



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A
FINAL YEAR PROJECT REPORT
ON
RAID ON CODE PIRATE (ROCOP): A Plagiarism Detection System

[EG777CT]

By:

Kailash Budhathoki (75012)

Rakesh Manandhar (75026)

Shilpa Singhal (75033)

A PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND
COMPUTER ENGINEERING IN PARTIAL FULLFILMENT OF THE REQUIREMENT
FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINNERING
LALITPUR, NEPAL

JANUARY, 2011

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance, a project report entitled "Raid On Code Pirate (ROCOP): A Plagiarism Detection System" submitted by Kailash Budhathoki, Rakesh Manandhar and Shilpa Singhal in partial fulfilment of the requirements for the Bachelor's degree in Computer Engineering.

Supervisor, Mr. Daya Sagar Baral

Lecturer / Director CIT

Department of Electronics and Computer Engineering, Pulchowk Campus

Internal Examiner, Nanda Bikram Adhikari, PhD

Lecturer

Department of Electronics and Computer Engineering, Pulchowk Campus

External Examiner, Mr. Bijay Kumar Roy

Deputy Director, Engineering Section

Nepal Telecommunications Authority

Project Coordinator, Surendra Shrestha, PhD

Deputy Head

Department of Electronics and Computer Engineering, Pulchowk Campus

DATE OF APPROVAL: January 31, 2011

COPYRIGHT

The author has agreed that the Library, Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the supervisors who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering in any use of the material of this project report. Copying or publication or the other use of this report for financial gain without approval of to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering and author's written permission is prohibited. Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

S.R. Joshi, PhD/ Professor
Head of Department
Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering
Lalitpur, Kathmandu
Nepal

ACKNOWLEDGEMENT

We owe our deepest gratitude to the Department of Electronics and Computer Engineering, IOE, Pulchowk Campus for providing us with an opportunity to work on a Major project as part of our syllabus. We are heartily indebted to our project supervisor Mr. Daya Sagar Baral for his constant support and guidance throughout this project. It was his valuable suggestions that helped us to cope up with the emerging obstacles during the development of this project.

We are also grateful to our Head of Department Prof. Dr. Shashidhar Ram Joshi for his interest in our project that motivated us to put more effort in this project. We are also grateful to all our teachers for their suggestions and inspirational lectures that paved the way towards the completion of this project.

We express our gratitude towards our External Examiner Mr. Bijay Kumar Roy and our Internal Examiner Nanda B. Adhikari for their valuable suggestions.

It is an honour for us to express our gratitude towards the D2 Hawkeye Services Pvt. Ltd for their faith upon us and tremendous support and motivation by our mentors Jeny Amatya and Biraj Shakya.

We also thank Er. Dhurba Adhikari for his immense support during this project. Last but not the least; we would like to thank our colleagues for their valuable comments and suggestions during this project. Any kind of suggestions or criticism will be highly appreciated and acknowledged.

Kailash Budhathoki (kailash.buki@gmail.com)

Rakesh Manandhar (razenmask@gmail.com)

Shilpa Singhal (shilpa2109@gmail.com)

January, 2011

ABSTRACT

The information present in the internet is substantially getting large these days. Additionally, the ease of access to such information has raised issues like plagiarism and its detection. Plagiarism detection itself is a vague topic since the approach of defining plagiarism varies among people and the prospective source of plagiarism is huge.

Various techniques are implemented to detect plagiarism but the quest for more accuracy always remains. In this project, we aim to attempt plagiarism detection using structure metric technique. We have used fingerprinting technique for identifying the plagiarism between documents. Fingerprints are generated by implementing Hashing Technique on the text of the document and winnowing those hash values. We have also created a web base by crawling a small portion of the web. The system encompasses a database, which is created by adding the fingerprints of the web pages in the web base along with their location. It checks the fingerprints of the provided document with that stored in the database.

We believe the research carried out during the project shall be invaluable for people with interest in plagiarism detection. It should also interest those who are keen in carrying out research in processing large scale data. The project has immense scope and it should discourage people from plagiarizing their writings.

Keywords

Database, Fingerprints, Hashing Technique, Internet, Plagiarism Detection

TABLE OF CONTENTS

PAGE OF APPROVAL.....	ii
COPYRIGHT.....	iii
ACKNOWLEDGEMENT	iv
ABSTRACT.....	v
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF ABBREVIATIONS.....	xi
1. INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Problem Description.....	2
1.3 Objectives.....	2
1.4 Scope of the project.....	2
1.5 Project Schedule.....	3
1.6 Organization of the report	3
1.7 Summary	3
2. REQUIREMENT ANALYSIS AND SPECIFICATION	4
2.1 Introduction	4
2.2 High Level Requirements	4
2.2.1 Features and modules.....	5
2.3 Functional Requirements	6
2.3.1 Specification of actors	7
2.3.2 Specification of use cases	7

2.4	Non-Functional Requirements	10
2.4.1	Usability.....	10
2.4.2	Reliability.....	10
2.4.3	Performance	11
2.4.4	Security	11
3.	LITERATURE REVIEW	12
3.1	Introduction	12
3.2	Plagiarism Detection or Prevention?.....	13
3.3	Common Techniques used by Plagiarists	13
3.4	Existing Plagiarism Detection Systems	13
3.5	GUI and Visualization.....	16
3.6	Programming Tools.....	16
3.7	Methodology	16
3.8	Summary	17
4.	DESIGN	18
4.1	Introduction	18
4.2	High Level Design	19
4.2.1	System architecture.....	19
4.2.2	Deployment diagram.....	20
4.2.3	Data flow diagram	21
4.2.4	Class diagram.....	21
4.2.5	Project tools	22
4.2.6	Design rules	23
4.3	Detail Design.....	24
4.3.1	Fingerprint generation system	24
4.3.2	DBHandler class	26
4.3.3	fingerprintDB maintainer class.....	26

4.3.4	Document class	27
4.4	Summary	27
5.	IMPLEMENTATION	29
5.1	Introduction	29
5.2	Tasks Undertaken.....	29
5.3	System Components.....	30
5.3.1	GUI	30
5.3.2	Preprocessing of the input document.....	31
5.3.3	Fingerprint generator	31
5.3.4	Web base creator.....	32
5.3.5	Fingerprint comparator	33
5.3.6	Fingerprint database maintainer.....	34
5.4	Optimization.....	34
5.5	Error Handling	35
5.6	Testing.....	36
5.7	Summary	36
6.	RESULT AND ANALYSIS.....	37
6.1	Output.....	37
6.2	Benchmarks.....	37
6.3	Comparison with Other Systems.....	42
7.	CONCLUSION AND FURTHER WORKS	43
7.1	Conclusion.....	43
7.2	Further Works	44
	REFERENCES	45
	APPENDIX A: Gantt Charts	46
	APPENDIX B: Data Flow Diagrams.....	48

APPENDIX C: Test Cases.....	51
APPENDIX D: Benchmarks.....	55
APPENDIX E: Different Implementation of Winnowing.....	57
APPENDIX F: Web Front-End	58

LIST OF TABLES

Table 6.1	Comparison of ROCOP vs. Viper.....	41
Table C.1	Test Case table for input URL to URL dispenser list	50
Table C.2	Test Case table for command to make/update DB of fingerprints.....	50
Table C.3	Test Case table for login to the system.....	51
Table C.4	Test Case table for input document.....	52
Table C.5	Test Case table for usability.....	52
Table C.6	Test Case table for performance.....	53

LIST OF FIGURES

Figure 2.1	Block diagram of the system.....	5
Figure 2.2	UCD for the system.....	6
Figure 4.1	System architecture.....	19
Figure 4.2	Deployment diagram.....	20
Figure 4.3	Context level diagram.....	21
Figure 4.4	Class diagram of the system.....	22
Figure 4.5	Fingerprinting some sample text.....	25
Figure 5.1	Back-end structure of the repository.....	32
Figure 6.1	Window size for input document vs. Percentage similarity and total fingerprint.....	37
Figure 6.2	Database rows vs. Size of web-base, database update time with indexing and without indexing.....	38
Figure A.1	Initial schedule plan.....	45
Figure A.2	Final schedule plan.....	46
Figure B.1	Level 1 DFD.....	47
Figure B.2	Level 2 DFD – checking system.....	48
Figure B.3	Level 2 DFD – web-base system.....	49
Figure D.1	Number of line position changed vs. Percentage similarity.....	54
Figure D.4	Total characters vs. File size and generation time.....	54
Figure D.5	Window size vs. Fingerprint count.....	55
Figure D.6	System load while using multi-processing.....	55
Figure E.1	Winnowing loop with higher computational complexity.....	56
Figure E.2	Winnowing loop with lower computational complexity.....	56
Figure F.1	Homepage.....	57
Figure F.2	Registration page.....	58
Figure F.3	Login page.....	59
Figure F.4	File upload page.....	60
Figure F.5	Admin-panel page.....	61
Figure F.6	Result page.....	62
Figure F.7	Plagiarised-part display page.....	63

LIST OF ABBREVIATIONS

CPU	Central Processing Unit
DB	Database
DFD	Data Flow Diagram
FP	Fingerprint
GB	Gigabyte
GUI	Graphical User Interface
HTML	Hypertext Mark-up Language
I/P	Input
KB	Kilobyte
MB	Megabyte
MS	Microsoft
O/P	Output
ROCOP	Raid on Code Pirate
SVN	Sub Version
TB	Terabyte
UCD	Use Case Diagram
URL	Universal Resource Locator

1. INTRODUCTION

1.1 Motivation

Nowadays all ideas and knowledge are based on previous knowledge and experiments, especially in academic field. Thus it is of great importance that one should clearly understand the actual meaning of plagiarism. There are a lot of misconceptions regarding what plagiarism actually means. The subject is usually confused with the terms defined as paraphrasing and common knowledge. Plagiarism is considered to be using other's ideas, thoughts, and innovations or even work without actually acknowledging the source of that information. The critical requirements like changes in a sentence with at least a single word conversion, or change in a way a sentence is presented or even making a slight change by just replacing the analogy or synonyms of some particular words fulfils the criteria for the acceptance of plagiarism.

Plagiarism has been on steep rise in recent times particularly in the field of academia. Tendency to plagiarize has been found to be increasing in students. The voluminous digital content available to students online has made plagiarism only a matter of few clicks. This seriously thwarts the goal of academic institution of producing students who can think creatively and contribute positively to society.

Such activity of copying one's thoughts and ideas without understanding the subject and without acknowledging the author should be stopped. Professors and teachers in the institutions are now finding ways to prevent such activities. Their main focus is now on encouraging students to use their own ideas as much as possible and even if they need to use any common knowledge then it should be their responsibility to use the information and form their own sentences rather than using the original words.

1.2 Problem Description

The exact definition of plagiarism seems to be unbounded, it seems vague. Plagiarism either refers, to be copied and transformed or an unacknowledged copying. It applies not only to the text or the documents but also various other aspects such as the source code of the project assignments, musical notations, literature and many more other fields. Our system ROCOP is focused to make its move against the plagiarism in text as well as source codes. The main intention is to provide a system with suitable efficiency and accuracy while detecting copying of the digital contents or plagiarized documents along with appropriate speed.

1.3 Objectives

The main objective of this project is to develop plagiarism detection software capable of detecting the extent of plagiarism in a particular document. The steps in achieving such can be summarized as follows:

- Develop a web crawler capable of crawling web pages under some particular domain.
- Create a web base of size 10MB containing pages of short-listed sites mined from the Internet.
- Create a system that checks the provided document with the pages in the web base within the time constraint imposed.
- Construct a system that provides platform independent services.

1.4 Scope of the project

As discussed above, plagiarism is visible in numerous fields, i.e. either in texts, programming assignments, musical notations or in other practical situations. Variety of approaches is applicable to handle such plagiarized contents. The approach used here is structure metric. The system will be able to detect the plagiarism mainly in the text as well

as in source codes. The project is purely based on the research along with study and analysis of various similar types of existing applications.

1.5 Project Schedule

Appendix A shows the Gantt chart for the initial project plan as well as for the schedule actually implemented.

1.6 Organization of the report

The report is divided into seven chapters. The first chapter is the introduction, and covers the objectives and scope of the project. Chapter two consists of the requirement specification of the project and includes both functional and non-functional requirements. The next chapter covers the literature review and deals with various researches done during this project. Chapter four describes the design of the project and includes the high level design and also the detail design of this project. The next chapter details out the various tasks undertaken and the system components in detail. It also covers the issues such as optimization, error handling and testing. Chapter six covers the output and various other aspects such as benchmarks and comparison of this system with an existing system. The final chapter is dedicated to the conclusion and further works of the project.

1.7 Summary

Today it has become an ease to access digital libraries with the technological advancement. Large content of information being fuelled into the world wide server and browsers makes it readily available on the Internet service. But this technology now tends to tilt towards the illegal or an unacknowledged copying which is technically termed as the plagiarism. Thus, after the simple overview of what the problem is, necessary objectives, and scope, the requirement analysis started which is stated in chapter 2.

2. Requirement Analysis and Specification

2.1 Introduction

The need of presenting System Requirement Specification is to display a detailed description of our system ROCOP. The Requirement Specification points out, what might be the reason and features that the system would offer, the interfaces of the system, what the system is supposed to do, the services and constraints on which the system will be operating. It is obvious that, being the system developed to serve the users; these specifications are must for them. Based on this specification, the user ensures whether the documentation has been done correctly and completely. Also they check in whether the system being developed meet the requirements and the features that they had proposed. Along with this, it has of great importance in around the system developers and system architects. The system developer makes this specification as a reference order to design and build the system and its capabilities.

2.2 High Level Requirements

The diagram shown below represents the way the system will be functioning. Initially - the user provides a document to the system, which is to be checked for plagiarism. The system then undergoes numerous internal processing i.e. the system compares the provided document with the large number of documents in the web base, and displays the matched documents in the hierarchy based on the percentage of plagiarism.

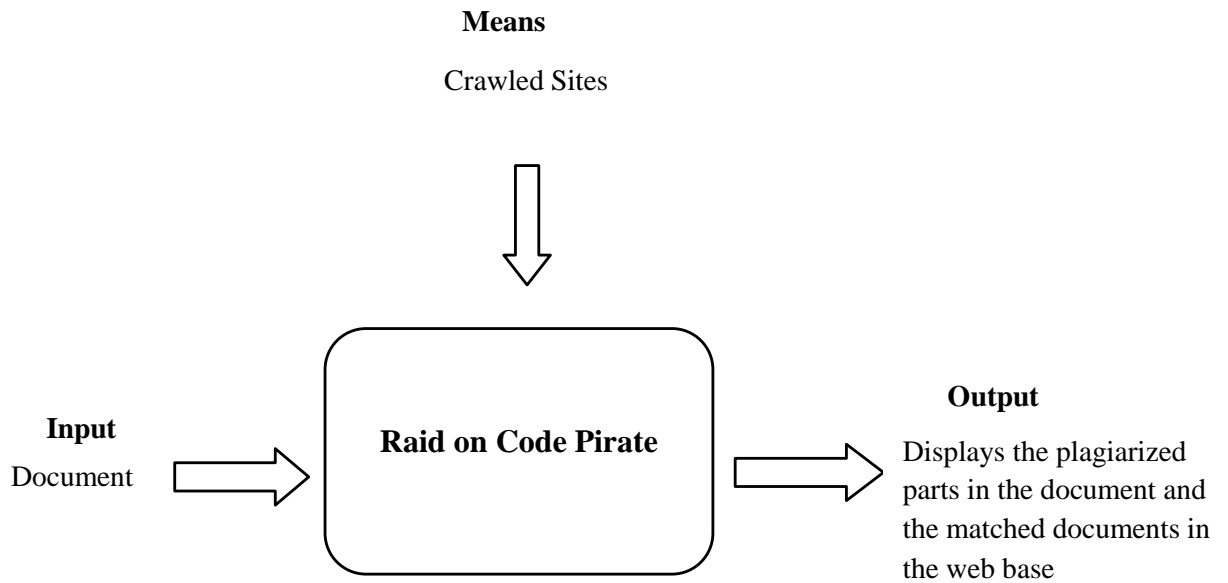


Figure 2.1

2.2.1 Features and modules

The features that the system offers can be listed as below:

- Detection

The major task for ROCOP is to detect the extent of plagiarism in a document. Thus, detection is considered as the most important feature of this system. It scans the content of the document and displays the extent of plagiarism in it and also shows the link of the pages from where the content has been plagiarized.

- Interactive User Interface

To whatever extent the system has been perfectly developed or whatever extent the system performs the functions, if the user becomes unable to operate it or faces the complications to use it, it is of no use professionally. Keeping this in mind, this system offers an interactive user interface so that they can use it with ease.

- Web-base

The system provides a web base where the large numbers of pages are crawled for particular domain and are stored. These pages are then further implemented in the detection process.

2.3 Functional Requirements

The system provides a web base where a large number of pages are crawled for a particular domain and are stored. These pages are then further implemented in the detection process.

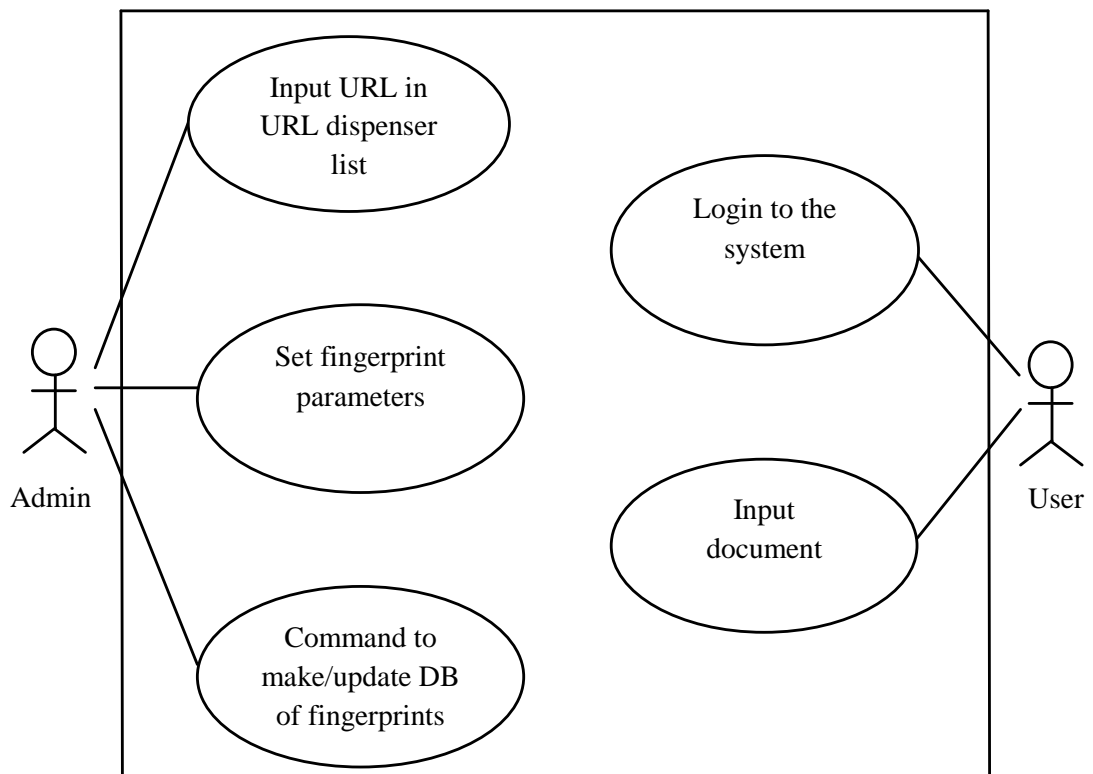


Figure 2.2

2.3.1 Specification of actors

The following actors are defined in the analysis phase of the ROCOP development process.

- Administrator

Administrator	
Element	Details
Description	An Administrator is the one who performs the back-end tasks like creating the web-base, maintaining the database of fingerprints.

- End-User

End-User	
Element	Details
Description	End-User is the person who has limited privileges and uses this system for detecting plagiarism in documents.

2.3.2 Specification of use cases

- Input URL in URL dispenser list
 - ◆ Purpose

The Administrator initially needs to specify URLs manually.

- ◆ Specifications

For each URL, pages under the same domain are downloaded.

Input URL in the URL dispenser list	
Element	Details
Actor	Administrator
Details	The administrator needs to provide the list of all possible URLs whose pages are to be crawled. At any point of time, the dispenser list can be updated.

- Command to make/update DB of fingerprints

- ◆ Purpose

A database is designed to store the large number of fingerprints for the contents present in the crawled web pages. For this, the administrator gives the specific command.

- ◆ Specifications

Once the document is provided, its fingerprints are calculated and are stored along with their corresponding document identifier, much like an inverted index.

Command to make/update DB of fingerprints	
Element	Details
Actor	Administrator
Details	The administrator must give the command to make/update the database of fingerprints of the web pages in the web base.

- Set fingerprint parameters

- ◆ Purpose

The fingerprint parameters are set in order to generate fingerprints from the document.

- ◆ Specifications

The administrator should define the fingerprint parameters like length of k-gram and the size of window by the help of which the required fingerprints can be generated.

Set fingerprint parameters	
Element	Details
Actor	Administrator

Set fingerprint parameters	
Element	Details
Details	The administrator must set the parameters required while generating fingerprints i.e. the length of the k-gram and the size of the window

- Login to the system
 - ◆ Purpose

Basically, the authentication feature is added so that the systems' security is well maintained.

- ◆ Specifications

User needs to authenticate by logging in before using the system. Existing users can login directly whereas new users need to register in order to use the system. It is must because the system has to be maintained within the circle of security. After the user logs in, he/she can use the system for the intended purpose.

Login to the system	
Element	Details
Actor	End-User
Details	The end user needs to authenticate to the system in order to maintain the security of the system.

- Input document

- ◆ Purpose

The user needs to provide the document which is to be checked for the plagiarism.

- ◆ Specifications

Once the user is directed to the file upload page, after logging in, he/she should provide a document for determining whether its contents are plagiarized or not.

Input document	
Element	Details
Actor	End-User
Details	The End-User enters the document that is to be checked for plagiarism.

2.4 Non-Functional Requirements

2.4.1 Usability

The system will provide the web interface to the end users with the optimum user-friendliness so that they can get accustomed while operating the system.

2.4.2 Reliability

The system developed will be able to meet the customer expectations as well as, it would be reliable to match up with the existing products. As per the implemented research, this system would meet the objectives as specified and would be easy to maintain and carry out for further extensions.

2.4.3 Performance

The system shall return the matched documents from its web base within considerable amount of time.

2.4.4 Security

- The system provides a web interface, so in order to make the system secure, any user needs to get authenticated. This would help the system be free from spamming and other kinds of attacks.
- To prevent the system from being heavily loaded due to large file size, the size of the input document should be limited. The limit specified here is 500KB owing to the fact that the text file extracted from a report of size approximately 1.5MB hardly exceeds 200KB.

3. LITERATURE REVIEW

3.1 Introduction

As described above, this project is all about detecting the extent of plagiarism in a particular document. One of the important steps while developing such a system is to examine all the research areas thoroughly. After describing the objectives and requirement specifications, it is now necessary to know the basic details regarding plagiarism i.e. why people plagiarize, what are its consequences. Also for designing such systems, existing plagiarism detection systems are studied. Thus based on literature, the methodology and programming tools for this system is justified. This section includes various research aspects regarding the system.

Plagiarism is derived from two Latin words: ‘plagiarius’ which means an abductor and ‘plagiare’ which means to steal. In spite of the various available definitions regarding plagiarism, it is found that most of the students are still confused as per what shall be considered as plagiarism. Though using references the copied work can be acknowledged and not considered as plagiarism but improper citation also results in plagiarism. Plagiarism can also be categorized as intentional and unintentional plagiarism. Unintentional plagiarism occurs mainly when people are ignorant of using references. It has been found that plagiarism is caused by people most likely when they have too much work to be done in short time. Such plagiarism is not considered to be done intentionally rather it is believed that lack of proper information regarding references causes it.

However the consequences of such plagiarisms depend mainly on the person who has detected it and on that basis related actions are taken.

3.2 Plagiarism Detection or Prevention?

One of the major issues regarding plagiarism is that whether it is optimum to prevent plagiarism or to cure plagiarism. As discussed above, it is found that most people plagiarize unintentionally. Thus in such cases it is optimum to avoid such plagiarism. In order to prevent, such plagiarisms it is necessary to make people aware of using references, citations and also provide them proper knowledge regarding when to use them.

Prevention requires a lot more research but detection also plays a major role as without detecting the plagiarized work, it is impossible to prevent it or even cure it. Though this project concerns mainly about the detection of plagiarism, its future enhancement can even cover the prevention techniques.

3.3 Common Techniques used by Plagiarists

It is always beneficial to know what common plagiarism techniques are practiced so that it would be easier to detect them. Some common techniques are mentioned below:

- a. Changing the word using synonyms
- b. Altering the order of their occurrence
- c. Mixing original and copied text
- d. Incorrect references
- e. Changing the variable names, function names, class names

3.4 Existing Plagiarism Detection Systems

In order to develop a plagiarism detection system, it is important to study all the existing systems so that the system to be developed covers all or at least some of the deficiencies of the existing systems. As discussed above, this plagiarism detection system compares the uploaded suspect document with a large amount of other documents stored and tries to show the matched parts of the suspect document with that stored in the web base. But in order to find that a particular document is plagiarized or not it is necessary to include the

original source from where the document has been copied into the web base. Most of the plagiarism detection system uses large, internal databases that increase on addition of each uploaded documents for analysis.

Many factors are to be considered while designing a plagiarism detection system. These factors include scope of the search, the delay time between submission of document and result obtained, the number of document that can be processed per unit time by the system, the algorithms used, the number of document that are flagged as plagiarized and the number that are not actually plagiarized but flagged as plagiarized.

Some of the existing plagiarism detection systems are mentioned below:

- Viper

It provides the free access to plagiarism detection service (on monetary basis) for text and has more than 10 billion resources in its internal database. Viper client needs to be downloaded on the desktop in order to function and the user needs to register before use. The result is shown providing the links to plagiarized works and also highlights the potential areas of plagiarism.

- MOSS

MOSS (Measure of Software Similarity) measures similarity of source codes in programming classes. MOSS is accessed as a web service and displays results using HTML pages, giving large amount of user feedback on where similarities occur [3]. Like other systems however MOSS lacks methods which can measure the similarity of a number of factors and in particular semantics [3].

- DOC Cop

DOC Cop is a plagiarism detection tool that creates reports displaying the correlation and matches between documents or a document and the Web. It is an Australian service with fast turn-around, capable of comparing multiple documents at a time.

- **CopyTracker**

CopyTracker is an online text based plagiarism detection tool. It is free, open source and easy to use, requires no installation or even login. It has no database i.e. the document uploaded by the user is deleted after processing the analysis. It supports formats such as text, HTML, MS Word 2003, MS Word 2007 and even OpenOffice Writer.
- **SeeSources**

SeeSources works by extracting all the unique keywords of the uploaded document and then uses the Internet to search for them. The result is shown as the Internet sites which matches with the document ordered according to relevancy. The results are filtered by calculating the signature similarity and matched passages are shown.
- **eTBLAST**

eTBLAST is a text similarity search engine currently offering access to the MEDLINE database, the National Institutes of Health (NIH) CRISP database, the Institute of Physics (IOP) database, and the NASA technical reports database. The eTBLAST server compares a user's natural text query to target databases using a hybrid search algorithm consisting of a low-sensitivity weighted keyword-based first pass followed by a novel sentence-alignment based second pass.
- **Chimpsy**

Chimpsy is a text plagiarism detection system that searches duplicate words within a set of documents using Google for its web base. It needs user account in order to be accessed.
- **PlagiarismCheck**

PlagiarismCheck is a text based plagiarism detection system that works by taking a document upload by user, processing it and shows the result as a .PDF report.
- **Plagiarism-Detect**

This system is used to check unlimited pages accurately without any timeout

3.5 GUI and Visualization

The main focus of this project is to detect the extent of plagiarism in a particular text uploaded by the user. Hence user plays an important role here. Another concern of the developer also lies in providing the user with an easy to access, user friendly, well designed interface. Providing such interface always helps in increasing the usability of the system. Such an interface is very necessary as that is the only way any user can interact with the system. This helps in visualizing various amounts of data in an appropriate way providing accurate and easy to understand results to the user.

Visualization plays an important role while displaying large amount of information or data because it is very essential to compress such large amount of data into small area showing comparisons.

3.6 Programming Tools

The design which is to be used for this project can be implemented using almost all programming languages. However the implementation language used here is Python programming language. Python is a platform independent language that runs on Windows, Linux/Unix, Mac OS X, and has been ported to the Java and .NET virtual machines as well. It is an OSI-approved open source language i.e. is free to use.

3.7 Methodology

Methodology is analysis of the tasks to be done in order to obtain the desired output. An appropriate methodology mainly results into a successful project and vice-versa. Here, for this system, a number of methodologies were considered and the most efficient ones are used. This doesn't mean that one particular method is used. According to the system, the most appropriate ones are used in combination.

The model used here is an iterative model i.e. in the beginning a small subset of the software requirement is developed and then using the concept of redesign and redevelopment its further versions are enhanced. This process is continued until and unless

the desired system is developed that produces results as mentioned in the system requirements.

The methodology once decided is changed during the project if there arise any circumstances where the design emerged any flaws. Thus based on the situations appropriate methodologies are implemented.

3.8 Summary

The background research focused on the clear definition of plagiarism, what causes it, who causes it? It also focused on how can one detect plagiarized text and then apply methods to prevent further plagiarism or even cure it. It also emphasized on the existing plagiarism detection systems, their plus points and minus points. It helped to decide what design, methodologies and programming tools should be used while developing this project so that it overcomes at least some of the weaknesses of the existing systems. After considering all the above factors, the following section explains the design chosen in order to solve the problems.

4. DESIGN

4.1 Introduction

This section gives a detail review on the design on which the system developed is implemented. It includes

- System architecture
- Data flow diagram
- Deployment diagram
- Detail class diagram

4.2 High Level Design

4.2.1 System Architecture

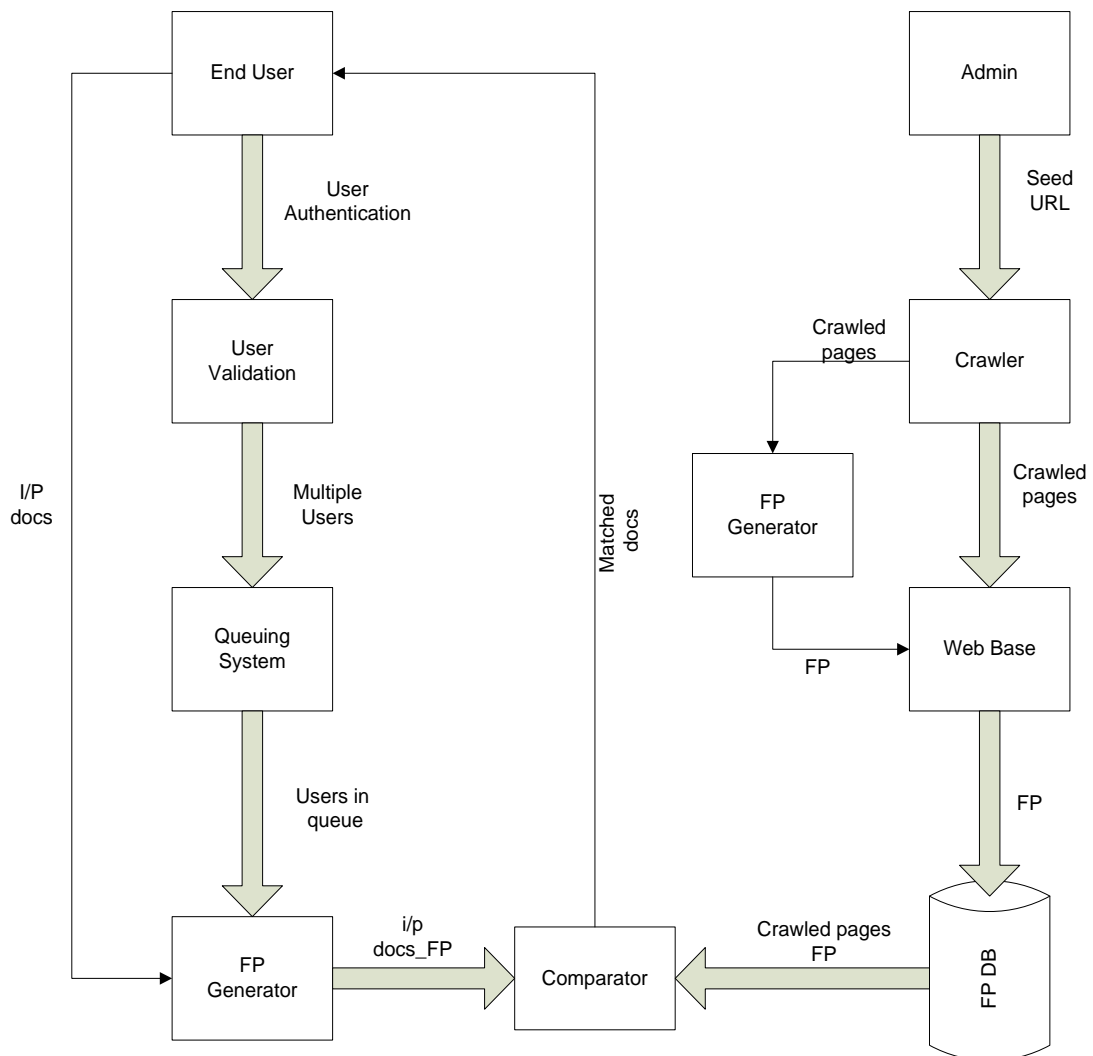


Figure 4.1

The system has client server architecture. Administrator of the system fetches the seed URLs to the URL dispenser list. The web pages from the provided URLs are then crawled and put to the Web Base. Fingerprints of the pages are then generated and kept on the fingerprint database. The end-user first needs to authenticate and then only they can upload the documents to check for plagiarism. Then the fingerprints of the input document is generated. Traffic of end-users is managed by the queuing system. Finally, the fingerprints of input document is matched with the fingerprints of web pages in the fingerprint

database. The result is shown as the percentage similarity of the input document in context to the pages stored in the web-base. It displays the plagiarized part within the document. It also shows the percentage of similarity of the document with each document in the web-base along with the links for the matched pages.

4.2.2 Deployment diagram

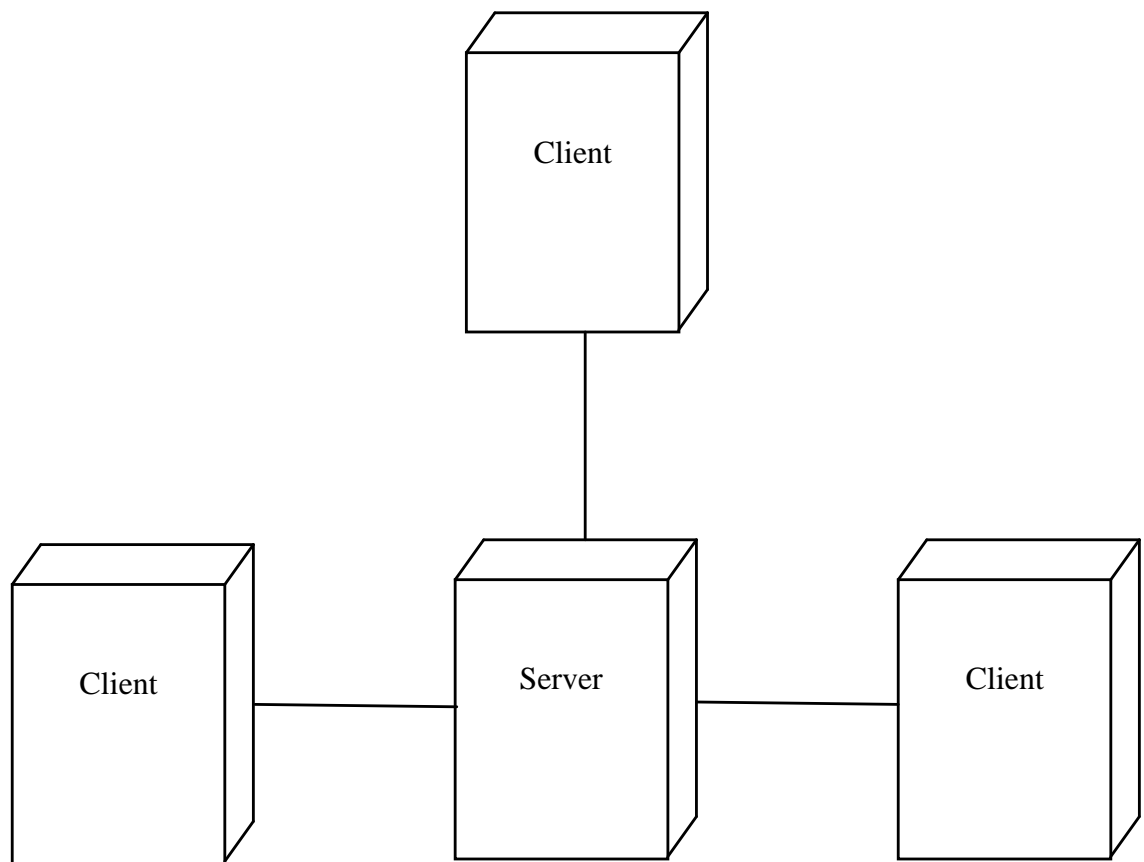


Figure 4.2

The application is built around client/server architecture. Multiple client machines can interact with the server simultaneously. Clients can interact with the system through an interactive GUI, while the server serves the client's request and does the processing in the backend.

4.2.3 Data flow diagram

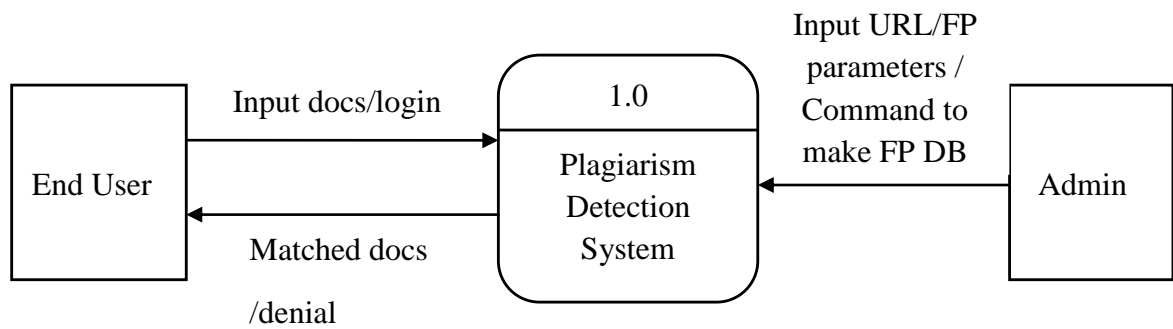


Figure 4.3

The further classification of data flow diagram is shown in Appendix B

4.2.4 Class diagram

The system is implemented by mixing both the object oriented and procedural programming methodology. Some portion of the system is built on top of framework, so using OOP in such parts has been a major difficulty. Some of the core classes of system along with their association is shown below:

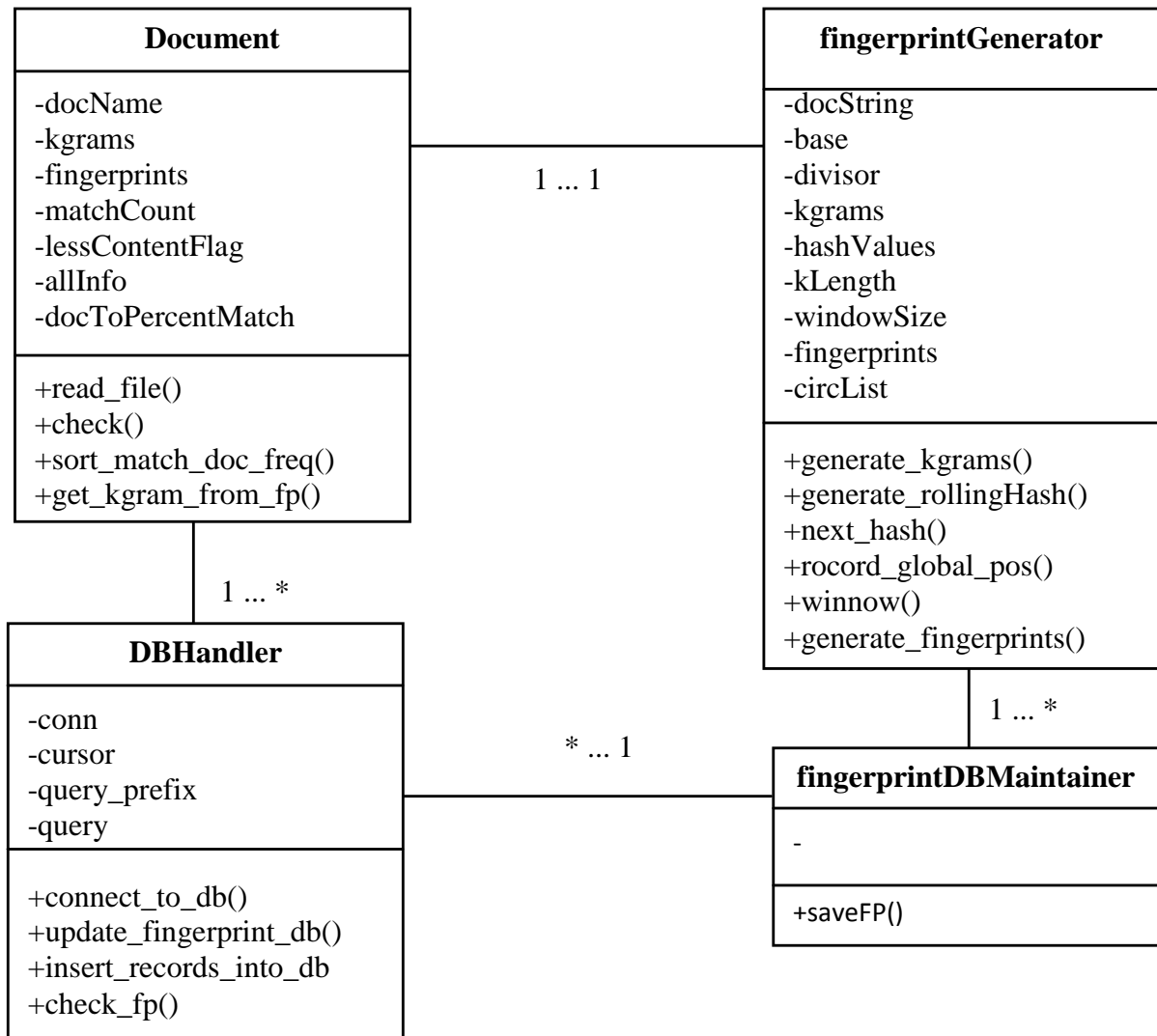


Figure 4.4

4.2.5 Project tools

- Programming Language: python
- Web Framework: Django
- External Crawler Library: Chilkat 2.0.0
- Database: MySQL Server Version 5.1.41
- Testing: PyUnit
- Versioning: SVN (<http://svn.collaborate.d2labs.org/svn/rocop>)
- Tracking: D2 Labs (<http://collaborate.d2labs.org/projects/rocop/>)

- Drawings: MS Paint, MS Visio, GIMP
- Documentation: MS Word/Excel/PowerPoint, Google Docs, OpenOffice.org Word Processor/Spreadsheet/Presentation
- Platform: Ubuntu Release 9.10
- IDE: Stani's Python Editor
- SVN Client: Rabbit VCS
- Webserver: Django development server

4.2.6 Design rules

The system is accompanied by the following set of design rules:

- There will not be any race conditions while using threads or processes for handling multiple end users.
- The uploaded (input) documents are kept on the local file system of the server and remains there until it has been checked or for a certain time.
- Garbage collection and file chunking must be considered to make the system memory efficient.

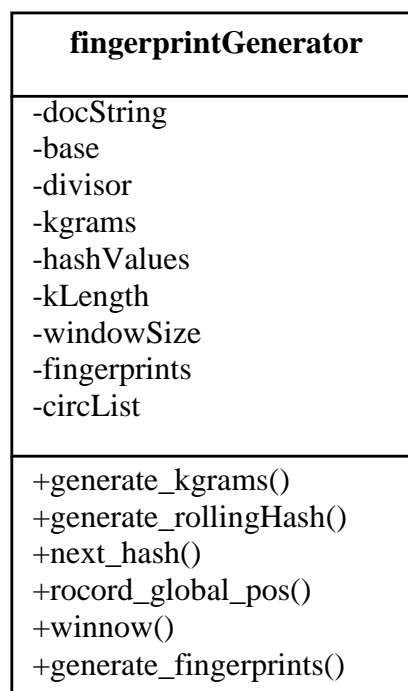
4.3 Detail Design

4.3.1 Fingerprint Generation System

- Purpose

This system is used to generate the fingerprints of both input documents and crawled pages.

- Class Diagram



- Algorithms Used

- K-grams are generated by reading the entire document as a string and then slicing that string starting from first index up to the length of k-gram and repeatedly incrementing the front index and last index simultaneously until the number of k-grams is equal to the number of characters in the document string.
- Calculate the hash from $i+1^{\text{st}}$ k-gram easily from the hash for the i^{th} k-gram using Karp-Rabin rolling hash function. Treat a k-gram $C_1 C_2 C_3 \dots C_k$ as a k-digit number in some base b , usually a large prime.

$$\text{Hash}(C_1 \dots C_k) = C_1 \times b_{k-1} + C_2 \times b_{k-2} + \dots + C_{k-1} \times b + C_k$$

$$\text{Hash}'(C_1 \dots C_k) = \text{Hash}(C_1 \dots C_k) \times b$$

$$\text{Then, Hash}'(C_2 \dots C_k) = ((\text{Hash}'(C_1 \dots C_k) - C_1 \times b_{k-1}) + C_{k+1}) \times b$$

- Modulo of the resulting hash value is calculated afterwards to confine the hash values within a certain bit limit. Karp-Rabin algorithm uses $\Theta(m)$ processing time and its worst case running time is $\Theta((n-m+1)m)$ where n is the length of the text and m is the length of the pattern that is to be discovered in the text [4].
- By taking a window size, group the hash values into window. In each window, select the minimum hash value. If there is more than one has with the minimum value, select the rightmost occurrence. This process is called winnowing and it is guaranteed to detect at least one k -gram in any shared sub-string of length $w+k-1$ where w is the window size and k is the k -length [2]. The following figure illustrates the fingerprinting process of a sample text.

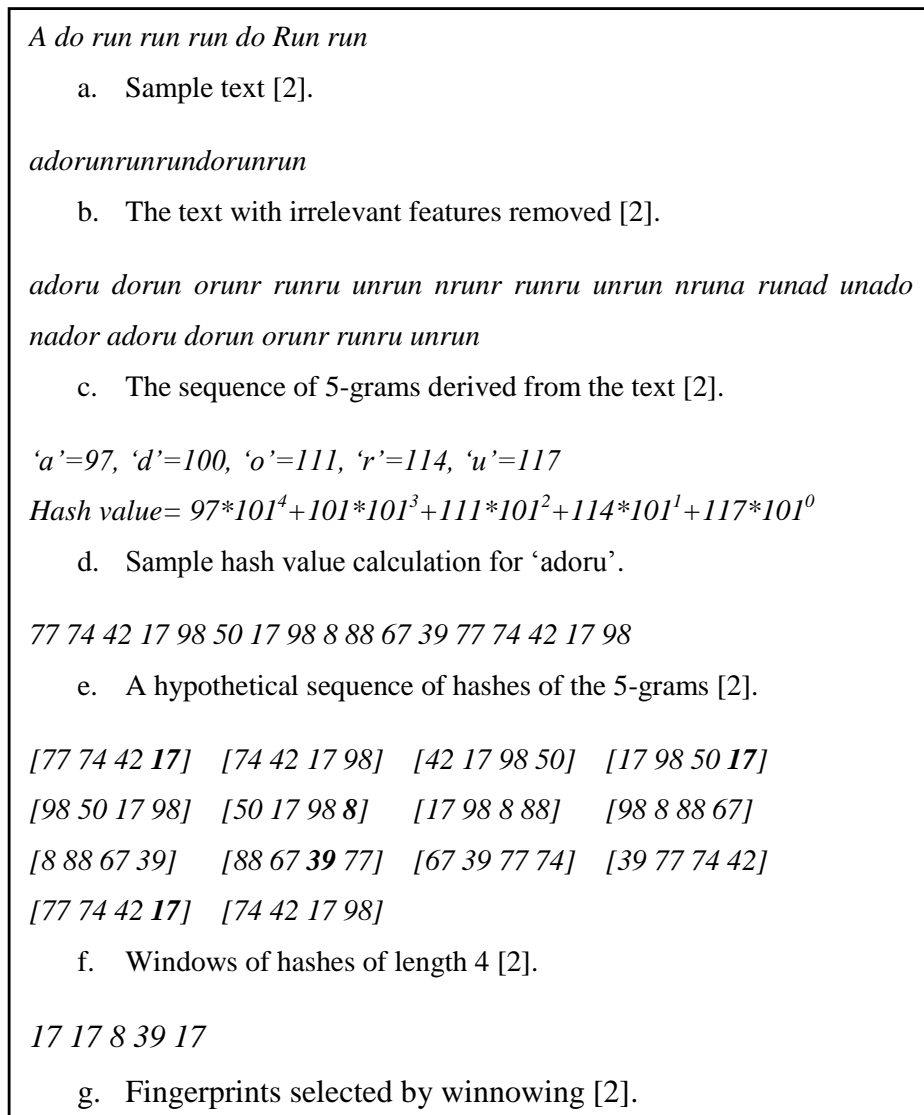


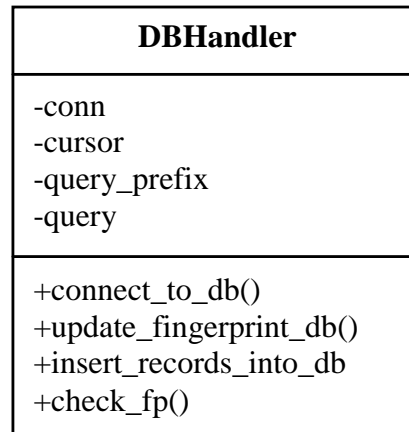
Figure 4.5

4.3.2 DBHandler Class

- Purpose

This class interfaces the database and facilitates various queries.

- Class Diagram

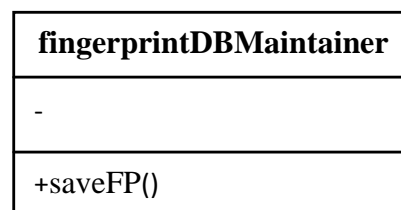


4.3.3 fingerprintDB Maintainer Class

- Purpose

The instance of this class is used for maintaining the database of fingerprints.

- Class Diagram

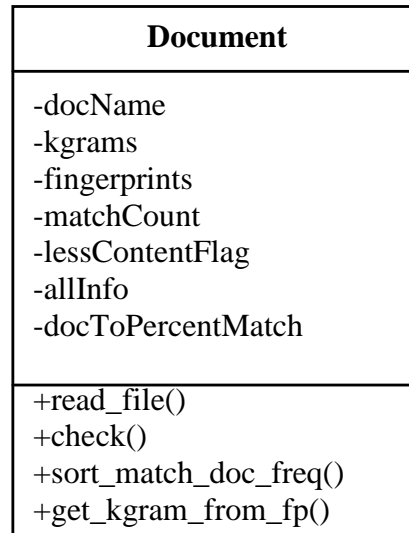


4.3.4 Document Class

- Purpose

For each uploaded document, an instance of this class is created.

- Class Diagram



4.4 Summary

After discussing the design in detail, it would be easier to implement these in order to generate a successful system. The implementation details regarding this project are explained in the following section.

5. IMPLEMENTATION

5.1 Introduction

The system components as identified in the design specification from the design phase and the software requirements specification from the analysis phase are built either from scratch or by composition in the implementation phase. This section documents the issues that arose during the implementation phase together with the adopted solutions.

5.2 Tasks Undertaken

Every tasks identified in the design specification has been carried out in this phase. Building the detection module took more time than expected because of the incorrect pseudo code presented in the research paper. Reducing the computational complexity and system load has also been the time-consuming tasks worth noting down. Dealing with multiple end users by implementing multi-threading as specified in the design specification has been modified and later multi-processing has been used by observing the systems' performance to the multiple users' simultaneous request. The process of crawling the websites, saving the WebPages to html file and then extracting the text data from the crawled html files has also been an issue because the whole documentation has to be read carefully thanks to the third party library used for the crawler. Numerous issues rose out of the blue while implementing the system as per the specifications from the earlier phase. The system which is supposed to be deployed on distributed server architecture has now been implemented in the single server architecture considering the time frame of the project's end date. Various anticipated and unanticipated problems that emerged during the implementation phase compelled us to alter the project schedule slightly as indicated in the Appendix A.

5.3 System Components

5.3.1 GUI

The application is web based so the service can be accessed through the browser. End users can interact with the system from the web page. The entire front-end for end-user is built using Django framework for python. Django provided a high level python web framework also termed Model Template View framework and facilitated faster, cleaner and pragmatic design. A separate template is used for rendering each page. Homepage of the website is the first place where the users land. There are two links for login and register. New users can register themselves by filling up the valid information and clicking on the register button. This system does not require extra entities except the username, password and email which are provided in default by the django's *forms* module so that no models have been created for the web application. Those who already have an account can directly login to the system using their username and password in the login page. Once the end users login to the system, they are directed to the file upload page. For handling the file upload box, a custom class has been created by inheriting the features of django's *forms* module and its instance is used. As soon as users select the file for detection and click the *Begin Detection* button, view file of the django app handles the client's request and spawns new processes for each user's request. The result of the detection is presented in the new page where overall document similarity, relevancy & individual similarity percentage with each source from which document was copied is presented. Moreover a part of the document which has been plagiarized is also displayed in the page to follow. The plagiarized source URL has been trimmed to 50 characters so that the entire result page wouldn't be overwhelmed with the long URLs. Django does not have any native template tags to perform this function. So a custom template tag named *truncatechars* was also created. Implementing the system as a web based service has entailed many advantages: one of them is being the platform independent access to the service.

Administrator must login in to the system before accessing the admin panel. Admin panel provides facilities like initiating the crawler, updating the fingerprint database. It also displays the name of the websites which are in the local repository. Django Celery library

is used for implementing asynchronous calls to the crawler and to the fingerprint database maintainer module.

5.3.2 Preprocessing of the input document

Documents are pre-processed before carrying out the comparison by the system. Once the document is uploaded to the remote server by the end-user, it is then read using simple file handler and then pre-processed using list comprehensions feature included in the python programming language. Irrelevant features like white spaces, upper cases, line feed characters, newline characters, etc are eliminated by this component of the system. Output of this component is the string of standard form which is then fed to the fingerprinting engine.

5.3.3 Fingerprint generator

This is one of the core components of our system. The system fully relies on the fingerprinting technique. A string of standard form which is obtained after pre-processing the input document is taken and the fingerprints are generated. The generation of fingerprints has been achieved through a series of steps in order: generating k-grams from the string, generating the hash value for each k-gram, winnowing the hash values to finally obtain the fingerprints of the document.

A simple single *for* loop with few conditional statements has been applied for generating k-grams from the standard string.

Karp-Rabin rolling hash function has been implemented in generating the hash values from each k-gram. At first simple Karp-Rabin hash function was used, the hash value generated by which was later realized to be insignificantly changing among consecutive k-grams. Once the hash values are generated, they are put on a circular list for their further use in winnowing.

Winnowing is a vital process in fingerprint generation. It has been tested with two different algorithmic implementations. Efficiency being the important consideration in fingerprinting, it has been found that implementation in figure E.2 substantially reduces the fingerprint generation time compared to the one presented in figure E.1 as shown in Appendix E. The implementation presented in figure E.2 takes benefit from the fact that in most of the time, minimum hash value from the preceding window is still in the current window.

List operations and comprehensions of python have been extensively used in this component for text processing.

5.3.4 Web base creator

Web base is a local repository of the web pages crawled from the Internet. So this component is not only concerned with the creation but also with its maintenance of web base. Chilkat, a third party python library, has been implemented in crawling the websites from the Internet. Crawler takes the list of URLs along with the number of web pages to crawl from the URLs and then crawls the web pages from each web site. The web pages from each website are stored in a directory named after the website's URL inside the repository directory. The web pages crawled from `http://www.xyz.com/` would thus be stored inside the directory named `xyz.com` which itself resides inside the repository directory. For each crawled pages, the html files are stored with the integer name in an increasing fashion. For `http://www.xyz.com/` the WebPages like `http://www.xyz.com/`, `http://www.xyz.com/abc`, `http://www.xyz.com/pqr`, `http://www.xyz.com/mno` etc will be stored inside the `xyz` directory under the name of `0.html`, `1.html`, `2.html`, `3.html`. Later only the texts are extracted from the html files using chilkat's functionality and stored again in the integer names but now with a `.txt` extension. That would mean `x.html` would thus map to `x.txt`. The URL corresponding to all text/html files are lastly written in a log file. Doing that has helped retrieving the web page URL easily by using the web page identifiers just by reading the log file. Moreover using web page identifiers reduces the size of database because of the reduced namespace. The entire back end structure of the repository is shown in figure 5.1.

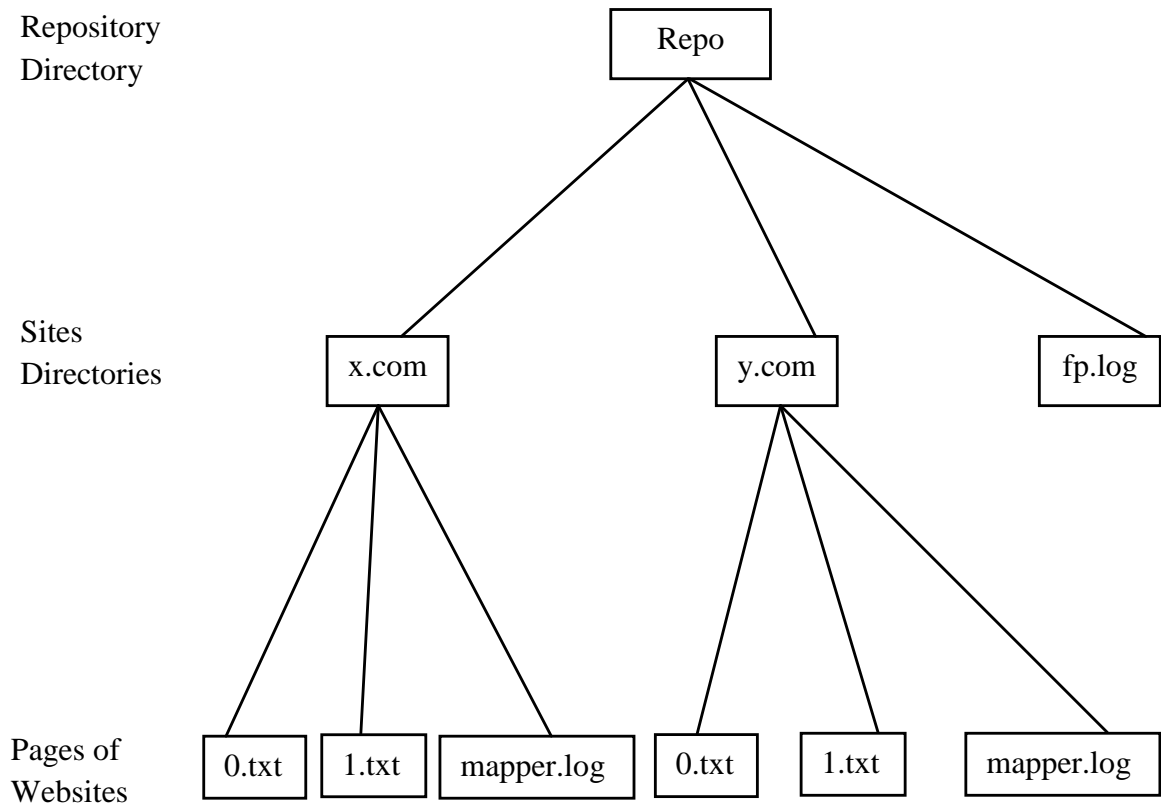


Figure 5.1

5.3.5 Fingerprint comparator

This component has been implemented using the functionality of various other modules. First, the input file is read using the file handler of python. It is then pre-processed before carrying out the comparison using the pre-processor component.

The standard string obtained from the pre-processor component is then passed to the fingerprint generator which generates the fingerprints of the string. Each fingerprint of the string is then compared against the database by calling the method, which checks the fingerprint, from the database handler module. The list of document identifiers for matching fingerprints are then stored in a dictionary containing fingerprint to list of web page identifiers as a key, value pair.

The overall similarity index is obtained by calculating the fraction of total number of matching fingerprints of a document against the database to the total number of

fingerprints of the document. The web page identifier is represented in a *web site name hyphen count* format. For the website `http://www.xyz.com/abc` stored in the repository, the web page identifier for the page `abc` would be `xyz-0` if the page `abc` is stored as `0.txt` in the repository. The web page identifiers for all matching fingerprints are collected in a single list and their corresponding web page URL along with the percentage of match between that particular web page and the input document is calculated and saved in a dictionary.

Likewise k-gram match of the input document belonging to each web page is calculated by implementing couple of loops. Also consecutive k-grams are discovered and merged if found any.

5.3.6 Fingerprint database maintainer

Building a database of fingerprints has been one of the most problematic parts in our system. This component first reads a log file before fingerprinting the web pages inside the web base. The log file contains the list of websites, the web pages of whose has been fingerprinted and inserted in the database. If any website has been recently crawled, its name won't be in the log file and by scanning this log file, this component would thus generate the fingerprints of the web pages inside those websites are generated by this component and the database is updated. Lastly the log file is also updated indicating that the web site's fingerprint has been already inserted in the database. For each web page, the standard string after pre-processing the web page file is then fetched to the fingerprint generator which generates the fingerprints. Fingerprints of a particular web page are updated in the database one by one. Database handler is called every time a fingerprint is to be updated in the database. If the database already contains the fingerprint, the web page identifier is appended on the documents column of that row much like an inverted index used by search engines else new record is inserted. The record would contain fingerprint and its web page identifier. The frequent updates within the actual web pages of a website are yet to be dealt with.

5.4 Optimization

Performance of the system is the stringent requirement of the system. The database has been first implemented without indexing the table structure. The performance of the system has been observed to be significantly slow without indexing the table. Indexing the table with fingerprint has increased the systems' performance. Owing to the optimal utilization of system hardware configuration and the independence of clients' request, multi-processing has been preferred to multi-threading. Winnowing loop has also been re-implemented considering the resource hungry nature of our previous implementation as in Appendix E.

5.5 Error Handling

Any system is exposed to both anticipated and unanticipated inputs. Thus handling of error is always crucial since at times system can malfunction because of the change in the inputs and external factors and system must handle such errors gracefully. Try, except statements provided in native python installation has been extensively used for dealing with the errors in the system.

Identification of the points and operations in the system where errors are most likely to occur plays an important role in building a robust system capable of handling errors. Opening files, registering users, login, selecting invalid file type for detection, file size exceeding the limit, providing the result when there is no case of plagiarism, crawling an invalid web site, & mostly errors while getting inputs from the users has been discovered as the error prone operations in the system. Simple conditional statements has been inserted for general problems like detecting file type, file size, etc whereas built-in try, except statements are used for solving specific issues that requires special care like `IndexError`, `ValueError`, `KeyError`, etc. Instead of forcing the system to quit, error handling has facilitated the user with an option to retry by notifying that some kind of error has occurred at a certain section or point. This has proven to be really helpful for systems' robustness.

5.6 Testing

The system has been tested since its inception for the quality assurance. The traditional approach of testing software after completion of the project has not been adopted. But rather testing has been carried out throughout the development time as indicated in the schedule in Appendix A.

Unit testing has been carried out on each module before integrating them. Black box testing and white box testing has been undertaken for ensuring the quality of the software. Black box testing has been advantageous to find out the errors in the modules just by observing the inputs and outputs of the modules. For internal details on modules, white box testing has been used. Some modules functioning properly in their standalone form might not perform well when used in conjugation with other modules. Thus the system has also undergone a thorough integration testing after integrating modules.

The manual test plan along with their expected result and observed result is presented in Appendix C. While conducting tests, some of them were unsuccessful and the errors encountered during those tests has been noted which were immediately accounted while correcting those bugs. Continuous testing has been an important factor for introducing the enhancements in the system at times. Alpha testing has also been conducted on the system. Criticisms and bugs reported during alpha testing have also been incorporated into the system. Sometimes the bug in one part of the system has an adverse effect on the other part. And fixing those bugs has taken more time than expected at many times thereby leading to the change in initial schedule plan as depicted in Gantt chart in Appendix A.

5.7 Summary

This section has documented the implementation of design solutions presented in design specification to meet the software requirements specification. The changes in initial project schedule has also been documented and justified. Moreover, testing of the system has been documented too. The following section documents the output of the system and its analysis.

6. RESULT AND ANALYSIS

6.1 Output

The system provides the result to the client in an easy to understand way. The web crawler and the fingerprint DBMaintainer has functioned successfully in the back-end. When document is uploaded to the system, it carries out the comparison and returns the result back to the end-user. The web front-end not only shows the overall document similarity but also displays the relevancy with individual documents and the parts of the document which has been plagiarized. The screenshot of the output is shown in Appendix F.

6.2 Benchmarks

The various benchmarks are shown in Appendix D. The explanation of graph is shown below:

- Number of line position changed vs. Percentage similarity:
The changes in line position inside a document results into the change in the percentage similarity of the document with the source. This change is decreasing in nature as represented by the figure D.1.
- Window size for input document Vs Percentage similarity Vs Number of fingerprints:
The database of fingerprints is generated by winnowing the web pages with window size 100 and k-grams of length 50 by observing their best pair for the optimal performance. Then the input documents are fingerprinted using various window sizes. Fingerprinting the query documents with larger window size reduces its number of fingerprints. The overall similarity percentage is highest at the window size 100 which is the exact value for both the query documents and the database generation. The most notable characteristic has been the reduction in the number of

fingerprints which implies less memory or disk accesses to lookup the fingerprints. This could be useful when it is aimed to perform faster but coarser estimate of matching in documents or reduce the work per query in a heavily loaded system. This phenomenon has been observed as depicted by the figure below:

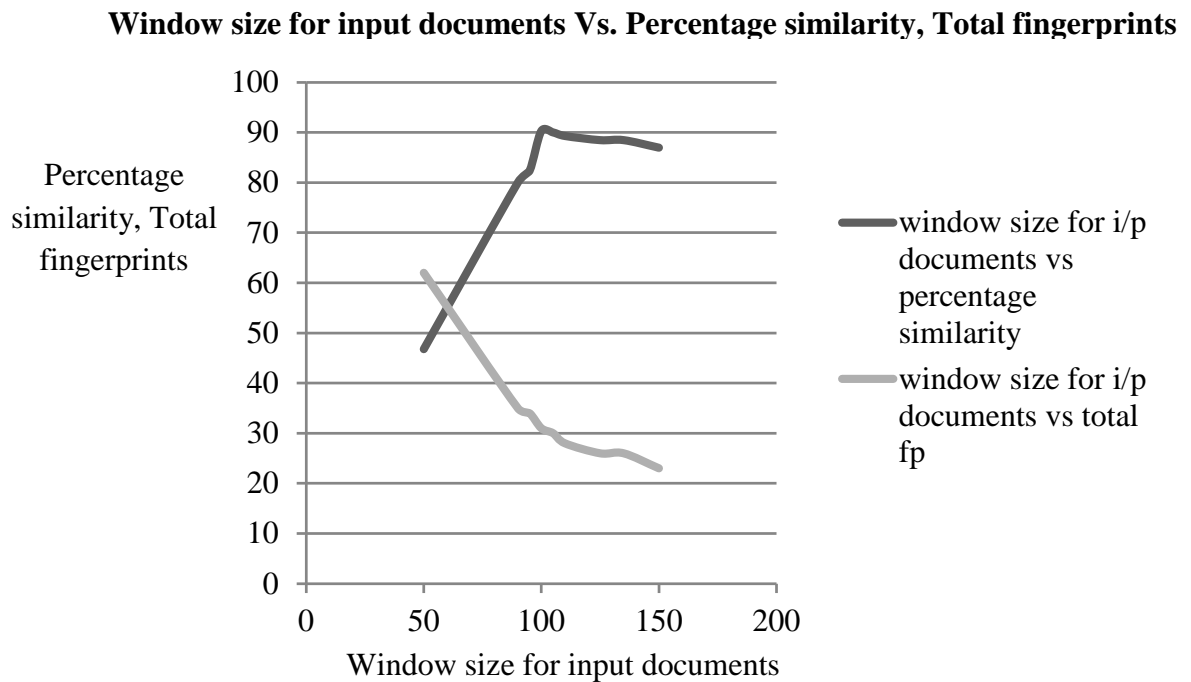


Figure 6.1

- File size (number of characters) Vs. Fingerprint generation time (CPU seconds):
As trivial as it seems, size of file is linearly proportional to the number of characters inside the file. Likewise, the fingerprint generation time which is calculated in CPU seconds is also linearly proportional to the number of characters in the document. The more the characters in the file or the more the file size, the higher the time to generate its fingerprints. This phenomenon has been described by figure D.4.
- Size of web base Vs. DB update time:
In ROCOP, fingerprints of the web pages are stored in the database in an inverted index like fashion mapping fingerprint to its document identifier. The larger the web

base size, more and more fingerprints are generated and thus the rows inside the database table scales linearly with the size of web base. The test has been carried out with two different structures of table inside the database; by indexing the table with the fingerprint and without indexing the 'fingerprint' table. The change in performance observed is highly significant as indicated by figure 6.2. The database update time corresponding to the size of web base without indexing the table has been exponential in nature. Later when the table is indexed, the database update time has been almost linearly proportional to the size of the web base. This plays an important role in scaling the system for large size of web base. However, by using a rough calculation, it would still take more than 100 days to update the database for a web base of size 1 TB. The system can gracefully scale up to web base of size 1GB even for which the database update time would be roughly around two and half hour in the current hardware configuration. NoSQL is a definitely a better alternative to opt for scaling the system to a larger web base.

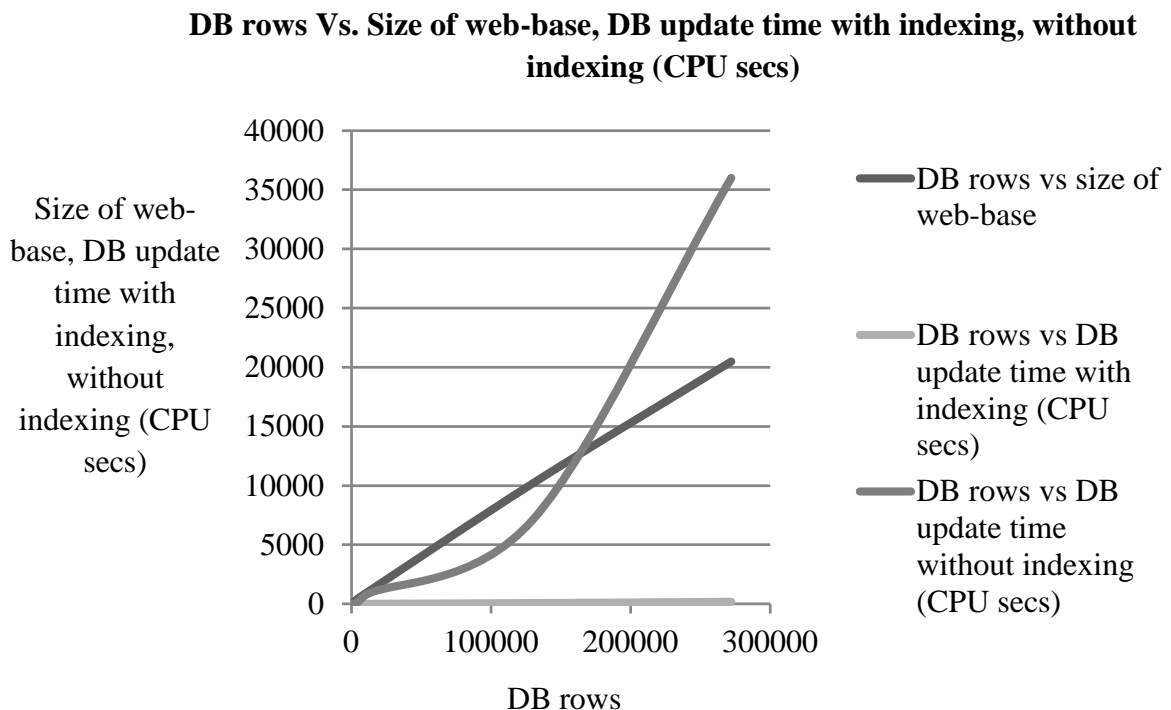


Figure 6.2

- Different pairs of k-length and window size Vs. Percentage similarity:

It is always desirable to pick the k-length and window size, which accumulates as low error as possible and picks the lesser number of fingerprints. While considering those pairs, the size of k-length shouldn't be neither too large so that the k-grams with size less than that, which would be considered a reasonable cheating, couldn't be detected nor too small to increase the time for processing. A compromise has to be made between those factors. With due observation of the system using all possible kinds of pair, k-length of 50 and window size of 100 were chosen for the future use.

- Multi-Threading Vs. Multi-Processing:

Multi-Processing was used due to the following reasons:

- Whatever be the number of threads, they run merely on one core despite using a machine with multiple cores. Processes however run on different cores utilizing the hardware configuration of the machine at its optimum.
- Spawning processes is faster than spawning threads in Linux.
- The detection requests by multiple clients in our application are independent so a parallel way of processing is required. Considering the above benefits of multi-processing over multi-threading in our typical application, we prefer multi-processing to multi-threading

The system has been tested with both multi-threaded and multi-processed architecture. Due to lack of resources it has been tested with four clients and the result showed that the system performed gracefully with multi-processing as depicted in figure D.6.

The current configuration is as shown below:

- Hardware Configuration:
 - Computer Model: MacBook2,1
 - ROM BIOS size: 2048KB
 - Processor: Intel(R) Core(TM) 2 CPU, 2 GHz
 - Cache Size: 4096 KB
 - Physical Memory: 2GB, DDR2, 667MHz
- Software Configuration:
 - OS: Ubuntu Release 9.10(Karmic)
 - Kernel: Linux 2.6.31-16-generic
 - Desktop Environment: GNOME 2.28.1
 - Softwares: Python 2.6.4rc2 (r264rc2:75497)
 - Django 1.2.3
 - MySQL Server Version 5.1.41-3

6.3 Comparison with Other Systems

The following table shows comparison of ROCOP vs. Viper:

Table 6.1

S.N.	Features	RO COP	Viper
1	Free/Open Source	Free and Open Source Software	Free (on monetary basis)
2	File Format	.txt	.doc, .pdf, .html, .rtf, .cs, .java
3	Platform Support	Platform independent	Windows only
4	Client Interface	Web Page	Viper Client (software must be downloaded for use)
5	Upload Limit	500 KB	Unlimited
6	Database Size	Small	Large (10bn resources)
7	Reliability	Higher	High
8	Comparison Algorithms	Hashing, Winnowing	Undisclosed
9	Analysis Time (for a file size of 3KB)	1.87 seconds	3 seconds
10	Citation Detection	No	Yes
11	Scope of search	Internal Database	Internal Database
12	Threshold	50 characters	No such threshold limit exists
13	Accuracy (for a particular document which is replicated from a page in the web-base)	97%	100%
14	Links to plagiarized work	Yes	yes
15	Relevancy	Yes	Yes
16	Percentage similarity index	Yes	No
17	Accepts an empty file	No	No

7. CONCLUSION AND FURTHER WORKS

7.1 Conclusion

Detection of plagiarism and hence its prevention is a very laborious work that requires deep research of the subject. This project aimed to develop a plagiarism detection system that detects the extent of plagiarism in a particular document uploaded by the client. Subsequent numbers of literatures were reviewed before starting the project. Design considerations were then carefully undertaken and implemented. The result obtained by implementing different algorithms and methods are within the desired framework. Different algorithms and methods are used and the result is shown as desired. The developed system is also compared with the existing plagiarism detection system, Viper. Though the system needs some improvements and the future enhancement is also a challenging task, the overall outcome of the project is as expected in its design consideration.

Enormous knowledge has been gained throughout the project work. The importance of the background research, requirement analysis and specifications, well designing concept, and superior methodology were learnt. Also implementation techniques, testing, error handling, optimisation issues and the predictability of problems such as when to perform a certain task, have been exercised. Thus we hope that the system developed will certainly contribute for the plagiarism detection and prevention and will be supported by many Free/Open source enthusiasts for its enhancement in the future.

7.2 Further Works

Due to time constraint, many features couldn't be incorporated in the project. The system can be upgraded in many aspects. Below is the list of tasks that can be incorporated in future.

- Using NoSQL for storing fingerprints and list of web page ids.
- Adding the feature to compare between two or more documents.
- Implementing the system in distributed server architecture.
- Using a better algorithm to find the consecutive k-grams match.
- Implementing captcha while registering users as an added security.
- Using a distributed crawler for crawling, a large portion of web.
- Compressing the crawled content.
- Incorporating the ability to update the database for re-crawling of the websites which have already been crawled and fingerprinted.
- Implementing the ability to detect citation in the document.
- Implementing the ability to insert reference in the document.
- Customizing the system to include the uploaded documents in the repository.

REFERENCES

- [1] Junghoo Cho, *et al.*, "Stanford WebBase Components and Applications", *ACM Transactions on Internet Technology (TOIT)*, May 2006.
- [2] Saul Schleimer, *et al.*, "Winnowing:Local Algorithms for Document Fingerprinting", *In the proceedings of the ACM SIGMOD international conference on management of data*, 2003.
- [3] Matthew Salisbury, "Plagiarism Detection: An Architectural and Semantic Approach", BE project, University of Leeds, 2009.
- [4] Thomas H. Cormen, *et al.*, "Introduction to Algorithms", 2nd ed., MIT press, pp 906-923, 2008.

APPENDIX A: Gantt Charts

ID	Task Name	Start	Finish	Duration	Q3 10	Q4 10		
					Sep	Oct	Nov	
1	Planning	01-09-2010	08-09-2010	1.2w				
2	Requirements	08-09-2010	15-09-2010	1.2w				
3	High Level Design	16-09-2010	22-09-2010	1w				
4	Detail Design	22-09-2010	30-09-2010	1.4w				
5	Implementation	01-10-2010	12-11-2010	6.2w				
6	Coding	01-10-2010	08-11-2010	5.4w				
7	Self Unit Testing	07-10-2010	12-11-2010	5.4w				
8	Code Integration	25-10-2010	17-11-2010	3.6w				
9	System Testing	12-11-2010	24-11-2010	1.8w				
10	Documentation	22-11-2010	30-11-2010	1.4w				

Figure A.1

ID	Task Name	Start	Finish	Duration	Q3 10	Q4 10			Q1 11
					Sep	Oct	Nov	Dec	Jan
1	Planning	01-09-2010	08-09-2010	6d	■				
2	Requirements	07-09-2010	17-09-2010	9d	■				
3	High Level Design	15-09-2010	22-09-2010	6d	■				
4	Detail Design	22-09-2010	30-09-2010	7d	■				
5	Implementation	01-10-2010	18-11-2010	35d	▼■■■■■▲				
6	Coding	01-10-2010	16-11-2010	33d	■■■■■				
7	Self Unit Testing	08-10-2010	18-11-2010	30d	■■■■■				
8	Code Integration	18-10-2010	22-11-2010	26d	■■■■■				
9	System Testing	22-11-2010	29-11-2010	6d				■	
10	Documentation	23-12-2010	06-01-2011	11d					■

Figure A.2

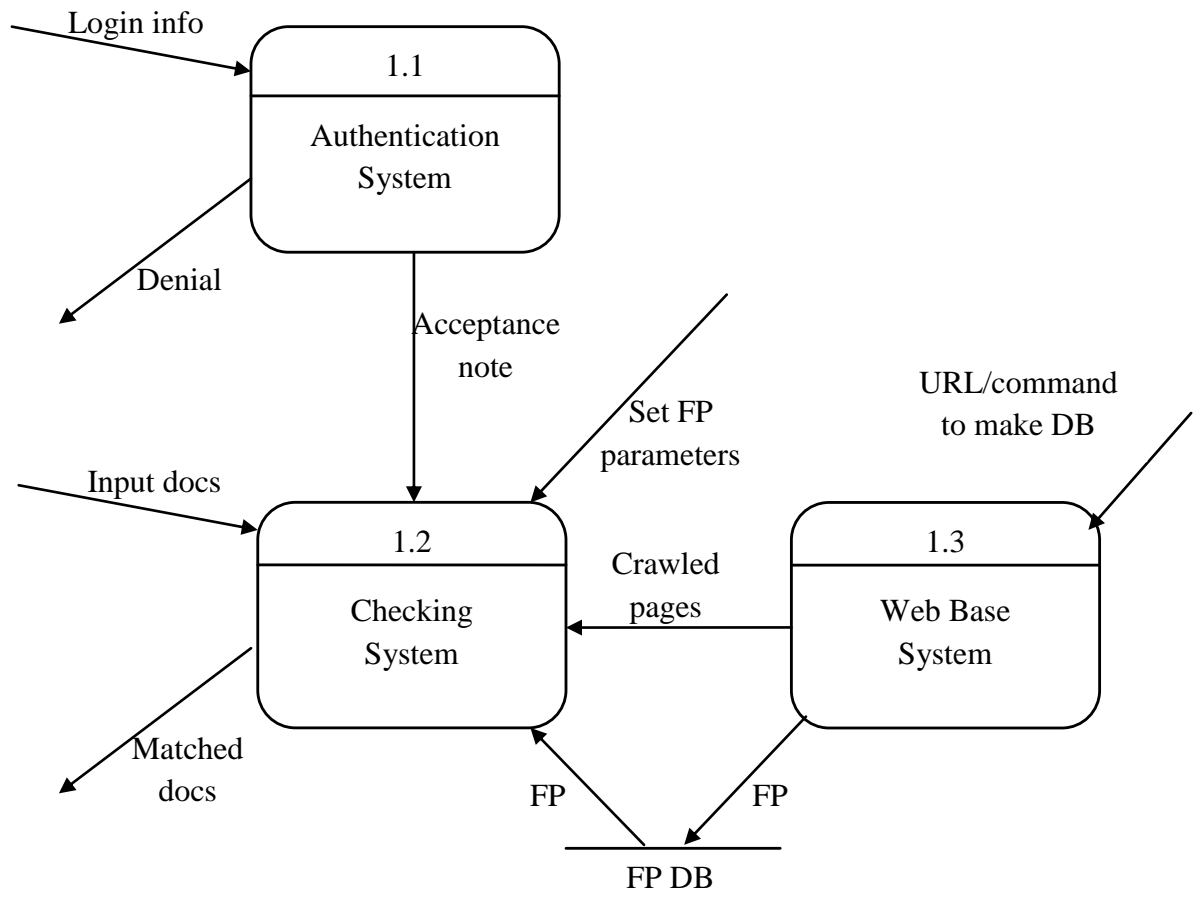
APPENDIX B: Data Flow Diagrams

Figure B.1

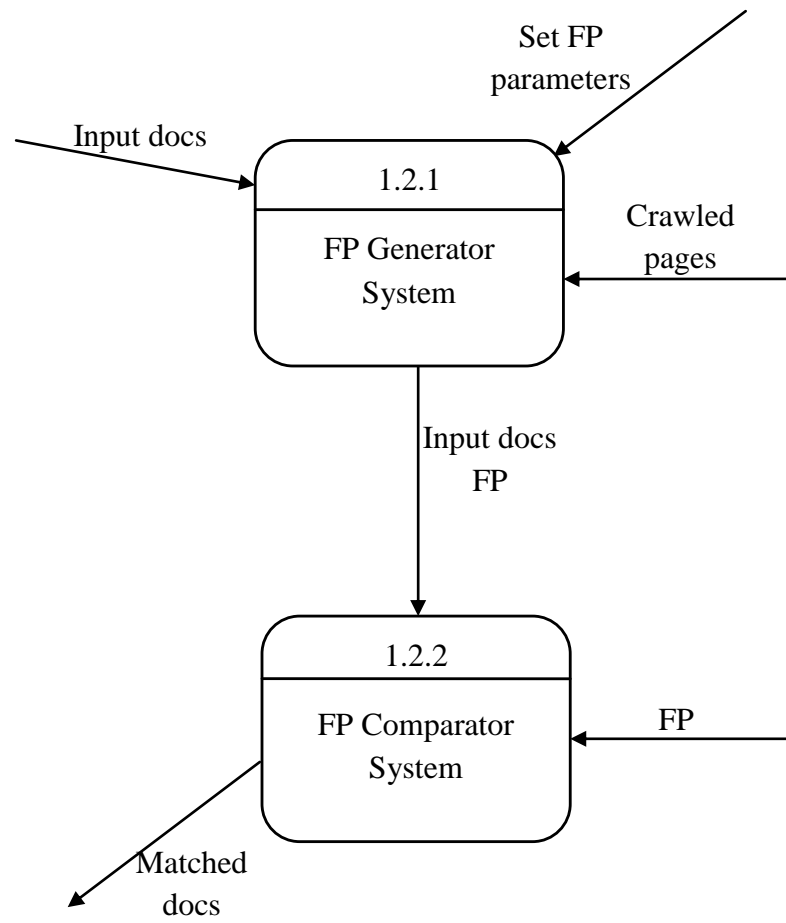


Figure B.2

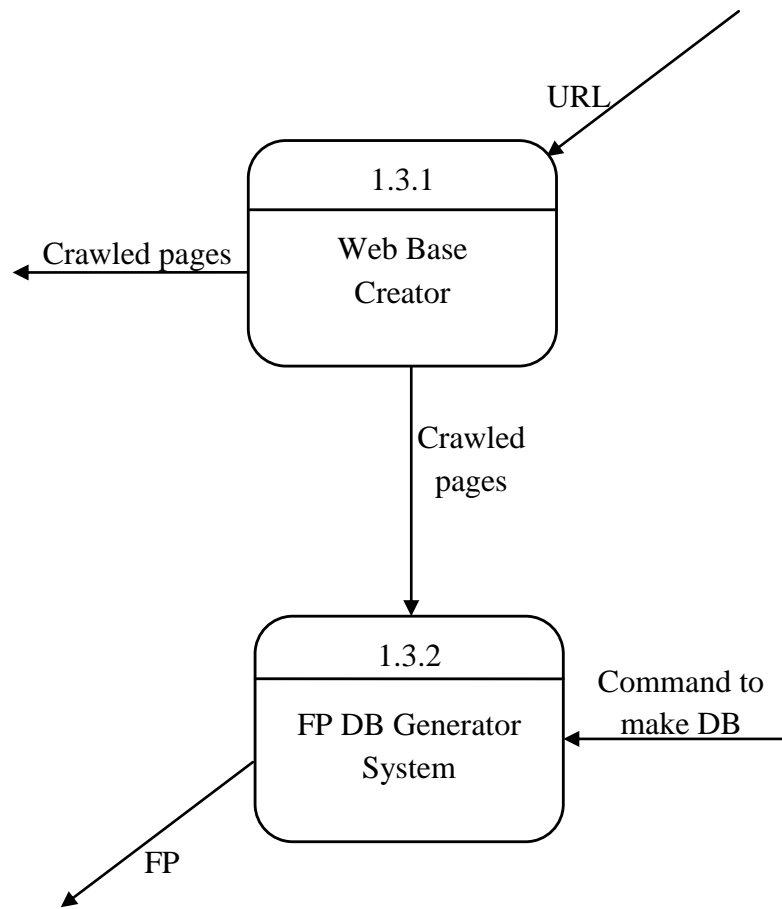


Figure B.3

APPENDIX C: Test Cases

- **On the basis of functional requirements**
 - Input URL in URL dispenser list

Table C.1

S. No.	Test Cases	Expected Results	Actual result
1	Running the crawler without providing URL	Crawler indicates that the URL is missing	As expected
2	Running the crawler without providing the number of pages to crawl from the URL	Crawler indicates that the number is missing	As expected
3	Running the crawler by providing a fake URL	Crawler signals error in the URL. It shows that URL doesn't	As expected
4	Running the crawler successfully	WebPages of the websites are downloaded on the repository along with their corresponding text files(text part extracted from html files) in addition to a mapper log file corresponding to each website containing the webpage id to URL mapping for all the WebPages inside that website directory	As expected

- Command to make/update DB of fingerprints

Table C.2

S. No.	Test Cases	Expected Results	Actual result
1	Administrator gives the command to update the database by the fingerprints	Fingerprint database maintainer scans the log file and updates the database by adding the fingerprints	As expected

	of the recently crawled websites which have not been added to the database yet	of the websites whose name is not there on the log and writes the name of the websites on the log after updates the database	
--	--	--	--

- Login to the system

Table C.3

S. No.	Test Cases	Expected Results	Actual result
1	Trying to register with an existing username	Registration error indicating that the user with that	As expected
2	Trying to register with an incorrect email format	Registration error indicating that the user with that email	As expected
3	Trying to login with an invalid username/password	Login error indicating invalid username or password	As expected
4	First logout and then try to open other pages except login & register by directly typing the URL in the browser's address bar	Redirect the end-user to the login page	As expected
5	Trying to register with different password in password and confirm password field	Registration error showing that the two password fields do not match	As expected
6	Trying to register with a new username with an existing email id	Registration error indicating that the user with that email address already exists	As expected
7	Trying to login with a valid username and password pair	User passes the authentication and a webpage with an file input field is displayed	As expected

- Input document

Table C.4

S. No.	Test Cases	Expected Results	Actual result
1	Click “Begin Detection” button without selecting a file	Stay at the same page	As expected
2	Select wrong file type when choosing a file	Stay at the same page displaying the file type error	As expected
3	Select an empty .txt file for plagiarism detection	Stay at the same page indicating that the file size is zero	As expected
4	Selecting a text file containing letters less than the k-length of the system for detection	Stay at the same page showing that the contents inside the file is too less to detect	As expected

- **On the basis of non-functional requirements**

- Usability

Table C.5

S. No.	Test Cases	Expected Results	Actual result
1	Open web page	Load home page with buttons to login and register	As expected
2	Clicking on the logout button	Log out the user from the website and redirect back to the homepage	As expected
3	Running the crawler and uploading the document for detection at the same time	Both run independently and the system handles the end users request gracefully.	As expected
4	Generating fingerprints and uploading document for detection at the same time	Both run independently and the system handles the end users request gracefully.	As expected

- Performance

Table C.6

S. No.	Test Cases	Expected Results	Actual result
1	Perform plagiarism detection on exactly replicated text doc from the webpage stored in the web base and whose fingerprints are stored in the database	System indicates 100% plagiarism on the input text doc	Observed 0-10 percentage error
2	Perform plagiarism detection on a test doc with approximate assumption that the doc is plagiarism free	System indicates no plagiarism on the input text doc	As expected
3	Simultaneous detection request to the system from multiple clients	Gracefully handle the multiple clients and return the result back to all of them	As expected
4	Does the crawler download the pages of the website in the same domain?	Downloads the pages of the website in the same domain	As expected

APPENDIX D: Benchmarks

Number of line position changed Vs. Percentage similarity

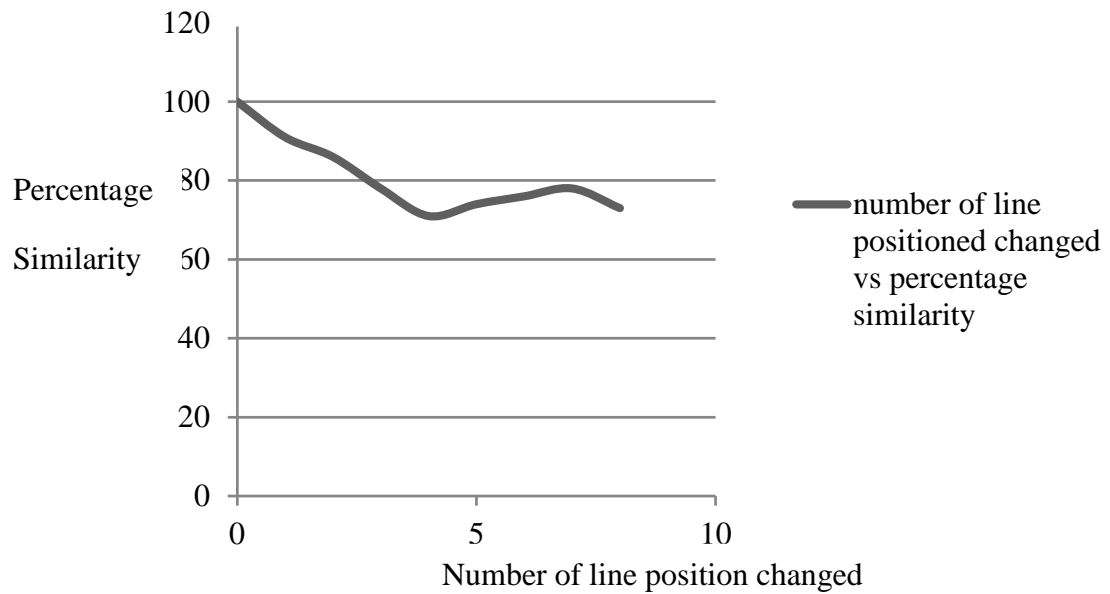


Figure D.1

Total Characters Vs. File size, Generation time (CPU)

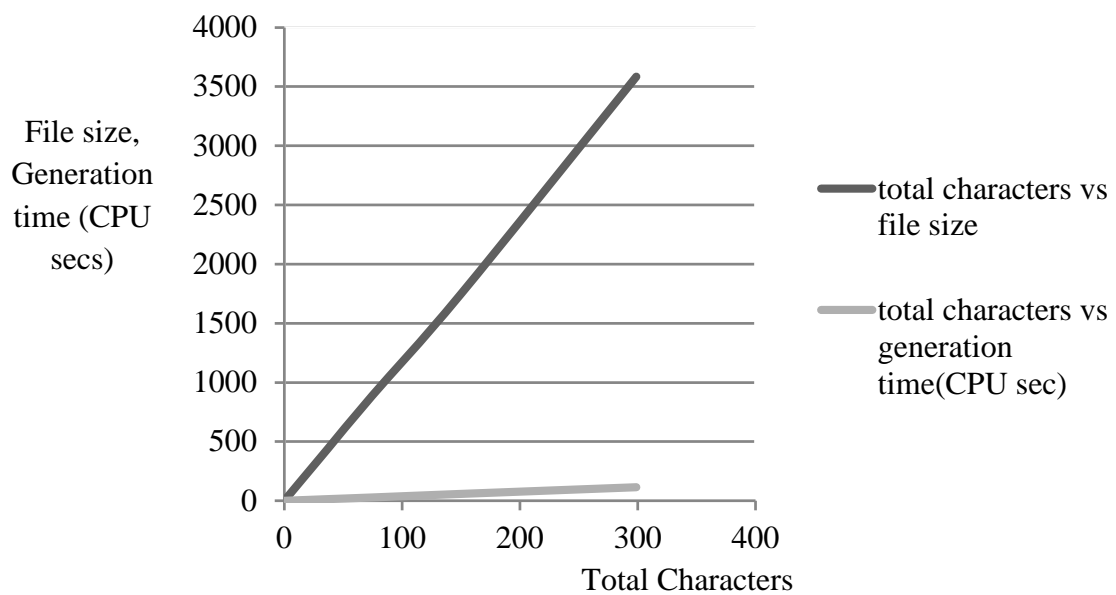


Figure D.4

Window size Vs. Fingerprint count

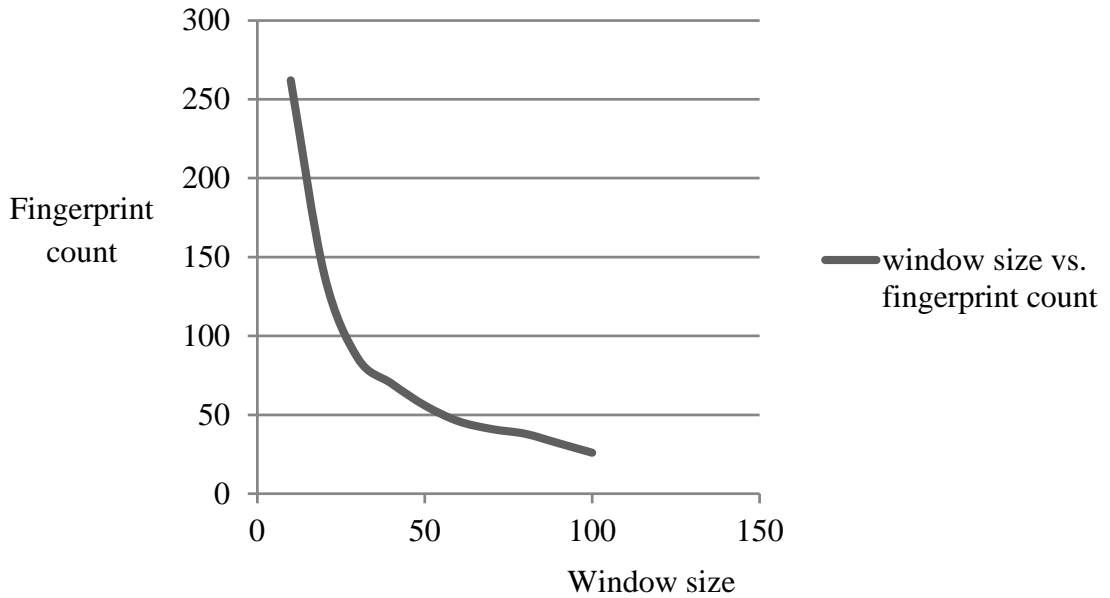


Figure D.5

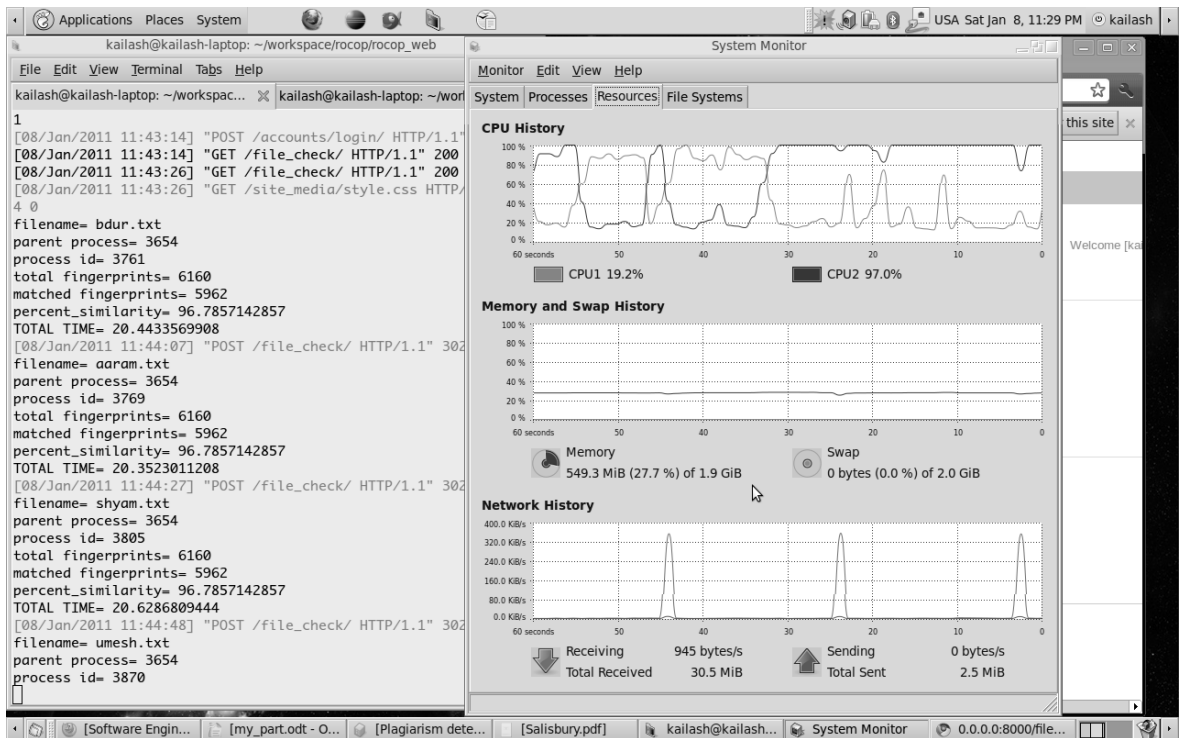


Figure D.6

APPENDIX E: Different Implementation of Wincrowing

```

def winnow(self):
    for i in xrange(0, len(self.hashValues)-self.windowSize+1):
        tempWindow = []
        for j in xrange(0, self.windowSize):
            temp = []
            temp.append(self.hashValues[i+j])
            temp.append(i+j)
            tempWindow.append(temp)
        minval = min(tempWindow)[0]
        fingerprint = max([x for x in tempWindow if x[0] == minval])
        self.fingerprints.append(fingerprint)
        self.hashWindows.append(tempWindow)
    self.fingerprints = self.unique_elements(self.fingerprints)

```

Figure E.1

```

def winnow(self):
    """
    Carries winnowing operation to produce fingerprints fromt the hash values
    """
    h = []
    for i in xrange(0, self.windowSize):
        h.append(self.next_hash())
    r = self.windowSize-1 → → → → → # window right end
    min_hash_pos = self.windowSize-1 → → → # index of minimum hash
    for i in xrange(0, len(self.hashValues)-self.windowSize+1):
        if min_hash_pos==r:
            # The previous minimum is no longer in this window
            # Scan h leftward starting from r for the rightmost minimal hash
            min_hash = min(h)
            min_hash_pos = h.index(min_hash)
            self.record_global_pos(min_hash, min_hash_pos)
        else:
            # Otherwise the previous minimum is still in this window
            # Compare against the new value and update min if necessary
            if h[r] <= min_hash:
                min_hash_pos = r
                min_hash = h[r]
            self.record_global_pos(min_hash, min_hash_pos)
        r = (r+1)%self.windowSize # shift the window by one
        h[r] = self.next_hash()

```

Figure E.2

APPENDIX F: Web Front-End

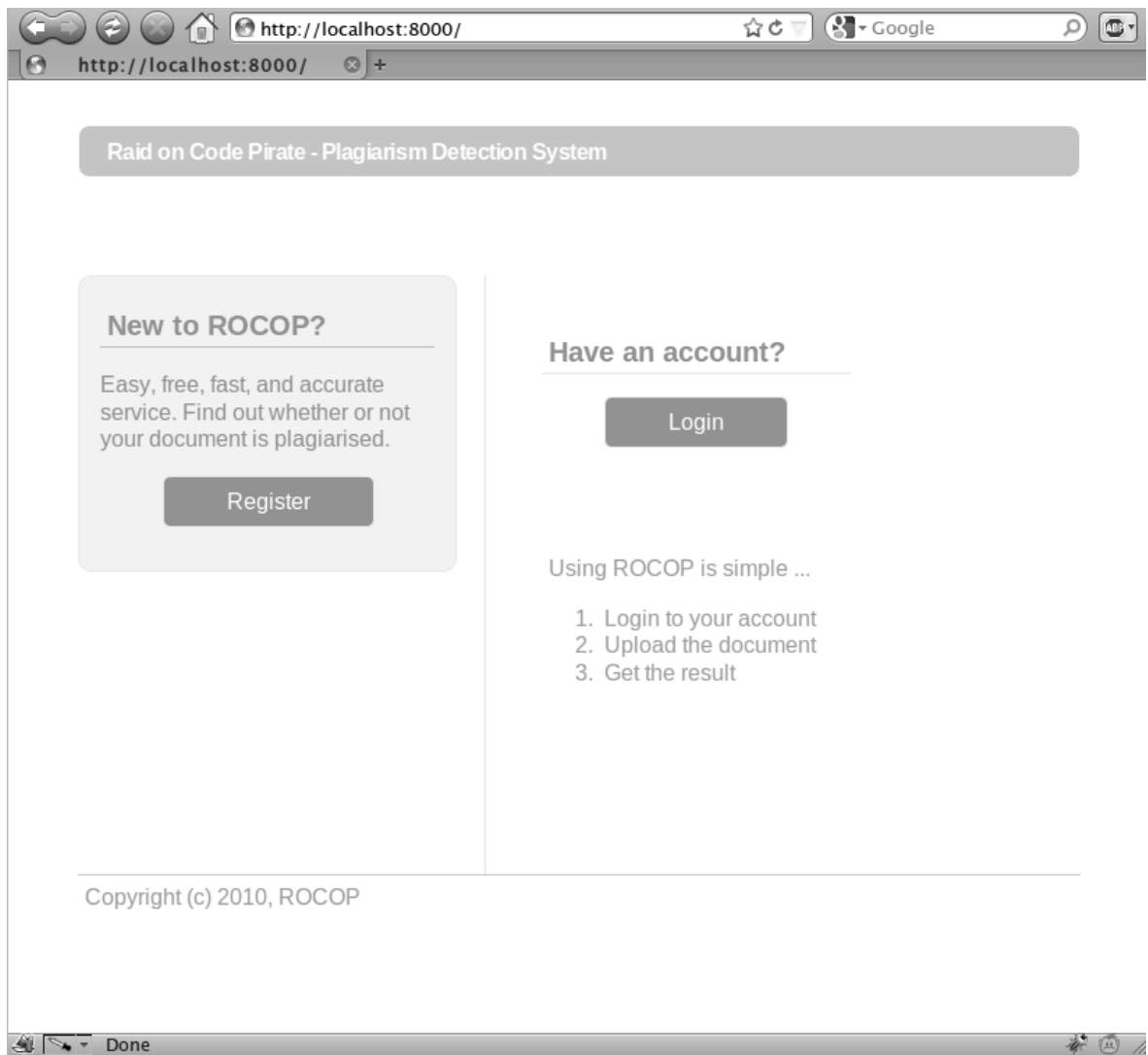


Figure F.1

The image shows a web browser window with the address bar displaying `http://localhost:8000/accounts/register/`. The page title is "Raid on Code Pirate - Plagiarism Detection System". The main content is a registration form with the following fields:

- Register** (Section Header)
- Username** (Text input field)
- Email Address** (Text input field)
- Password** (Text input field)
- Confirm password** (Text input field)

Below the form is a **Register** button. At the bottom of the form area, there are links for [Login](#) and [Home](#). At the very bottom of the page, there is a copyright notice: "Copyright (c) 2010, ROCOP".

Figure F.2

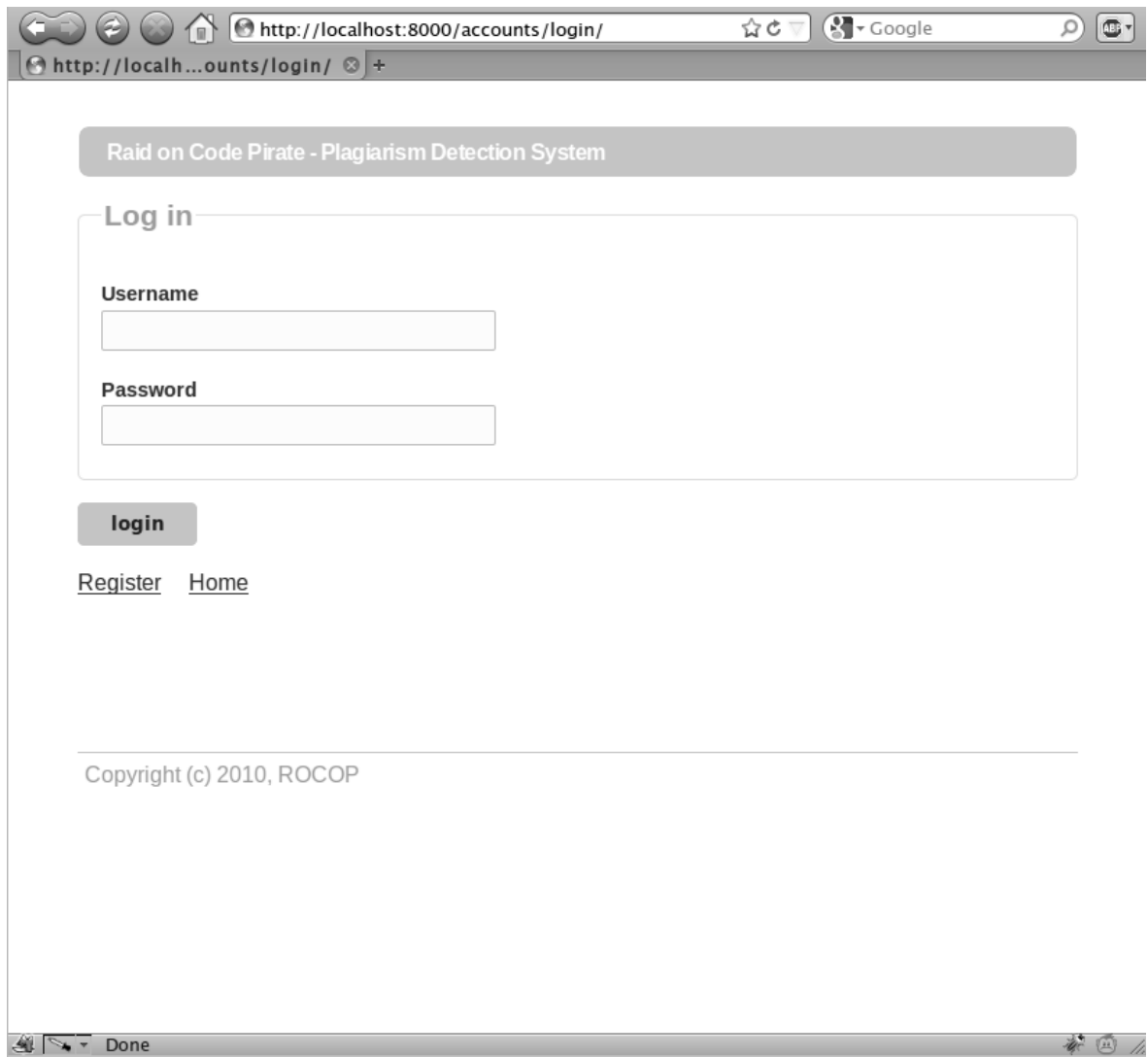


Figure F.3

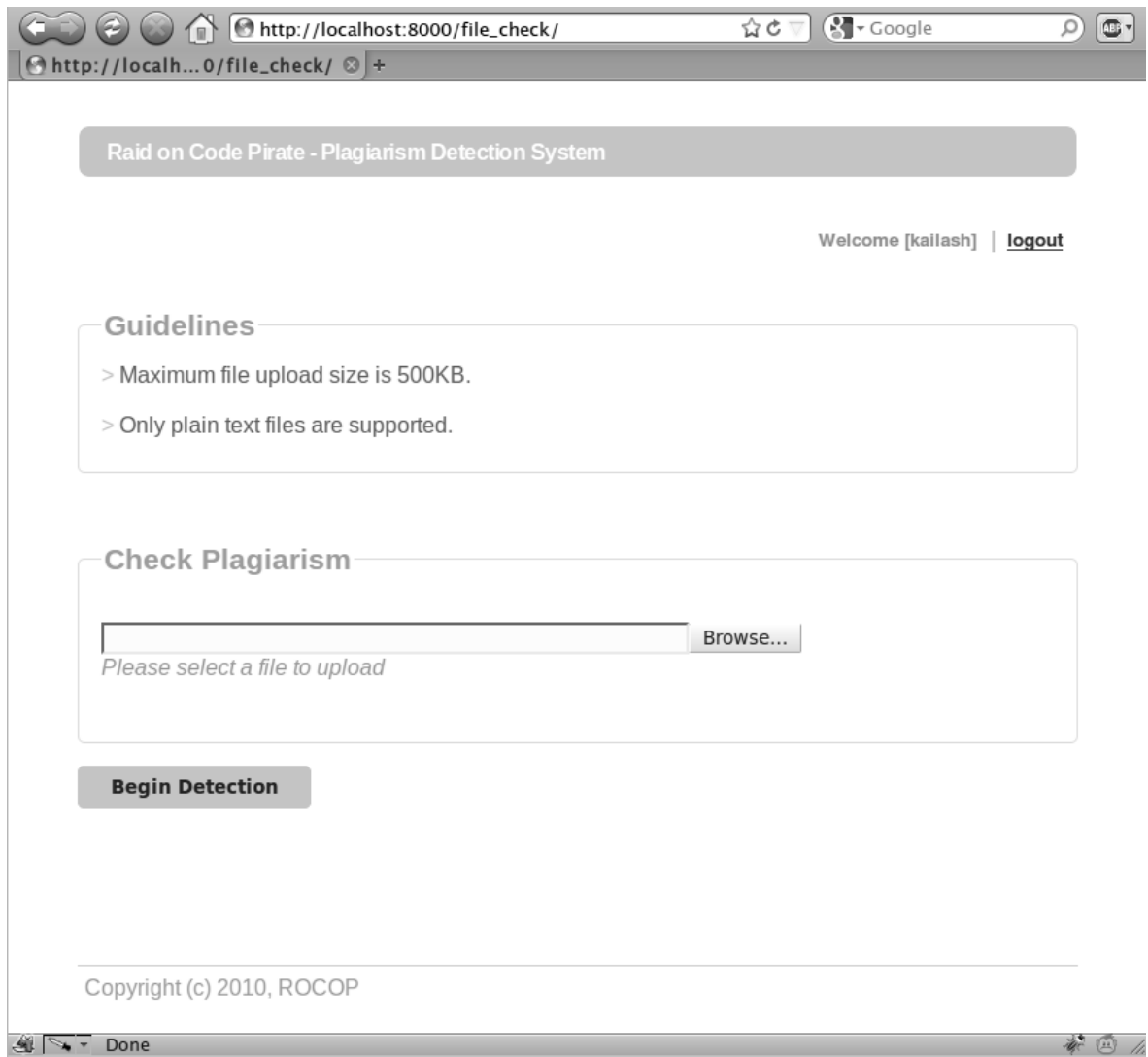


Figure F.4

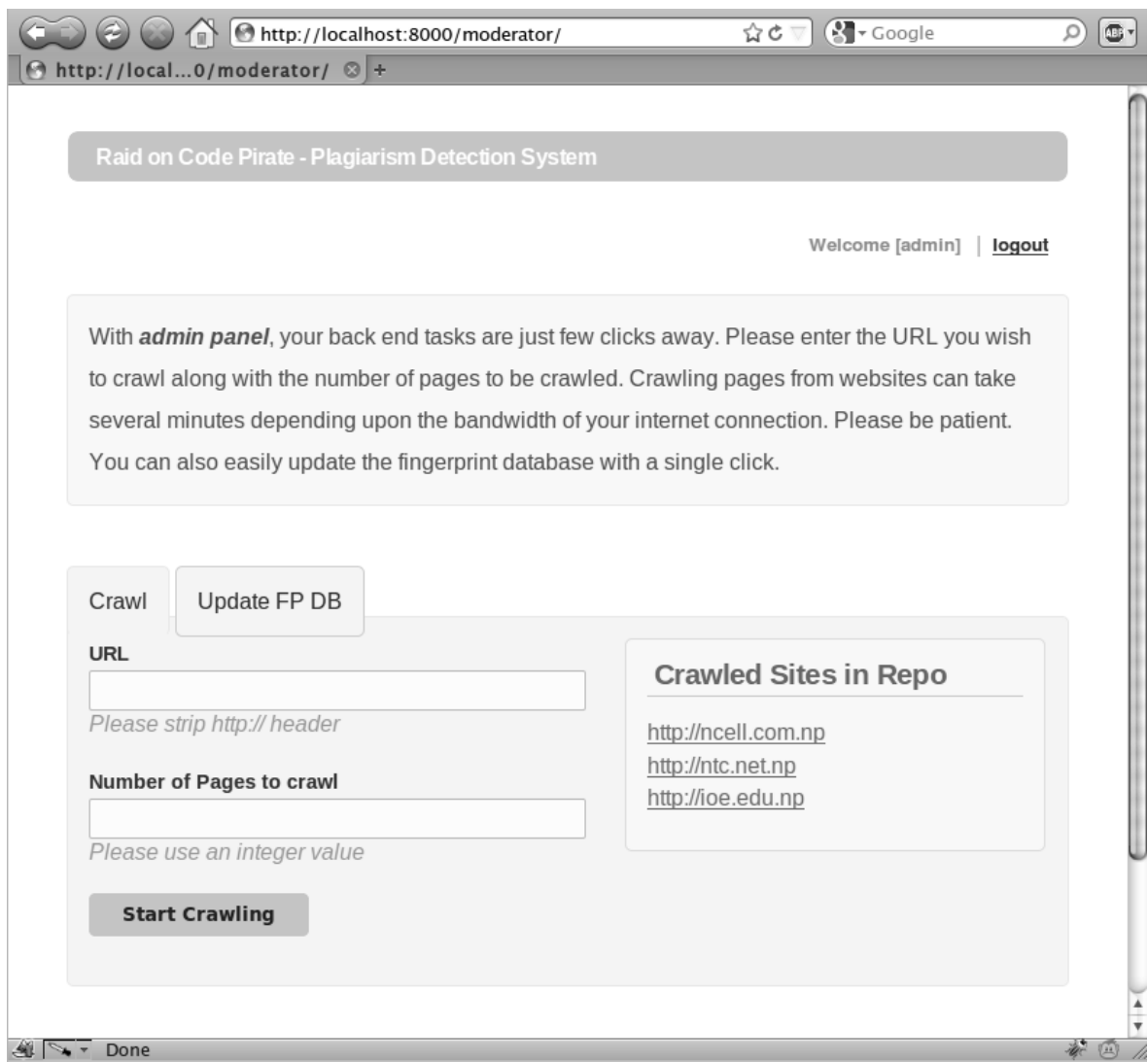
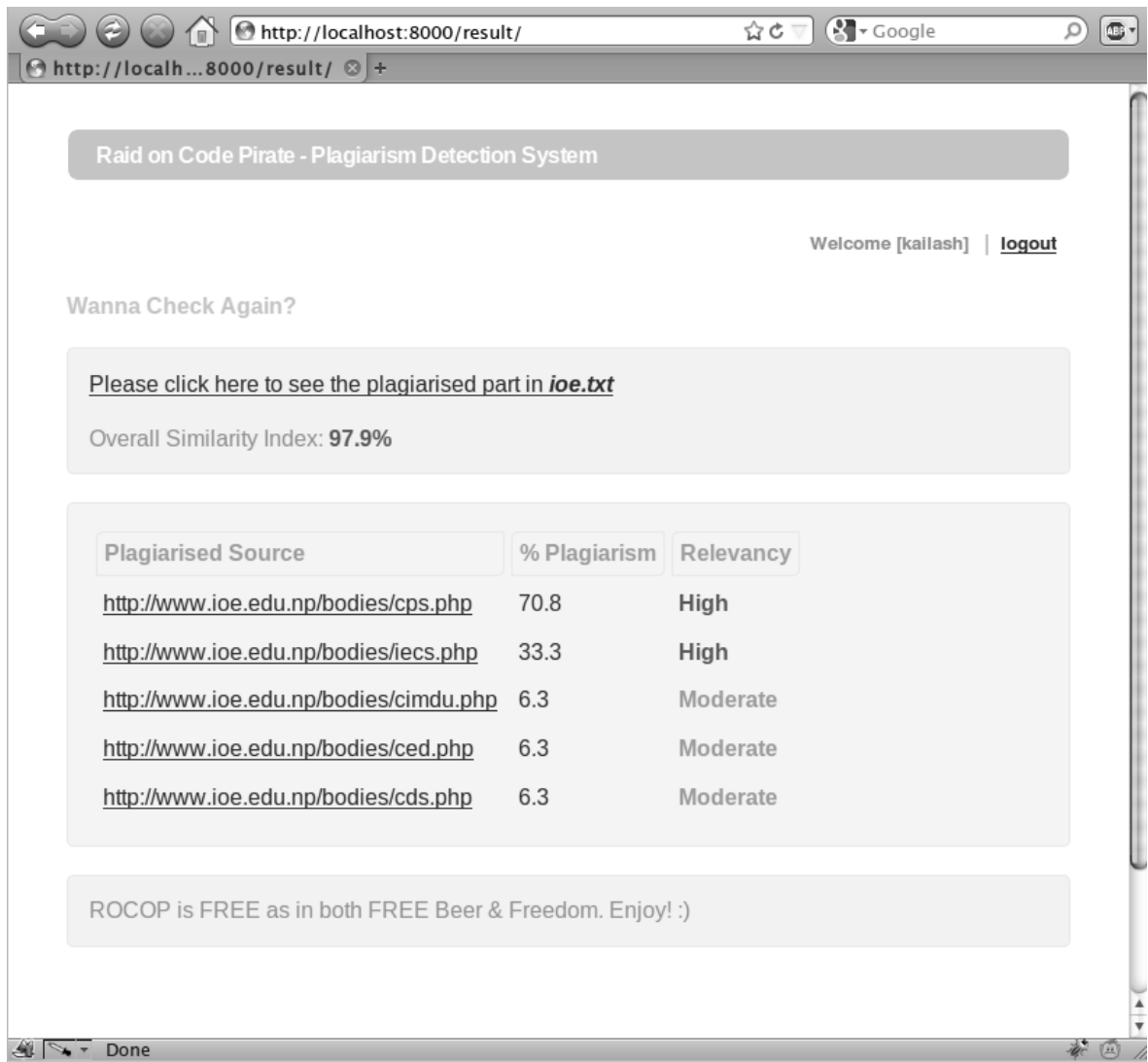


Figure F.5



Raid on Code Pirate - Plagiarism Detection System

Welcome [kailash] | [logout](#)

Wanna Check Again?

Please click here to see the plagiarised part in *ioe.txt*

Overall Similarity Index: **97.9%**

Plagiarised Source	% Plagiarism	Relevancy
http://www.ioe.edu.np/bodies/cps.php	70.8	High
http://www.ioe.edu.np/bodies/iecs.php	33.3	High
http://www.ioe.edu.np/bodies/cimdu.php	6.3	Moderate
http://www.ioe.edu.np/bodies/ced.php	6.3	Moderate
http://www.ioe.edu.np/bodies/cds.php	6.3	Moderate

ROCOP is FREE as in both FREE Beer & Freedom. Enjoy! :)

Figure F.6

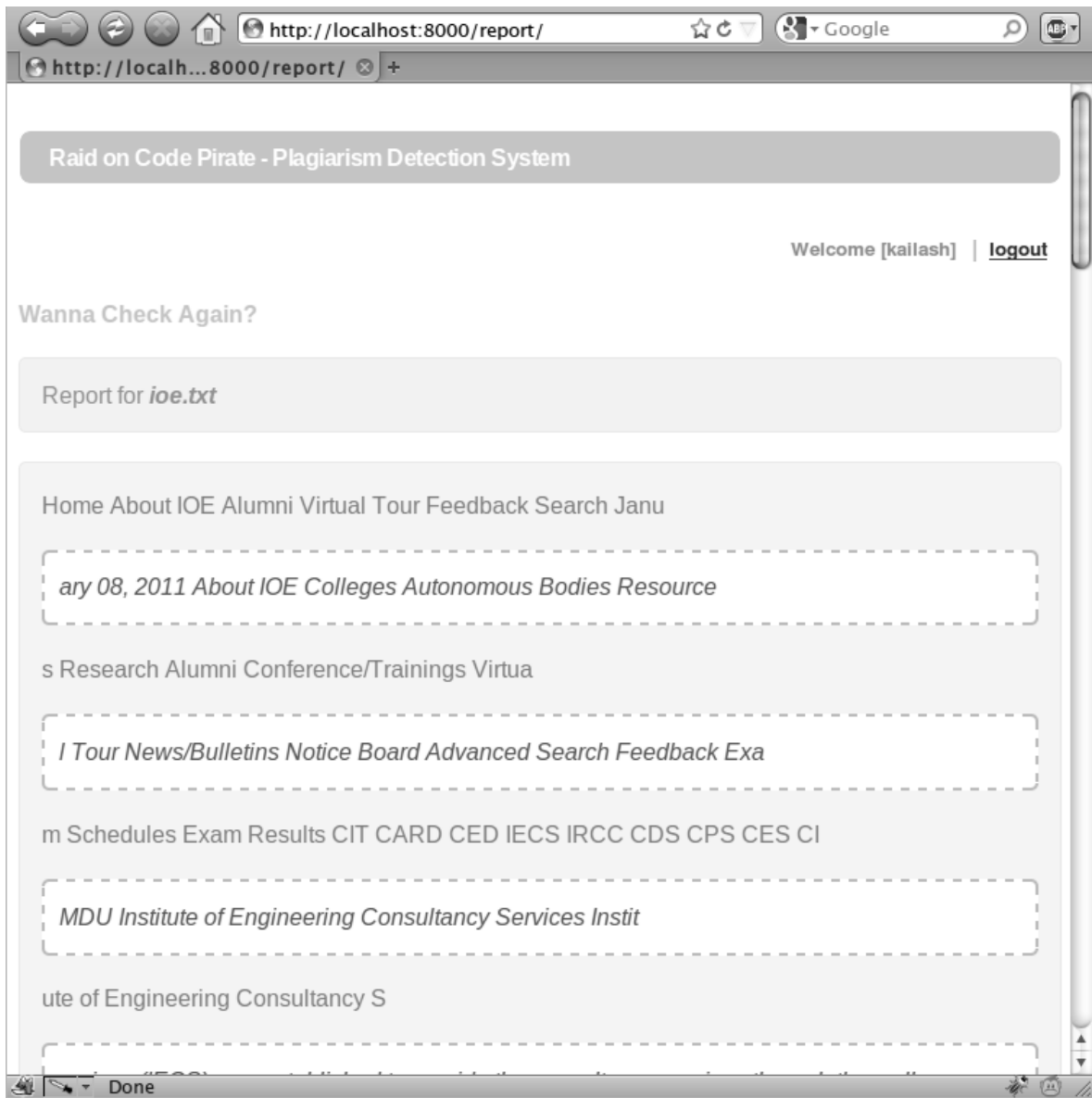


Figure F.7