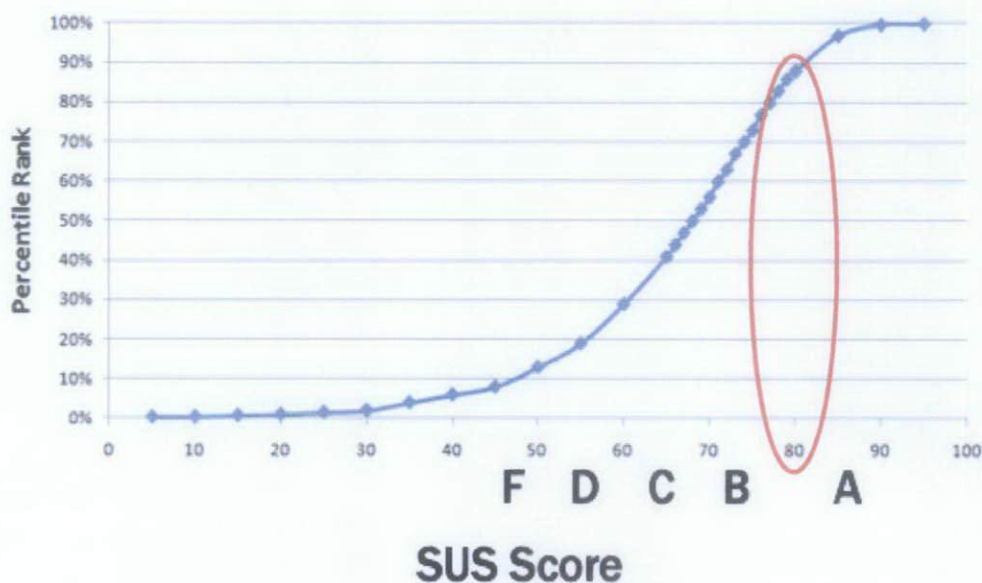


Figure 15: The percentile ranks associate with SUS scores and letter grades

4.2 System User Interface and Functions

4.2.1 Description on System User Interface

User interface for the Prevention of Textual Plagiarism Application has been designed and finalized. A splash screen is created as a welcome screen for using the system (Refer Appendices Figure 18). Then, a main form will appear which consist of two main buttons (Refer Appendices Figure 19). Those buttons are for selecting a directory and generating a report. There is also a menu tab on top of the windows form which consists of “File” and “Help” menu.

User just simply needs to select a directory that stores all of the text documents submitted by students. Those text documents must be in either .txt or .doc format since this application is providing the capability to check for those two formats only. By pressing the “Select a directory” button, a window will appear which require user to select a directory that he or she desired to be checked (Refer Appendices Figure 20).

Then, the “Generate a report” button which also functions as a check button will compare the similarities between those files and generate a report. That report will appear in a text form in the same window. It will be displaying the percentage of similarities between those files (Refer Appendices Figure 21). For example, 1113.doc is 55% similar with 1111.doc.

The user interface for this application is designing to be a very simple one. This is due to the aim of this application where it is mainly focusing on minimizing the lecturers’ effort in detecting plagiarism activities among students. Thus, it is crucial to ensure the application is efficient enough and not consuming much time.

4.2.2 System Functions

- **File menu:**
 - Consist of quit function. User can simply exit the system by using Ctrl + q.

- **Help menu:**
 - To give user an overview about what the system is all about

- **Select a directory button:**
 - To select a directory to be checked for plagiarism activities. All of the files in the directory will be compared against each other.

- **Generate a report button:**
 - To view the percentage of similarities among those compared files. The generated report is viewed in a text form.

CHAPTER 5

RECOMMENDATIONS

There are some recommendations that have been identified and listed as a further improvement in developing this plagiarism detection tool in the future. The enhancement should be in terms of:

i. **Reporting features:**

Based on the SUS survey, most of the lecturers are suggesting for further enhancement in the reporting part. They would consider the system if effective enough if it will be able to rank the percentage of similarities on the most high percentage basis. Any files that have high percentage of similarities will be ranked on top and highlighted with some colors. Thus, it will more likely ease the lecturers in detecting for plagiarism activities among students.

ii. **Online accessibility:**

This system should be made accessible through web. This project is just tending to be developed as a PC installation as it is providing the capability of multi-searching batches of files each time. Moreover, PC installation also provides the quality and speed which does not depend on the network connection. However, by making this system accessible via online, it will enable the system to compare the

submitted files with sources from the Internet. Moreover, this can also give easy access to the authenticated users.

iii. Ability to cater for any types of files

For this time being, this project is just mainly focus in handling with the .doc and .txt files. In order to widen its functionality, it should be able to cater for numerous formats of files such as PDF, HTML, .rtf, Open Office and other types of files.

CHAPTER 6

CONCLUSION

As a conclusion, the development of plagiarism detection tool is basically to reduce the plagiarism activities among students. This vital goal should be achieved successfully in order to produce high quality graduates that submitted their projects or assignments which meet the education quality standard.

Thus, in order to ensure that the vital goal is able to be achieved successfully, this Final Year Project 2's concern to accomplish the previous mentioned objectives. All of the listed objectives are to enable the lecturers to detect the similarities of students' assignments by using *Prevention of Textual Plagiarism Application*, to identify any plagiarism activities in a better manner by checking all the documents against the material submitted by students at once, to generate an effective way of reporting features when plagiarism is detected by displaying the percentage of similarities between those documents.

Based on those listed objectives that have been mentioned above, this project is absolutely relevant to be developed since it is obviously will give advantages to the lecturers and university as a whole. This is because this tool will help in developing well rounded students who are creative and innovative with a potential to become a leaders of industry and the nation. Besides that, this project is also feasible and practicable to be developed as the method, equipments and the budget is possible and reasonable.

REFERENCES

- [1] Roberts, T. S. (2008). *Student Plagiarism in an Online World - Problems and Solutions*. United States of America: Information Science References.
- [2] Smith, W. S. (2008). *Plagiarism, the Internet and Student Learning*. New York: Routledge.
- [3] Birchard, K. (2006). "The Chronicle of Higher Education" *Cheating is rampant at Canadian Colleges*, 53(8).
- [4] University of Melbourne. (2011). *Academic Honesty and Plagiarism*. Retrieved February 20, 2011, from <http://www.academichonesty.unimelb.edu.au/plagiarism.html>
- [5] University of Oxford. (2011). *Plagiarism: Educational policy and standards*. Retrieved February 20, 2011, from <http://www.admin.ox.ac.uk/epsc/plagiarism/>
- [6] Universiti Teknologi Mara. (2011). *Institute of Quality & Knowledge Advancement: Student Survey on Plagiarism*. Retrieved February 20, 2011, from <http://inqka.uitm.edu.my/index.php/component/content/article/123-plagiarism/189-student-survey-on-plagiarism>
- [7] Clough, P. (2000). *Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies*, 30[4].
- [8] Maurer H, Kappe F, Zaka B. (2006). "Plagiarism – A Survey" *Journal of Universal Computer Science*, 12 [8].

- [9] Chun, K.S., Kuok, S.W., & Wei, L.W. (2008). "Text plagiarism detection method based on path patterns" *International Journal of Business Intelligence and Data Mining*, 3 [2].
- [10] White, DR., & Joy, MS. (2004). "Sentence-based natural language plagiarism detection" *Journal on Educational Resources in Computing (JERIC)*, 4 [2].
- [11] Bilic-Zulle, L., Azman, J., Frkovic, V., & Petrovecki, M. (2008). "Is there an effective approach to deterring students from plagiarizing?" *Science and Engineering Ethics*, 14 [1], 139-147.
- [12] Long, T.C., Errami, M. & George, A.C. (2009). "Responding to Possible Plagiarism" *Science*, 323, 1293-1294.
- [13] Lippi, G., & Favaloro, E.J. (2008). "Detection of duplicates and redundancies. A major responsibility of peer-reviewers?" *Clinical Chemistry and Laboratory Medicine*, 46, 1796-1797.
- [14] Debora, W.W. [2008]. *On the Utility of Plagiarism Detection Software*. Retrieved 1 March 2011 from <http://plagiat.htw-berlin.de/software/>
- [15] *Ephorus Plagiarism Control* (2010). Retrieved February 29, 2011 from <http://fronter.info/downloads/Ephorus.pdf>
- [16] Harris, R. (2009). *Anti-plagiarism strategies for research papers*. Retrieved February 20, 2010 from <http://www.virtualsalt.com/antiplag.htm>
- [17] Michael., & Steven, Grzywacz. (2011). *Dupe Free Pro Software*. Retrieved March 1, 2011 from <http://www.dudefreepro.com/>

- [18] Louis, A. Bloomfield. (2010). Software to detect plagiarism: WCopyFind. Retrieved February 25, 2011 from <http://plagiarism.phys.virginia.edu/Wsoftware.html>
- [19] *EVE Plagiarism Detection System* (2000). Retrieved February 25, 2011 from <http://www.canexus.com/>
- [20] Kumari, A. (2008). *An Introduction to Research Methodology*. India: Agrotech Publishing Company.

ID	Task Name	Duration	Start	Finish
1	Prevention of Plagiarism using Pop Up Notification	63 days	Wed 26/1/11	Fri 22/4/11
2	Stage 1 - Proposal & Approval	8 days	Wed 26/1/11	Mon 7/2/11
3	FYP Mass Lecture	1 day	Wed 26/1/11	Wed 26/1/11
4	Choose topic	1 wk	Thu 27/1/11	Wed 2/2/11
5	Met and discussed with SV on chosen topic	1 day	Thu 3/2/11	Thu 3/2/11
6	Submit Project Proposal	0 days	Mon 7/2/11	Mon 7/2/11
7	Stage 2 - Extended Proposal	19 days	Tue 8/2/11	Fri 4/3/11
8	Supervisor Consultation	1 day	Tue 8/2/11	Tue 8/2/11
9	Research Class #1	1 day	Wed 16/2/11	Wed 16/2/11
10	Conduct research	2 wks	Fri 18/2/11	Thu 3/3/11
11	Fact findings for literature review	2 wks	Fri 18/2/11	Thu 3/3/11
12	Supervisor Consultation	1 day	Fri 4/3/11	Fri 4/3/11
13	Research Class #2	1 day	Wed 2/3/11	Wed 2/3/11
14	Submission of Extended Proposal	0 days	Fri 4/3/11	Fri 4/3/11
15	Stage 3 - Viva : Proposal Defense	15 days	Mon 7/3/11	Fri 25/3/11
16	Information Gathering	1 wk	Mon 7/3/11	Fri 11/3/11
17	Progress Report	3 days	Mon 14/3/11	Wed 16/3/11
18	Supervisor Consultation	1 day	Thu 17/3/11	Thu 17/3/11
19	Proposal Defense Preparation	1 wk	Fri 18/3/11	Thu 24/3/11
20	Proposal Defense Presentation	0 days	Fri 25/3/11	Fri 25/3/11
21	Stage 4 - Interim Report	10 days	Mon 28/3/11	Fri 8/4/11
22	Further Research	2 wks	Mon 28/3/11	Fri 8/4/11
23	Supervisor Consultation	1 day	Thu 31/3/11	Thu 31/3/11
24	Documentation - Interim Report	1 wk	Fri 1/4/11	Thu 7/4/11
25	Stage 5 - Technical Report	11 days	Fri 8/4/11	Fri 22/4/11
26	Supervisor Consultation	1 day	Fri 8/4/11	Fri 8/4/11
27	Further Research	2 wks	Mon 11/4/11	Fri 22/4/11
28	Documentation - Technical Report	2 wks	Mon 11/4/11	Fri 22/4/11

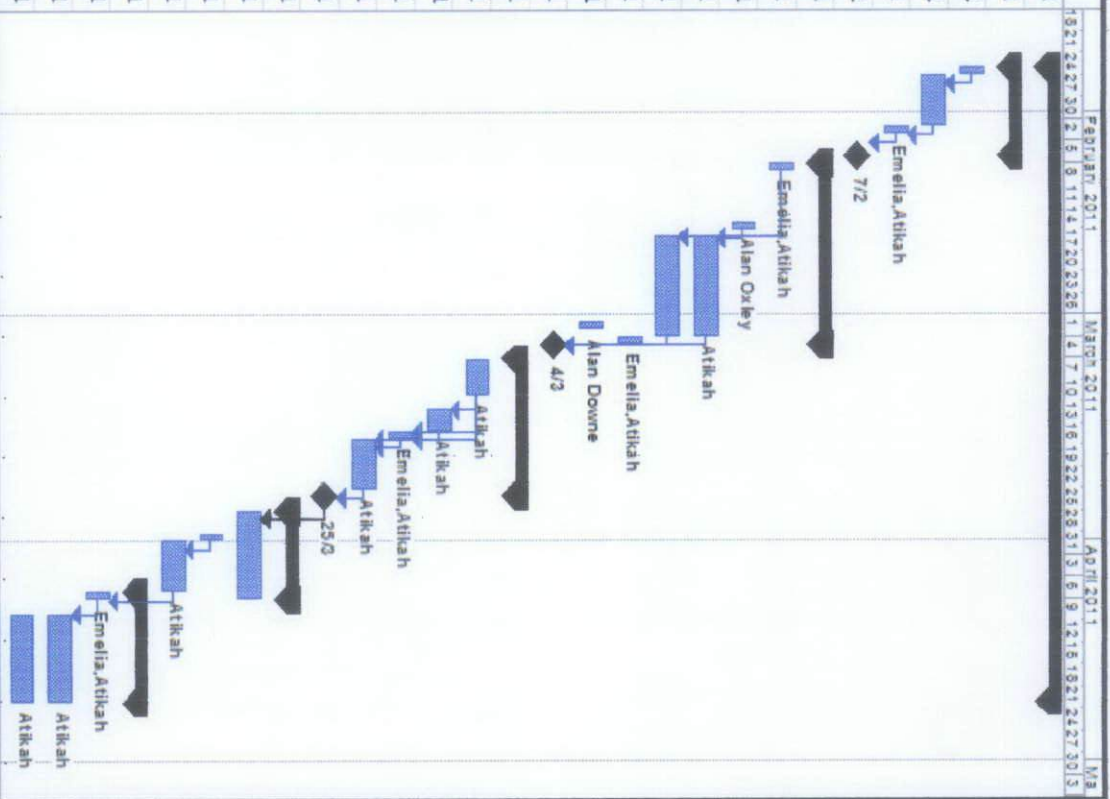


Figure 16: Timeline for FYP Part 1

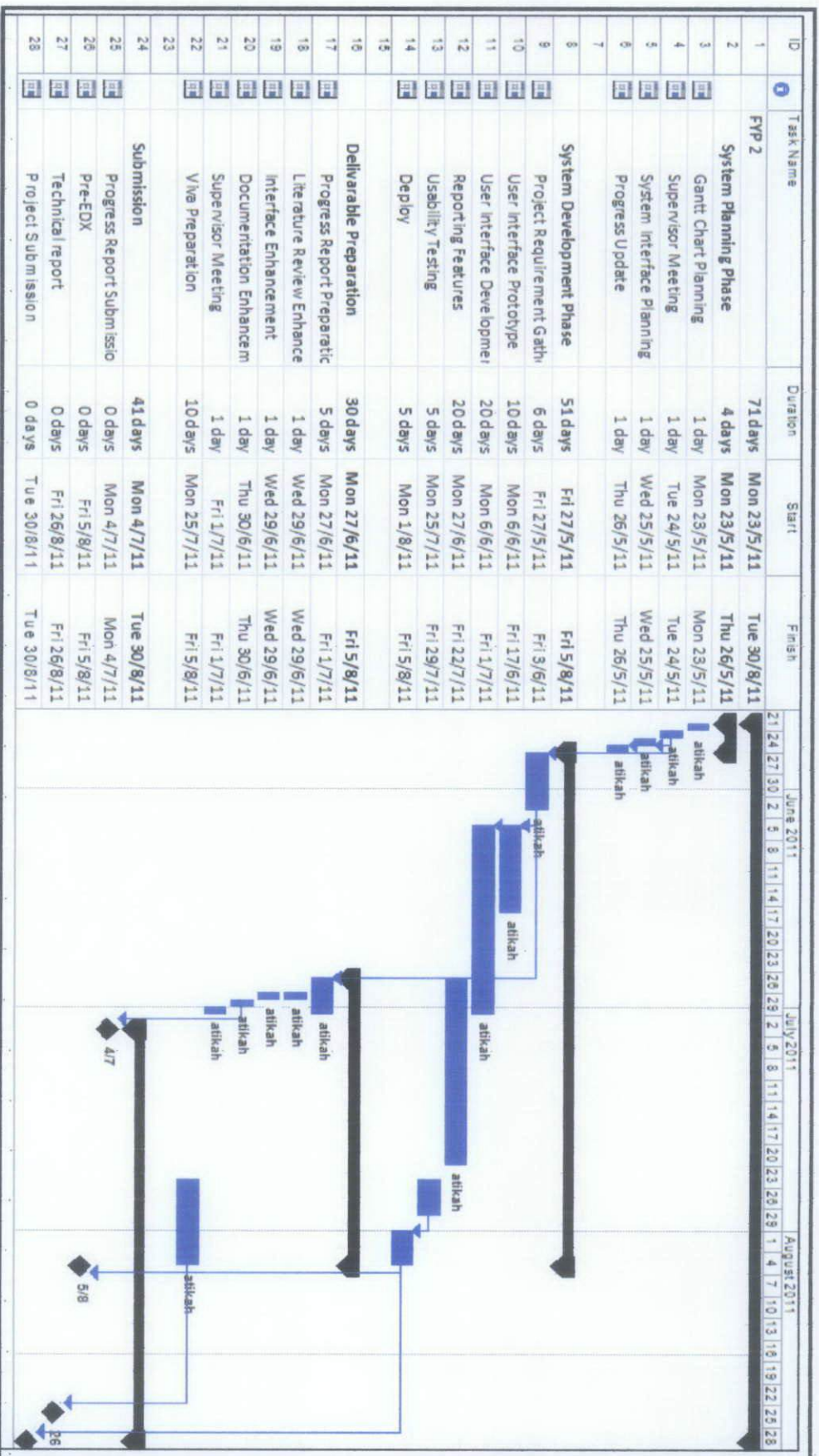


Figure 17: Timeline for FYP Part 2



Figure 18: Splash Screen

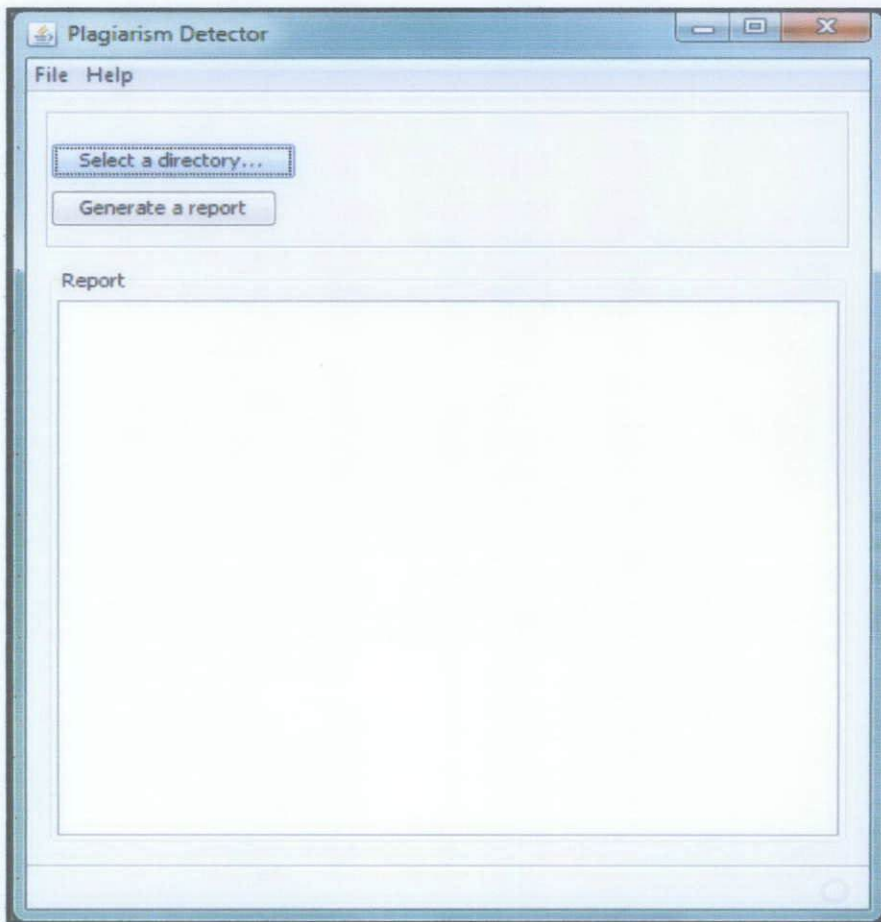


Figure 19: Main Window Form

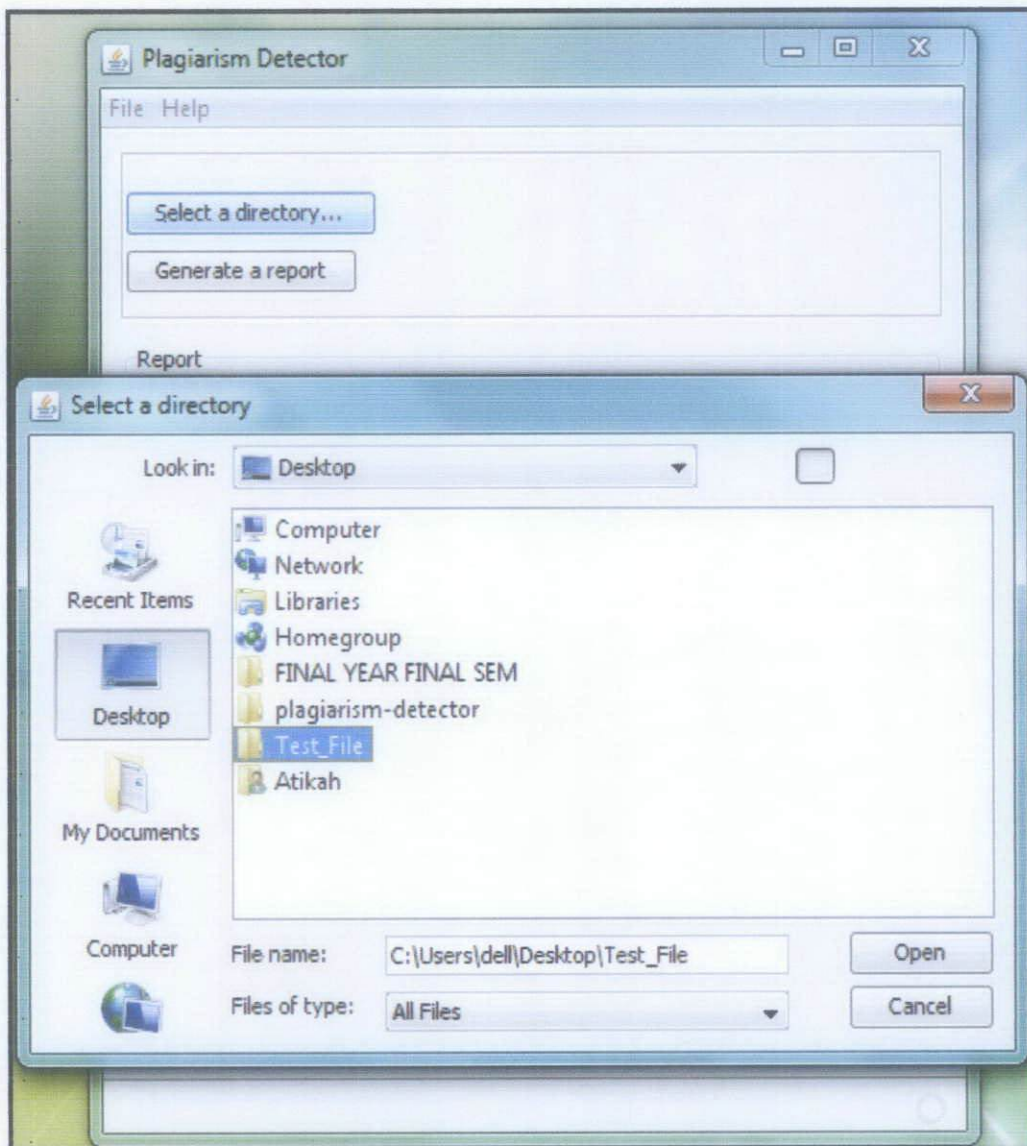


Figure 20: Select a directory

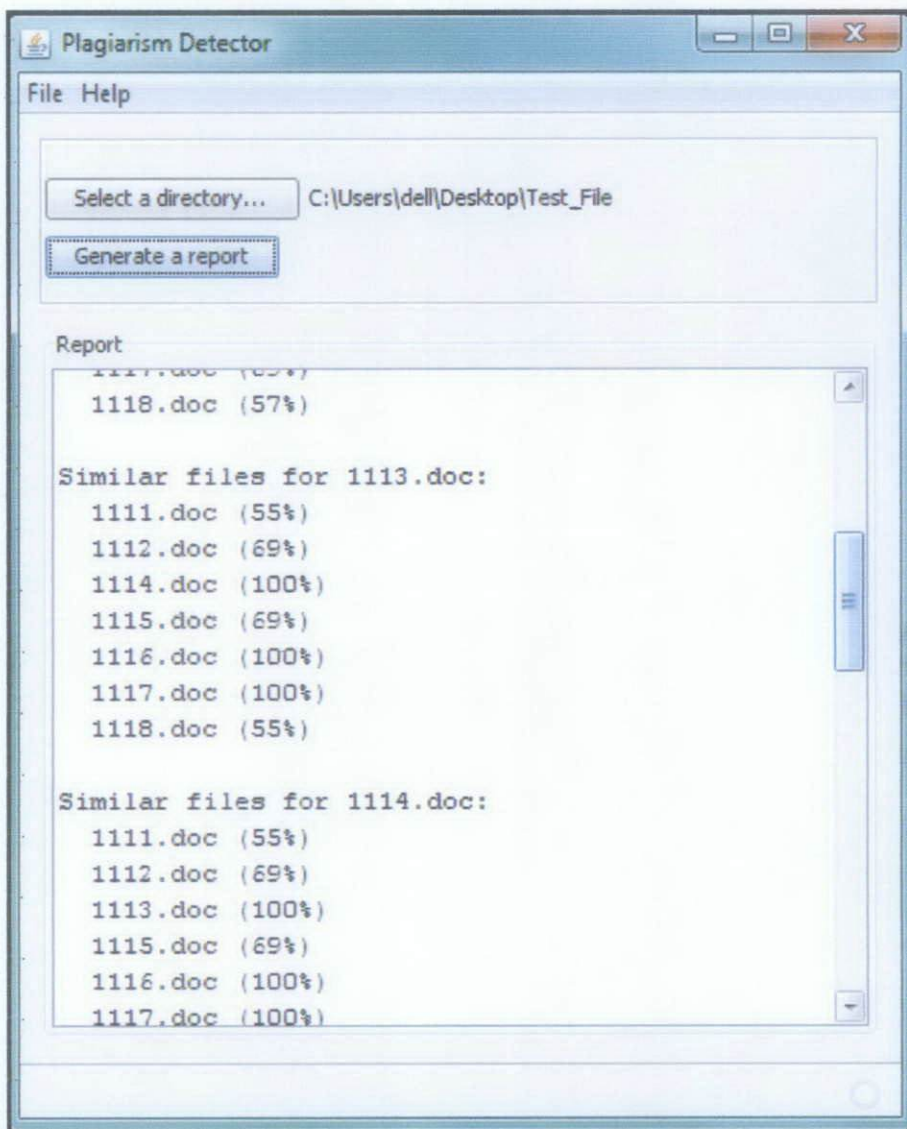


Figure 21: Generate a report



Figure 22: About Form

```

/*
 * PlagiarismDetectorApp.java
 */

package plagiarismdetector;

import org.jdesktop.application.Application;
import org.jdesktop.application.SingleFrameApplication;

/**
 * The main class of the application.
 */
public class PlagiarismDetectorApp extends SingleFrameApplication {

    /**
     * At startup create and show the main frame of the application.
     */
    @Override protected void startup() {
        show(new PlagiarismDetectorView(this));
    }

    /**
     * This method is to initialize the specified window by injecting resources.
     * Windows shown in our application come fully initialized from the GUI
     * builder, so this additional configuration is not needed.
     */
    @Override protected void configureWindow(java.awt.Window root) {
    }

    /**
     * A convenient static getter for the application instance.
     * @return the instance of PlagiarismDetectorApp
     */
    public static PlagiarismDetectorApp getApplication() {
        return Application.getInstance(PlagiarismDetectorApp.class);
    }

    /**
     * Main method launching the application.
     */
    public static void main(String[] args) {
        launch(PlagiarismDetectorApp.class, args);
    }
}

```

```

/*
 * PlagiarismDetectorView.java
 */

package plagiarismdetector;

import java.util.logging.Level;
import java.util.logging.Logger;
import org.jdesktop.application.Action;
import org.jdesktop.application.ResourceMap;
import org.jdesktop.application.SingleFrameApplication;
import org.jdesktop.application.FrameView;
import org.jdesktop.application.TaskMonitor;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import javax.swing.Timer;
import javax.swing.Icon;
import javax.swing.JDialog;
import javax.swing.JFrame;

/**
 * The application's main frame.
 */
public class PlagiarismDetectorView extends FrameView {

    public PlagiarismDetectorView(SingleFrameApplication app) {
        super(app);

        initComponents();

        initSsdeep();

        // status bar initialization - message timeout, idle icon and busy animation, etc
        ResourceMap resourceMap = getResourceMap();
        int messageTimeout = resourceMap.getInteger("StatusBar.messageTimeout");
        messageTimer = new Timer(messageTimeout, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                statusBarMessageLabel.setText("");
            }
        });
    }

```



```

    }
});
messageTimer.setRepeats(false);
int busyAnimationRate = resourceMap.getInteger("StatusBar.busyAnimationRate");
for (int i = 0; i < busyIcons.length; i++) {
    busyIcons[i] = resourceMap.getIcon("StatusBar.busyIcons[" + i + "]");
}
busyIconTimer = new Timer(busyAnimationRate, new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        busyIconIndex = (busyIconIndex + 1) % busyIcons.length;
        statusAnimationLabel.setIcon(busyIcons[busyIconIndex]);
    }
});
idleIcon = resourceMap.getIcon("StatusBar.idleIcon");
statusAnimationLabel.setIcon(idleIcon);
progressBar.setVisible(false);

// connecting action tasks to status bar via TaskMonitor
TaskMonitor taskMonitor = new TaskMonitor(getApplication().getContext());
taskMonitor.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
    public void propertyChange(java.beans.PropertyChangeEvent evt) {
        String propertyName = evt.getPropertyName();
        if ("started".equals(propertyName)) {
            if (!busyIconTimer.isRunning()) {
                statusAnimationLabel.setIcon(busyIcons[0]);
                busyIconIndex = 0;
                busyIconTimer.start();
            }
            progressBar.setVisible(true);
            progressBar.setIndeterminate(true);
        } else if ("done".equals(propertyName)) {
            busyIconTimer.stop();
            statusAnimationLabel.setIcon(idleIcon);
            progressBar.setVisible(false);
            progressBar.setValue(0);
        } else if ("message".equals(propertyName)) {
            String text = (String)(evt.getNewValue());
            statusMessageLabel.setText((text == null) ? "" : text);
            messageTimer.restart();
        } else if ("progress".equals(propertyName)) {
            int value = (Integer)(evt.getNewValue());
            progressBar.setVisible(true);

```

```

        progressBar.setIndeterminate(false);
        progressBar.setValue(value);
    }
}
});
}

@Action
public void showAboutBox() {
    if (aboutBox == null) {
        JFrame mainFrame = PlagiarismDetectorApp.getApplication().getMainFrame();
        aboutBox = new PlagiarismDetectorAboutBox(mainFrame);
        aboutBox.setLocationRelativeTo(mainFrame);
    }
    PlagiarismDetectorApp.getApplication().show(aboutBox);
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    mainPanel = new javax.swing.JPanel();
    jPanel1 = new javax.swing.JPanel();
    btnDir = new javax.swing.JButton();
    btnReport = new javax.swing.JButton();
    labelCurrentDir = new javax.swing.JLabel();
    jPanel2 = new javax.swing.JPanel();
    jScrollPane1 = new javax.swing.JScrollPane();
    txtAreaReport = new javax.swing.JTextArea();
    menuBar = new javax.swing.JMenuBar();
    javax.swing.JMenu fileMenu = new javax.swing.JMenu();
    javax.swing.JMenuItem exitMenuItem = new javax.swing.JMenuItem();
    javax.swing.JMenu helpMenu = new javax.swing.JMenu();
    javax.swing.JMenuItem aboutMenuItem = new javax.swing.JMenuItem();
    statusPanel = new javax.swing.JPanel();
    javax.swing.JSeparator statusPanelSeparator = new javax.swing.JSeparator();
    statusMessageLabel = new javax.swing.JLabel();

```



```

        .add(btnDir)
        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
        .add(labelCurrentDir))
        .add(btnReport))
        .addContainerGap(323, Short.MAX_VALUE))
    );
    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(jPanel1Layout.createSequentialGroup()
            .add(17, 17, 17)

.add(jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
        .add(btnDir)
        .add(labelCurrentDir))
        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
        .add(btnReport)
        .addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );

jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder(resourceMap.getString("jPane
l2.border.title"))); // NOI18N
    jPanel2.setName("jPanel2"); // NOI18N

    jScrollPane1.setName("jScrollPane1"); // NOI18N

    txtAreaReport.setColumns(20);
    txtAreaReport.setRows(15);
    txtAreaReport.setName("txtAreaReport"); // NOI18N
    jScrollPane1.setViewportView(txtAreaReport);

    org.jdesktop.layout.GroupLayout jPanel2Layout = new
org.jdesktop.layout.GroupLayout(jPanel2);
    jPanel2.setLayout(jPanel2Layout);
    jPanel2Layout.setHorizontalGroup(
        jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(jScrollPane1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 448,
Short.MAX_VALUE)
    );
    jPanel2Layout.setVerticalGroup(
        jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)

```

```

        .add(org.jdesktop.layout.GroupLayout.TRAILING, jPanel1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 449, Short.MAX_VALUE)
    );

    org.jdesktop.layout.GroupLayout mainPanelLayout = new
org.jdesktop.layout.GroupLayout(mainPanel);
    mainPanel.setLayout(mainPanelLayout);
    mainPanelLayout.setHorizontalGroup(
        mainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING,
mainPanelLayout.createSequentialGroup()
                .addContainerGap()

.add(mainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
            .add(org.jdesktop.layout.GroupLayout.LEADING, jPanel1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .add(org.jdesktop.layout.GroupLayout.LEADING, jPanel2,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap())
        );
    mainPanelLayout.setVerticalGroup(
        mainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(mainPanelLayout.createSequentialGroup()
                .addContainerGap()
                .add(jPanel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
                .add(jPanel2, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );

    menuBar.setName("menuBar"); // NOI18N

    fileMenu.setText(resourceMap.getString("fileMenu.text")); // NOI18N
    fileMenu.setName("fileMenu"); // NOI18N

    exitMenuItem.setAction(actionMap.get("quit")); // NOI18N
    exitMenuItem.setName("exitMenuItem"); // NOI18N

```

```

fileMenu.add(exitMenuItem);

menuBar.add(fileMenu);

helpMenu.setText(resourceMap.getString("helpMenu.text")); // NOI18N
helpMenu.setName("helpMenu"); // NOI18N

aboutMenuItem.setAction(actionMap.get("showAboutBox")); // NOI18N
aboutMenuItem.setName("aboutMenuItem"); // NOI18N
helpMenu.add(aboutMenuItem);

menuBar.add(helpMenu);

statusPanel.setName("statusPanel"); // NOI18N

statusPanelSeparator.setName("statusPanelSeparator"); // NOI18N

statusMessageLabel.setName("statusMessageLabel"); // NOI18N

statusAnimationLabel.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
statusAnimationLabel.setName("statusAnimationLabel"); // NOI18N

progressBar.setName("progressBar"); // NOI18N

org.jdesktop.layout.GroupLayout statusPanelLayout = new
org.jdesktop.layout.GroupLayout(statusPanel);
statusPanel.setLayout(statusPanelLayout);
statusPanelLayout.setHorizontalGroup(
    statusPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(statusPanelSeparator, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 480,
Short.MAX_VALUE)
        .add(statusPanelLayout.createSequentialGroup()
            .addContainerGap()
            .add(statusMessageLabel)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 310,
Short.MAX_VALUE)
            .add(progressBar, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(statusAnimationLabel)
            .addContainerGap())

```

```

    );
    statusPanelLayout.setVerticalGroup(
        statusPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(statusPanelLayout.createSequentialGroup()
            .add(statusPanelSeparator, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.add(statusPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
        .add(statusMessageLabel)
        .add(statusAnimationLabel)
        .add(progressBar, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
        .add(3, 3, 3))
    );

    jFileChooser1.setDialogTitle(resourceMap.getString("jFileChooser1.dialogTitle")); //
NOI18N
    jFileChooser1.setSelectionMode(javax.swing.JFileChooser.DIRECTORIES_ONLY);
    jFileChooser1.setName("jFileChooser1"); // NOI18N

    setComponent(mainPanel);
    setMenuBar(menuBar);
    setStatusBar(statusPanel);
} // </editor-fold>

private void btnDirMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
}

@Action
public void selectDir() {
    int returnVal = jFileChooser1.showOpenDialog(this.getComponent());
    if (returnVal == jFileChooser1.APPROVE_OPTION) {
        File file = jFileChooser1.getSelectedFile();

        mCurrentDir = file.getAbsolutePath();

        labelCurrentDir.setText(mCurrentDir);
        btnReport.setEnabled(true);
    }
}

```

```

    } else {
        System.out.println("File access cancelled by user.");
    }
}

@Action
public void generateReport() {
    txtAreaReport.setText("");

    if (mCurrentDir != null) {
        try {
            mSsdeep.hashAllInDir(mCurrentDir);
        } catch (IOException ex) {
            Logger.getLogger(PlagiarismDetectorView.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (InterruptedException ex) {
            Logger.getLogger(PlagiarismDetectorView.class.getName()).log(Level.SEVERE,
null, ex);
        }

        String[] files = mSsdeep.getFileList(mCurrentDir);
        for (String file : files) {
            if (file.endsWith(SsdeepWrapper.HASH_FILE_NAME)) continue;
            generateReportForFile(file);
        }
    }
}

private void generateReportForFile(String file) {
    ArrayList<String> similarFiles = null;

    try {
        similarFiles = mSsdeep.similarFilesFor(mCurrentDir, file);
    } catch (IOException ex) {
        Logger.getLogger(PlagiarismDetectorView.class.getName()).log(Level.SEVERE, null,
ex);
    } catch (InterruptedException ex) {
        Logger.getLogger(PlagiarismDetectorView.class.getName()).log(Level.SEVERE, null,
ex);
    }

    txtAreaReport.append("Similar files for " + file + ":\n");
}

```



```

    if (similarFiles != null) {
        for (String similarFile : similarFiles) {
            txtAreaReport.append(" " + similarFile + "\n");
        }
    }

    txtAreaReport.append("\n");
}

// Variables declaration - do not modify
private javax.swing.JButton btnDir;
private javax.swing.JButton btnReport;
private javax.swing.JFileChooser jFileChooser1;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JLabel labelCurrentDir;
private javax.swing.JPanel mainPanel;
private javax.swing.JMenuBar menuBar;
private javax.swing.JProgressBar progressBar;
private javax.swing.JLabel statusAnimationLabel;
private javax.swing.JLabel statusMessageLabel;
private javax.swing.JPanel statusPanel;
private javax.swing.JTextArea txtAreaReport;
// End of variables declaration

private final Timer messageTimer;
private final Timer busyIconTimer;
private final Icon idleIcon;
private final Icon[] busyIcons = new Icon[15];
private int busyIconIndex = 0;

private JDialog aboutBox;

private String mCurrentDir;

private SsdeepWrapper mSsdeep;

private void initSsdeep() {
    final String bin = File.separator + "bin" + File.separator + "ssdeep.exe";

```

```

    try {
        mSsdeep = new SsdeepWrapper(System.getProperty("user.dir") + bin);
    } catch(IllegalArgumentException ex) {
        mSsdeep = new SsdeepWrapper(System.getProperty("user.dir") + File.separator + "dist"
+ bin);
    }
}
}
}

```

```

package plagiarismdetector;

```

```

/**

```

```

 *

```

```

 * @author atikah

```

```

 */

```

```

import java.io.BufferedReader;

```

```

import java.io.File;

```

```

import java.io.FileWriter;

```

```

import java.io.IOException;

```

```

import java.io.InputStreamReader;

```

```

import java.util.ArrayList;

```

```

public class SsdeepWrapper {

```

```

    private String mBinPath;

```

```

    private final static String sSsdeepHeader = "ssdeep,1.1--
blocksize:hash:hash,filename\n";

```

```

    public final static String HASH_FILE_NAME = "hashes.txt";

```

```

    private File mHashFile;

```

```

    private String[] mFileList;

```

```

    private final static Runtime sRuntime = Runtime.getRuntime();

```

```

        public SsdeepWrapper(String ssdeepBinPath) {
File ssdeepBinFile = new File(ssdeepBinPath);

if (!ssdeepBinFile.exists()) {
    throw new IllegalArgumentException("Invalid ssdeep binary path given");
}

        mBinPath = ssdeepBinPath;
    }

    public void hashAllInDir(String dir) throws IOException, InterruptedException {
final String[] cmd = {"", ""};
String line = "";
Process pr = null;
BufferedReader buf = null;

mHashFile = createHashFile(dir);
final FileWriter writer = new FileWriter(mHashFile);
writer.write(sSsdeepHeader);

final String[] files = getFileList(dir);
for (String file : files) {
    cmd[0] = mBinPath;
    cmd[1] = dir + File.separator + file;
    pr = sRuntime.exec(cmd);
}
}

```

```

        if (file.endsWith(HASH_FILE_NAME)) continue;
        if (pr != null) pr.waitFor();

        buf = new BufferedReader(new InputStreamReader(pr.getInputStream()));

        while((line = buf.readLine()) != null) {
            if (!line.startsWith("ssdeep,") {
                System.out.println(line);
                writer.write(line + "\n");
            }
        }
    }
    writer.flush();
    writer.close();
}

    public String[] getFileList(String dir) {
        final File file = new File(dir);

        return file.list();
    }

    public ArrayList<String> similarFilesFor(String fileDir, String fileName) throws
    IOException, InterruptedException {
        final String[] cmd = {
            mBinPath,
            "-m",

```

```

        fileDir + File.separator + HASH_FILE_NAME,
        fileDir + File.separator + fileName    };

    final Process pr = sRuntime.exec(cmd);
    ArrayList<String> similarFiles = new ArrayList<String>(5);

    // if (pr != null) pr.waitFor();

    final BufferedReader buf = new BufferedReader(new
InputStreamReader(pr.getInputStream()));

    String line = "";

    while((line = buf.readLine()) != null) {
        final String similarFileName = processSsdeepLineOutput(line, fileDir,
fileName);
        if (similarFileName != null) similarFiles.add(similarFileName);
    }

    return similarFiles;
}

private String processSsdeepLineOutput(final String line, final String fileDir, final String
fileName) {
    final int start = line.lastIndexOf(HASH_FILE_NAME + ":",) +
HASH_FILE_NAME.length() + 1;

    final int fileDirLen = fileDir.length() + 1;

    if (start != -1) {
        final String path = line.substring(start + 1);

```

```

final String newLine = path.substring(path.indexOf(fileDir) + fileDirLen);
final String lineFileName = newLine.substring(0, newLine.lastIndexOf("/") - 1);

if (!lineFileName.equals(fileName)) {
    return (newLine.substring(0, newLine.length() - 1) + "%");
}
}
return null;
}

private File createHashFile(String dir) throws IOException {
    mHashFile = new File(dir + File.separator + HASH_FILE_NAME);
    if (mHashFile.exists()) {
        if (mHashFile.canWrite()) {
            mHashFile.delete();
        } else {
            throw new IOException("Could not write the hash file: " +
HASH_FILE_NAME);
        }
    }
    mHashFile.createNewFile();
    return mHashFile;
}

public String getReport() {
    return null;
}
}
}

```