

# **SoftPlag**

**Submitted in partial fulfillment of the requirements**

**of the degree of**

**Bachelor of Engineering**

**by**

**Deven Bhalerao**

**Roll No. 05**

**Supervisor:**

**Prof. Prasad Padalkar**



**Department of Information Technology**

**Don Bosco Institute of Technology**

**2016-2017**

**AFFILIATED TO**

**UNIVERSITY OF MUMBAI**

# DON BOSCO INSTITUTE OF TECHNOLOGY

Vidyavihar Station Road, Mumbai - 400070

## Department of Information Technology

### CERTIFICATE

This is to certify that the project entitled **”SoftPlag”** is a bonafide work of

**Deven Bhalerao      05**

**Manish Jain        28**

**Vishal Jha          29**

**Kunal Naik        45**

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Undergraduate in Bachelor of Information Technology**

**Date:        /        /**

**Prof. Prasad Padalkar**  
**Supervisor**

**Prof. Janhavi Baikerikar**  
**HOD, IT Department**

**Dr. Prasanna Nambiar**  
**Principal**

# **DON BOSCO INSTITUTE OF TECHNOLOGY**

**Vidyavihar Station Road, Mumbai - 400070**

**Department of Information Technology**

## **Project Report Approval for B.E.**

This project report entitled **"SoftPlag"** by **Deven Bhalerao, Manish Jain, Vishal Jha, Kunal Naik** is approved for the degree of **Bachelor of Engineering in Information Technology**

**(Examiner's Name and Signature)**

1. \_\_\_\_\_

2. \_\_\_\_\_

**(Supervisor's Name and Signature)**

1. \_\_\_\_\_

**(Chairman)**

1. \_\_\_\_\_

**Date:**

**Place:**

# **DON BOSCO INSTITUTE OF TECHNOLOGY**

**Vidyavihar Station Road, Mumbai - 400070**

## **Department of Information Technology**

### **Declaration**

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(\_\_\_\_\_ )  
(**Deven Bhalerao**)

**Date:**

# ABSTRACT

The purpose of this project is to detect plagiarism of source code in the files given by user. This plagiarism detection software will be able to detect high level plagiarism which involves minor changes in logic, changes in flow of program etc. as well as low level plagiarism which involves changing variable names, adding comments etc.

Plagiarism is widespread practice in both the software industry and the educational institutes. Companies face a constant risk of getting their code stolen by competitors and incurring a huge economic loss. Students in colleges and schools also indulge in plagiarism in their programming assignments . Plagiarism can be done easily and this makes a plagiarism detection software an extremely useful tool.

**Keywords:** Plagiarism, Java

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Problem Statement . . . . .	2
1.2	Scope of the Project . . . . .	2
1.3	Current Scenario . . . . .	2
1.4	Need for the Proposed System . . . . .	3
1.5	Summary of the Results / Task completed . . . . .	3
<b>2</b>	<b>Review of Literature</b>	<b>4</b>
2.1	Summary of the investigation in the published papers . . . . .	4
2.2	Comparison between the tools / methods / algorithms . . . . .	6
2.3	Algorithms . . . . .	7
<b>3</b>	<b>Analysis and Design</b>	<b>9</b>
3.1	Methodology / Procedure adopted . . . . .	9
3.2	Analysis . . . . .	10
3.3	Proposed System . . . . .	10
3.3.1	Hardware / Software requirements . . . . .	11
3.3.2	Design Details . . . . .	11
3.3.3	Implementation Plan . . . . .	13
<b>4</b>	<b>Results and Discussion</b>	<b>15</b>
4.1	Result of Comparison for various levels for Execution time . . .	15
<b>5</b>	<b>Conclusion</b>	<b>17</b>

# List of Figures

2.1	Spectrum of Plagiarism . . . . .	4
3.1	Timeline chart . . . . .	14

# List of Tables

1.1	Comparison of current tools . . . . .	2
2.1	Comparison of four source code detection tools with its characteristics, function and technique . . . . .	6
2.2	Comparison between different metrics : Structural Metrics and Similarity Metrics . . . . .	6



# Chapter 1

## Introduction

### 1.1 Problem Statement

To develop a tool for detecting plagiarism in software source code using the machine learning algorithms.

### 1.2 Scope of the Project

- For the first level of implementation , tool will be on a local machine for checking plagiarism.
- The source and target program should be in Java.

### 1.3 Current Scenario

	MOSS	SHERLOCK	JPLAG	CODEMATCH
Cost	Free	Open Source	Free	It is a Commercial Tool
Safety	Sign-in Required	Executes at Local machine	Sign-in Required	Executes at Local machine
Service	Internet	Standalone	Web-Service	Standalone
Algorithm	Winnowing	Token Matching	Greedy String Tiling	String Matching
Speed	Fast	More files requires more time	Fast	More files requires more time

**Table 1.1:** Comparison of current tools

## **1.4 Need for the Proposed System**

The Role of Plagiarism detection in Education, Role of Plagiarism Detection in Software, Industry especially copyright of software.

## **1.5 Summary of the Results / Task completed**

Study of different Research Papers, Thesis on Source Code Plagiarism. Survey on different source code plagiarism tools. Implementation upto Level 3 Plagiarism (Rule-based).

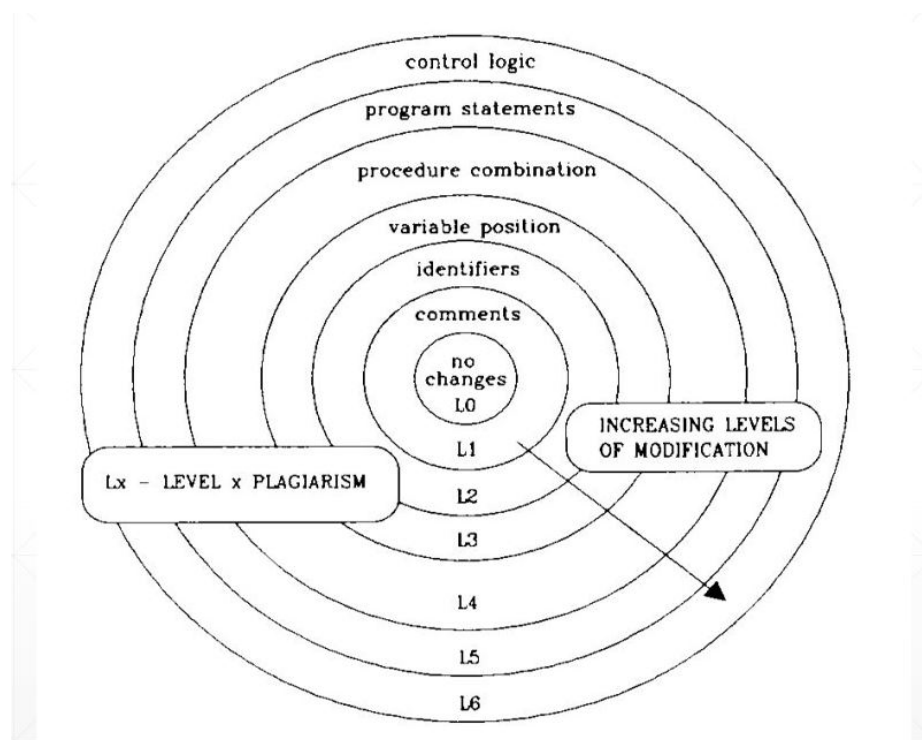
## Chapter 2

## Review of Literature

### 2.1 Summary of the investigation in the published papers

- **Computer Algorithms for Plagiarism Detection**

Described the various levels of plagiarism and the different metrics (approaches) like Halstead metric, ACCUSE metric, FORTRAN programs, etc. This paper also compared different algorithms based on the software metrics used. The different levels of plagiarism defined in this paper will form the foundation of our project and we'll be looking to detect plagiarism based on these levels.



**Figure 2.1:** Spectrum of Plagiarism

- **A Comparison of Similarity Techniques for Detecting Source Code Plagiarism**

This paper outlines different modern approaches to software similarity measurement. Algorithms like Levenshtein edit distance, Tree edit distance and graph edit distance were discussed. Knowledge of different approaches was gained and this will help us choose our approach.

- **An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis**

Latent Semantic Analysis, a statistical approach to detecting similarity is analysed in-depth in this paper. This paper also has an exhaustive description of what constitutes Plagiarism which includes surveys and research papers. Deeper understanding of Plagiarism was made possible by this research paper.

- **Plagiarism Detection in Java Code**

This thesis gives a step-by-step procedure on how to detect plagiarism in java source code using the Levenshtein Edit distance algorithm. It uses different normalization techniques and demonstrates these techniques using examples. Normalization techniques shown in this thesis will be used in our project.

- **Source Code Plagiarism Detection ‘SCPDet’: A Review**

In this paper author describes the real meaning of source code plagiarism and then described the different source code plagiarism detection tools and compared its function, characteristics and technique. In the last phase, authors discussed the different research papers and compared in tabular form with its technique, method, characteristics, functionality and its result.

## 2.2 Comparison between the tools / methods / algorithms

**Table 2.1:** Compression of four source code detection tools with its characteristics, function and technique

Tools	JPlag	SIM	MOSS	Plaggie
Open Source Tools/Paid	NO	YES	NO	YES
Local/online tool	Web	Local	Web	Local
Code Submit/File	Submit Code	Submit File	Submit Code	Submit Code
Lang. Support	6	5	23	1
Expandability	No	Yes	No	No
Founded in Year	1996	1989	1994	2002
Founded By	Guido Malpohl	Dick Grune	Aiken	Ahtiaine
Technique	Greedy String Tiling and Optimization and Tokenization	Flax lexical analyzer	Winnowing technique	Greedy String Tiling and Tokenization

**Table 2.2:** Comparison between different metrics : Structural Metrics and Similarity Metrics

Structural metrics	Similarity metrics
Structural metrics – no. of variables, no. of keywords, no. of loops, no. of comment lines	Similarity metrics – no. of characters per line, no of code lines, no. of blank lines
Structural metrics represent information about programming constructs and elements used in the code.	Similarity metrics are indicative of the style used in programming and are effective in detecting plagiarism.
Lots of rudimentary plagiarism detection algorithms like Halstead use only structural metrics and are ineffective for larger programs.	Modern approaches include algorithms that use a combination of structural and similarity metrics to detect plagiarism and are highly effective.

## 2.3 Algorithms

### • Level 0 Plagiarism Pseudo-Code

Read contents of both files

While all lines of file 1 and file 2 have not been visited

    if lines in file 1 and file 2 are not perfect match

        print “found difference”

        break out of loop

if all lines are perfect match

    print plagiarised

### • Level 1 Plagiarism Pseudo-Code

Read contents of both files

While all lines of file 1 and file 2 have not been visited

    While current lines in file 1 & file 2 are not code lines

        If line starts with “//” or is empty

            Skip line

        If line starts with /\*

            Skip all lines until \*/ is detected

        If line is code line

            set iterator condition as fulfilled and end loop

    if lines in file 1 and file 2 are not perfect match

        print “found difference” and break out of loop

if all lines are perfect match

    print plagiarised

- **Level 3 Plagiarism Pseudo-Code**

Read contents of both files

|While all lines of file 1 and file 2 have not been visited

    While current lines in file 1 & file 2 are not code lines

        If line starts with “//” or is empty

            Skip line

        If line starts with /\*

            Skip all lines until \*/ is detected

        If line is code line

count number of variables in code line and end loop

If number of variables in both files are equal

print plagiarised

# Chapter 3

## Analysis and Design

### 3.1 Methodology / Procedure adopted

- The common ways adopted by people for plagiarizing the code are:
  1. The original source code can be replicated as it is.
  2. Addition of comments in the source code.
  3. Modification in identifiers.
  4. Change in variable position.
  5. The procedure combination can be done in source code.
  6. Program statements can be changed by some modifications.
  7. Control Logic can be modified.
- Describe on the development methodology / model you would use. (E.g. Agile method or Iterative Model)
  1. In Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.
  2. An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.



- How you intend manage the weekly meetings ?

The weekly meetings need to managed properly because in order to accomplish the goals desired, you will need to have a good strategic and tactical plan. In the meeting, plans may be decided by each team member and the procedure is been planned.

- How do you intend to monitor and measure the progress of the project?

The monitoring and measure of progress of report is been done on basis of different modules. A schedule is maintained for each module to be complemented. Github is used for project monitoring ,as all the team members upload their work on completion.

## **3.2 Analysis**

Based on the requirements gathered, how was the feasibility study of the project carried out?

The project is about the detection of software source code plagiarism, so the study carried out on the comparison of the different plagiarism software's which are based on rule based algorithms.

On comparison of different features of some software's.

The papers were referred for plagiarism detection, which gives the idea of different levels of plagiarism and defining the metrics for the programs.

If any requirements, were modified why they were modified?

## **3.3 Proposed System**

Committing plagiarism is extremely easy as we can simply copy paste code and claim it as our own. There are lots of techniques like changing variable names, adding comments, changing the loop types which are used to hide the plagiarism. For this reason, sophisticated plagiarism detection software is needed to prevent plagiarism in both educational institutions and software industry.

Therefore, we propose this system that can detect different kinds of plagiarism.

### 3.3.1 Hardware / Software requirements

- Development Hardware / Software requirements

Software Requirements :

Java JDK 1.4 or more

Operating System : Window/Linux

Front-End : Java Fx

Back-End : MySQL

Hardware Requirements :

Minimum Memory : 512 Mb RAM

Hard disk space : 1 GB

- Deployment Hardware / Software requirements

Software Requirements :

Java Runtime Environment

Operating System : Window/Linux.

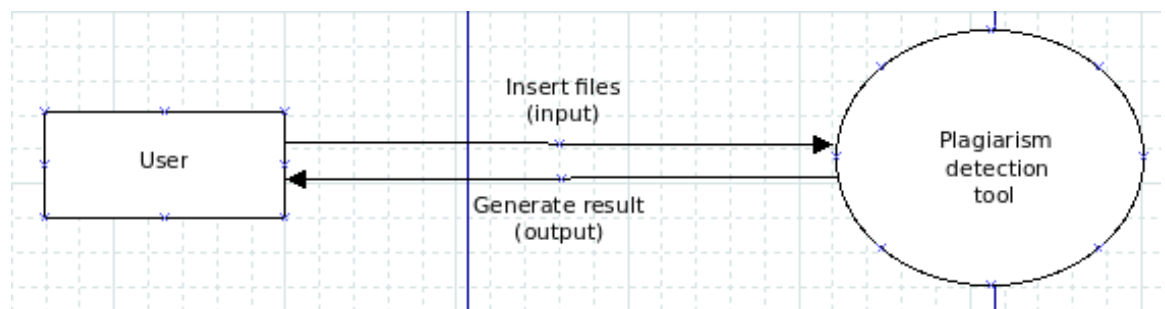
Hardware Requirements :

Minimum Memory : 512 Mb RAM

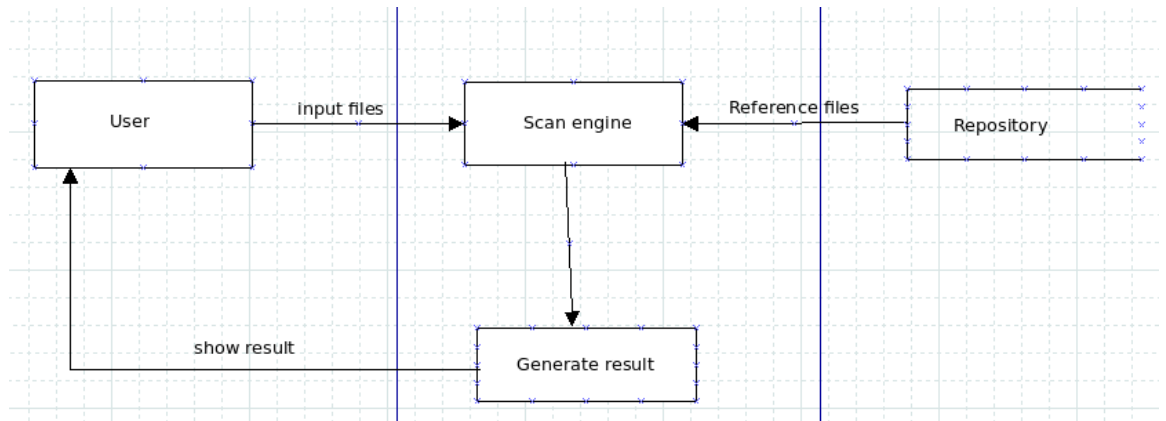
Hard disk space : 1 GB

### 3.3.2 Design Details

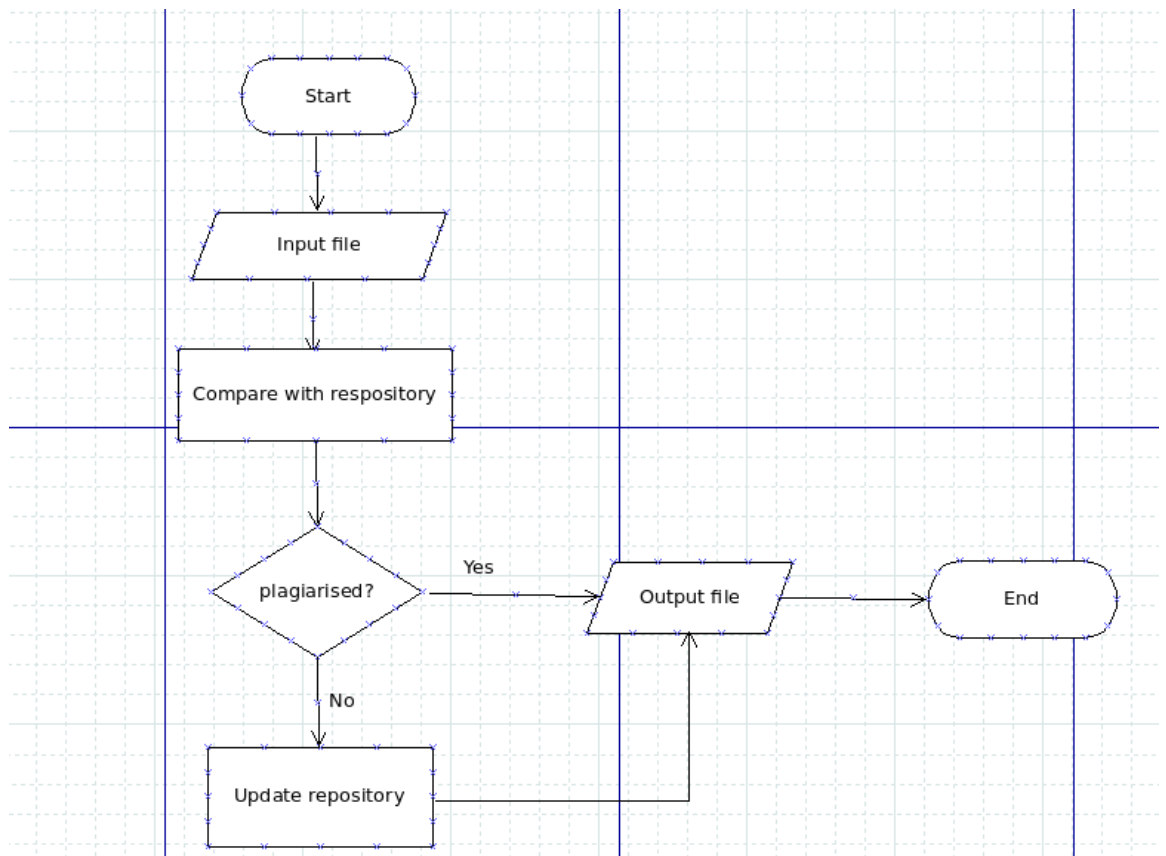
- Data Flow Diagram Level 0



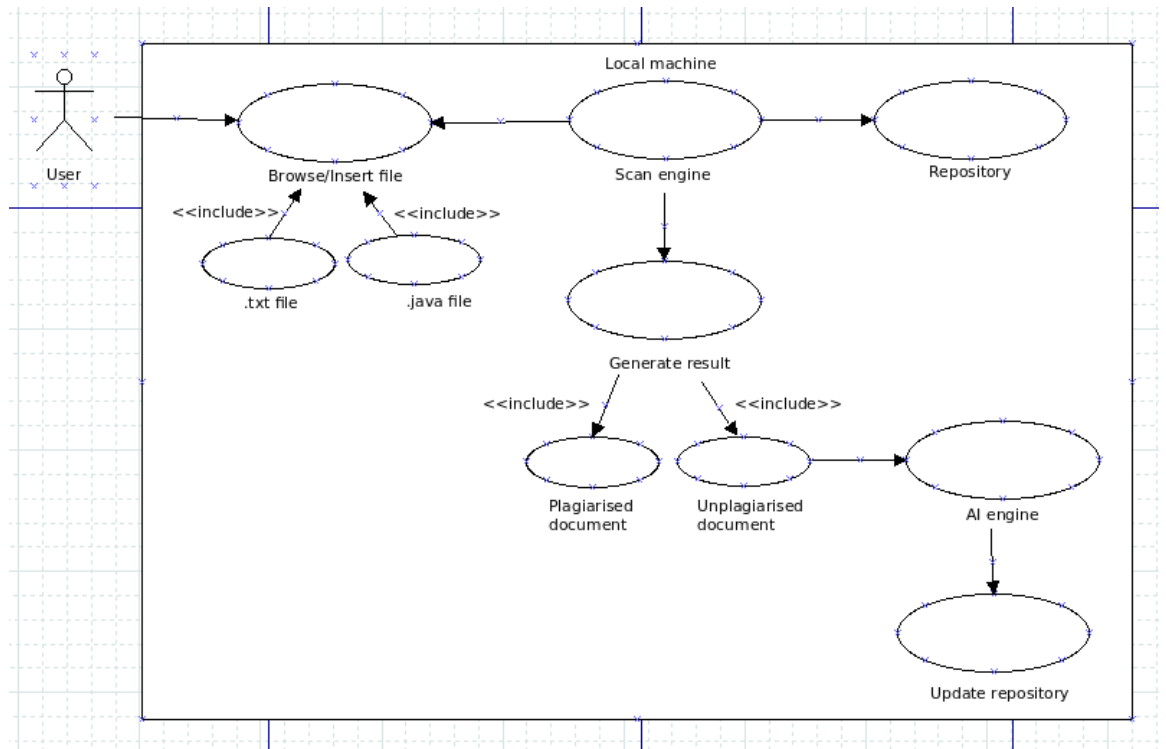
- Data Flow Diagram Level 1



- Flowchart



- Use Case Diagram



### 3.3.3 Implementation Plan

Timeline chart is as follows

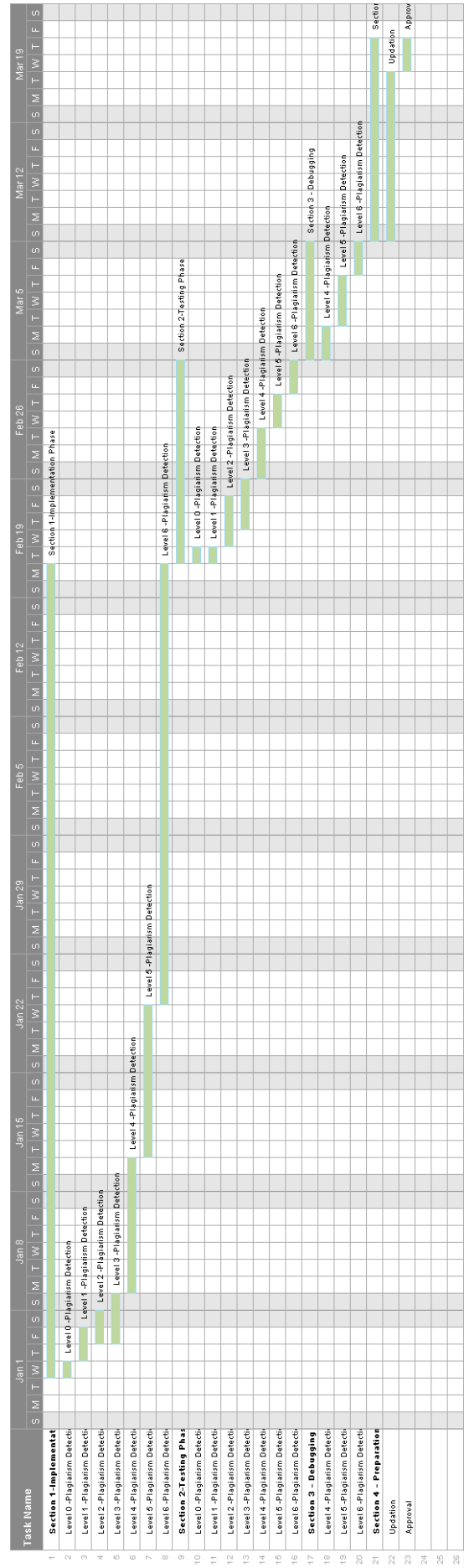


Figure 3.1: Timeline chart  
Department of Information Technology, DBIT, Mumbai

## Chapter 4

# Results and Discussion

### 4.1 Result of Comparison for various levels for Execution time

Two files were given as input and one of these files is plagiarised and the other is source code i.e original.

- Level One Plagiarism Detection.



```
C:\WINDOWS\system32\cmd.exe

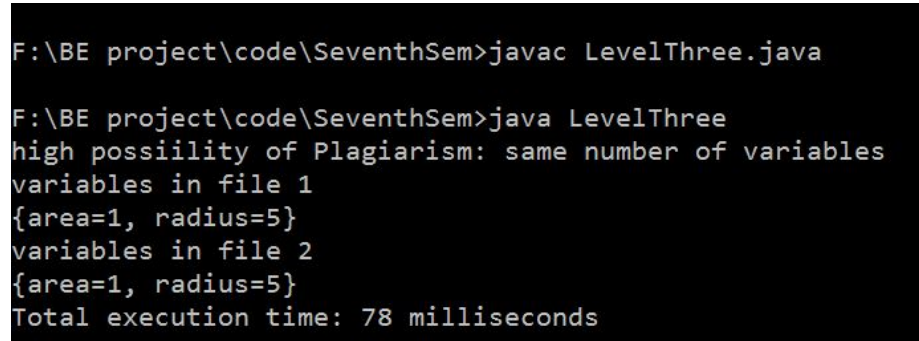
F:\BE project\code\SeventhSem>javac LevelOne.java

F:\BE project\code\SeventhSem>java LevelOne
PLAGIARISED with comments added or changed
Total execution time: 47 milliseconds

F:\BE project\code\SeventhSem>
```

The plagiarised has comments and extra whitespace added. The program correctly identifies that the file has been plagiarised.

- Level Three Plagiarism Detection.



```
C:\WINDOWS\system32\cmd.exe

F:\BE project\code\SeventhSem>javac LevelThree.java

F:\BE project\code\SeventhSem>java LevelThree
high possibility of Plagiarism: same number of variables
variables in file 1
{area=1, radius=5}
variables in file 2
{area=1, radius=5}
Total execution time: 78 milliseconds
```

The plagiarised has changed the names of the variables and shifted their positions around the file. The program correctly identifies that the file has been plagiarised.

- Result from JPlag.

```
F:\BE project\other software>java -jar jplag-2.11.8.jar -l java17 -r "F:\BE project\other software\result" -s "F:\BE project\other software\source"
Language accepted: Java1.7 Parser
Command line: -l java17 -r F:\BE project\other software\result -s F:\BE project\other software\source
Initialize ok
2 submissions
2 submissions parsed successfully!
0 parser errors!

Comparing File1.java-File1Copy.java: 100.0
Writing results to: F:\BE project\other software\result
```

These two files were given as an input to the popular plagiarism detection software tool called JPlag and the results shown by this tool matched the result given by our program.

- Result from MOSS.

#### Moss Results

Thu Oct 13 10:45:16 PDT 2016

Options -l java -m 3

[ [How to Read the Results](#) | [Tips](#) | [FAQ](#) | [Contact](#) | [Submission Scripts](#) | [Credits](#) ]

File 1	File 2	Lines Matched
<a href="#">File1.java (97%)</a>	<a href="#">File1Copy.java (97%)</a>	77

Any errors encountered during this query are listed below.

These two files were given as an input to the popular plagiarism detection software tool called MOSS and the results shown by this tool matched the result given by our program.

## Chapter 5

### Conclusion

SoftPlag is a useful tool to help detect plagiarism in source code. It is able to identify a wide range of plagiarism types which includes comment addition, changing names of variables and changing control flow of program. It can be used by both educational institutes as well as software companies to prevent plagiarism.



# Appendix - I

## Software / System Requirement Specification - IEEE format

### Table of Contents

- 1. Introduction
  - 1.1 Purpose
  - 1.2 Document Conventions
  - 1.3 Intended Audience and Reading Suggestions
  - 1.4 Product Scope
  - 1.5 References
- 2. Overall Description
  - 2.1 Product Perspective
  - 2.2 Product Functions
  - 2.3 User Classes and Characteristics
  - 2.4 Operating Environment
  - 2.5 Design and Implementation Constraints
- 3. External Interface Requirements
  - 3.1 User Interfaces
  - 3.2 Hardware Interfaces
  - 3.3 Software Interfaces
- 4. System Features
  - 4.1 System Feature 1
- 5. Other Nonfunctional Requirements
  - 5.1 Software Quality Attributes
  - 5.2 Business Rules

### 1. Introduction

Plagiarism Detection is the process of finding instances of plagiarism within a work or document. The worldwide and widespread use of computers and Internet has made it easier to plagiarize the work of others. Most cases of plagiarism are found in academia, where source code, art designs, documents like essays, reports etc. So Our Software is use to find plagiarism in source code.

In our plagiarism detection software detection of plagiarism is software-assisted. Software-assisted detection allows vast collections of documents to be compared to each other, making successful detection much more likely. Software assisted reduces effort and time required for plagiarism detection.

### **1.1 Purpose**

Plagiarism means the practice of taking someone else's work or ideas and passing them off as one's own. Plagiarism detection tool is useful to detect plagiarism. Plagiarism are found in colleges, organization or any institute level. So our software is useful for colleges ,Organizations, Institutes to find plagiarism in their respective work area. Our software tool only use to find plagiarism in source code. Our Aim to reduce plagiarism in all field. so everyone will look to think and present their idea or work instead of showing others ideas or work.

### **1.3 Intended Audience and Reading Suggestions**

This Document is intended for developers ,designers ,tester ,project managers, users and documentation writers. Document contains scope of product ,product Perspective ,functions, hardware and software specification ,features of product.

### **1.4 Product Scope**

Plagiarism detection software take user input as an java file and it checks with repository present on local machine and display result. Scope of product is only java file can be check to decide it is plagiarized or not. The source and target program should be in Java.

### **1.5 References**

- Fonte, V. Martins, R. Henriques and D. da Cruz, 1st ed. Portugal, 1998, pp. 147-148.
- Cosma, Georgina. A Thesis Submitted In Partial Fulfilment Of The Requirements For The Degree Of Doctor Of Philosophy In Computer Science. 1st ed. Warwick: N.p., 2008. Print.
- Parker, Alan and James Hamblen. Computer Algorithms For Plagiarism Detection. 2nd ed. Atlanta: N.p., 1989. Print.
- Ali, Asim M. El Tahir and Vaclav Snase. Overview And Comparison Of Plagiarism Detection Tools. 1st ed. Czech Republic: N.p. Web.
- MUFTAH, AHMED JABR AHMED. Document Plagiarism Detection al-

gorithm using semantic network. 1st ed. Malaysia: N.p., 2009. Print.

- Haider, Khurram Zeeshan, Tabassam Nawaz, and Ali Javed. Efficient Source Code Plagiarism Identification Based On Greedy String Tilling. 10th ed. Taxila: N.p., 2010. Web. 23 Aug. 2016.

## **2. Overall Description**

### **2.1 Product Perspective**

Plagiarism detection software is used to check plagiarism in source code in java language only. It is new ,self-contained product. It will able to detect all levels of plagiarism in java programming language. It will implement and extends all features that are not present in other tool and required for plagiarism detection. What Metrics we are consider to check whether file is plagiarized or not those all metrics are important parameter of our product. It will help to reduce plagiarism in an institutes, colleges and organizations.

### **2.2 Product Functions**

In Plagiarism detection software from user system will take java program file to check whether it is plagiarized or not. System will compare it with program in repository and if it is match then it will show that file it plagiarized and if it does not match then system shows file is not plagiarized. Main function of it that it will detect all levels of plagiarism. Effective for detecting plagiarized file.

### **2.4 Operating Environment**

Software Requirement :

Java Runtime Environment

Operating environment –Windows /Linux

Hardware Requirement:

Minimum memory – 512 Mb RAM

Hard disk space-1 GB

### **2.5 Design and Implementation Constraints**

Project Scope is only for java language so to work this tool properly both user file and files which are present in repositories should be in java .

## **3. External Interface Requirements**

### **3.1 User Interfaces**

There will be an screen where user has to upload file for plagiarism detection. So will give an upload button to user to upload any java file an then will compare it with our repository and then display result about it. Result will be shown

in an percentage format based on levels of plagiarism.

### **3.2 Hardware Interfaces**

For this software minimum 512 Mb RAM and 1 GB hard disk is required. It can be use in windows or linux operating systems.

### **3.3 Software Interfaces**

Plagiarism detection software contain repositories which has no of java files. So whenever user gives an input it will compare it with repository files and display result.

## **4. System Features**

### **4.1 Level wise Plagiarism Detection**

#### **4.1.1 Description and Priority**

In our system it will detect plagiarism in user file level by level by checking all plagiarism levels conditions.

Level wise plagiarism detection feature has high priority in our project.

#### **4.1.2 Stimulus/Response Sequences**

Here input from user will take like any java file then system will process it. compare With it repository including all levels of plagiarism. Then result will be display to the user.

#### **4.1.3 Functional Requirements**

To work all functions of the system properly on user system. Java runtime environment should be installed on user machine to run software and to use all functionality .

## **5. Other Nonfunctional Requirements**

### **5.1 Software Quality Attributes**

**Correctness:** When user give an input of an java file to check whether the file is plagiarized or not then system will shows an correct result to user by comparing it with an repository files.

**Reliability:** This will be an simple application even work in an minimum ram and space so it will not crash .

**Performance :**The performance of the system will be optimal. It will be easy to operate for an users.

**Usability:** We will make an software as simple as possible for the users understanding .So it is easy to use an software to any user.

## References

- [1] Zhi Zhou, Member, IEEE, Gonzalo R. Arce, Fellow, IEEE, and Giovanni Di Crescenzo; *Halftone Visual Cryptography*; IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 15, NO. 8, AUGUST 2006
- [2] [D. Fonte, V. Martins, R. Henriques and D. da Cruz, 1st ed. Portugal, 1998, pp. 147-148.
- [3] Cosma, Georgina. A Thesis Submitted In Partial Fulfilment Of The Requirements For The Degree Of Doctor Of Philosophy In Computer Science. 1st ed. Warwick: N.p., 2008. Print.
- [4] Parker, Alan and James Hamblen. Computer Algorithms For Plagiarism Detection. 2nd ed. Atlanta: N.p., 1989. Print.
- [5] Ali, Asim M. El Tahir and Vaclav Snase. Overview And Comparison Of Plagiarism Detection Tools. 1st ed. Czech Republic: N.p. Web.
- [6] MUFTAH, AHMED JABR AHMED. Document Plagiarism Detection algorithm using semantic network. 1st ed. Malaysia: N.p., 2009. Print.
- [7] Haider, Khurram Zeeshan, Tabassam Nawaz, and Ali Javed. Efficient Source Code Plagiarism Identification Based On Greedy String Tiling. 10th ed. Taxila: N.p., 2010. Web. 23 Aug. 2016.

# Acknowledgements

This project was supported by Don Bosco Institute of Technology. We thank our teachers who provided insight and expertise that greatly assisted the project. We thank Prof. Prasad Padalkar for assistance with forming the vision for our project.

We would also like to show our gratitude to the IT department teachers for sharing their pearls of wisdom with us during the course of this project.

( \_\_\_\_\_ )  
(Signature)

( \_\_\_\_\_ )  
(Deven Bhalerao[05])

**Date:**