

Plagiarism Detection Application

Source Code Plagiarism Detection Tool



UMAIR SAJID

HASSAN KHAN

MUHAMMAD UMER ARIF

SUPERVISOR

LEC. MOBEENA SHAHZAD

Submitted to the Faculty of Computing Software Engineering

National University of Sciences and Technology, Islamabad in partial fulfillment

for the requirements of a B.E Degree 2017 in Computer Software Engineering

ABSTRACT

PLAGIARISM DETECTION APPLICATION

When it comes to plagiarism checking possibly, it is the most common part of Educational Life, Research as well as in thesis submission. Since if the Technology of Optical Character Recognition is combined with the plagiarism checker, it will defiantly decrease hugely (huge amount of hard work load) and will help to eliminate plagiarism from the academia. The main objective is to make use of the technology such that all the Hard copies (research) can be plagiarism checked (multiple to be implemented) on papers, codes, thesis, Assignments, research and many more.

Some systems are available for plagiarism detection. Our system will be one of them will facilitate User (Teachers, Students) to get reports of similarity basing on the source code (Academic Tasks). The basic purpose of our system is to check the source code whether It is plagiarized from another source code or not. Our system is hope to be independent, easy to use and easily extensible for other document formats. The system will have source codes (.java) from the corresponding directories which is going to be made by the Users (instructors). Our System have an extended scope to get the source code from the User in the form of picture which is going to be later on read through OCR and after getting the text from the Source code we can check its plagiarism. This will be on the Android Interface of our System.

Furthermore, the System will generate Reports basing on the Similarity in the code.

CERTIFICATE FOR CORRECTNESS AND APPROVAL

Certified that the contents and form of project report entitled “**PLAGIARISM DETECTION APPLICATION**” submitted by

1) PC Umair Sajid, 2) PC Hassan Khan, and 3) NC Muhammad Umer Arif have been found satisfactory for the requirement of the degree.

Supervisor: _____

Lec. Mobeena Shahzad

MCS, NUST

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of

Any other award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent.

To our parents, without whose unflinching support and unstinting cooperation,

A work of this magnitude would not have been possible.

ACKNOWLEDGEMENTS

To begin with, there is no greater guide than **ALLAH (SWT)** Himself and we feel blessed that He gave us enough strength to complete this project well in time.

In addition to this, we all would deeply and genuinely like to thanks **Lec. Mobeena Shahzad** for her persistent guidance and continuous support.

Ma'am you are an exceptional supervisor and without you we could not have come this far.

Table of Contents

1	INTRODUCTION	13
1.1	Purpose.....	13
1.2	Scope.....	13
1.2.1	Extended Scope:	14
1.2.2	Benefits:.....	14
2	LITERATURE REVIEW	16
2.1	Introduction	16
2.2	Limitations of Prior Tools.....	16
2.3	Related Work	17
3.	SOFTWARE REQUIREMENT SPECIFICATION.....	19
3.1	Overview	19
3.2	Overall Description.....	19
3.2.1	Product Perspective	19
3.2.2	User Characteristics	20
3.2.3	Constraints	20
3.2.4	Assumptions and Dependencies	20
3.3	External Interface Requirements.....	20
3.4	Hardware Requirements	21
3.4.1	Software Requirements	21
3.5	System Features	22
3.5.1	Sign Up.....	22
3.5.2	Use Cases.....	23
3.5.3	Login.....	24
3.5.4	Create Directory	24

3.5.5	Upload Files.....	25
3.5.6	Check for Plagiarism	25
3.5.7	View Results.....	26
3.5.8	Store Results	26
3.6	Non-Functional Requirements.....	27
3.6.1	Response Time	27
3.6.2	Reliability	27
3.6.3	Security.....	27
3.6.4	Maintainability.....	27
3.6.5	Portability.....	27
3.6.6	Usability	28
3.6.7	Availability.....	28
4	DESIGN AND DEVELOPMENT:-	30
4.1	Introduction	30
4.2	Purpose of this document	30
4.3	Scope of the project	31
4.4	Benefits:	31
4.5	Objectives & Goals:.....	32
4.6	Definitions, Acronyms, and Abbreviations.....	32
4.7	Overview	32
4.8	System Architecture Description	33
4.8.1	Overview of Modules	33
4.8.2	Structure and relationships.....	34
4.8.3	State Machine Diagram (backend)	36
4.8.4	State Machine Diagrams (frontend).....	38

4.8.5	User Interface Issues.....	38
4.9	Detailed Description of Modules.....	39
4.9.1	Plagiarism Detection Module.....	39
4.9.2	Tokenization.....	41
4.9.3	Algorithm Development.....	41
4.9.4	Report Generation Module.....	42
4.9.5	Database Module.....	44
4.9.6	Desktop Interface Module.....	46
4.10	Detailed Design.....	52
4.10.1	Use Cases.....	52
4.10.2	Activity Diagrams.....	54
4.10.3	Sequence Diagrams.....	55
4.10.4	Class Diagram.....	56
4.11	Reusability.....	56
4.12	Design Decisions and tradeoffs.....	57
4.13	Pseudo Code for Modules.....	57
4.13.1	Plagiarism Detection Module.....	57
4.13.2	Report Generation Module.....	58
4.13.3	Database Module.....	58
4.13.4	Desktop Interface Module.....	58
5	System Implementation.....	60
5.1	Programming Language:.....	60
5.2	Development Tools:.....	60
5.3	Database:.....	60
5.4	Operating System:.....	60

5.5	Complete System Implementation:	60
5.5.1	Java Server Module.....	60
5.5.2	Client Module.....	61
6	System Testing.....	63
6.1	Test Items	63
6.2	Features to be tested	63
6.3	Approach.....	63
6.3.1	Unit Testing	64
6.3.2	Integration Testing.....	64
6.3.3	System Testing.....	64
6.3.4	Item Pass/Fail Criteria	64
6.4	Suspension Criteria and Resumption Requirements.....	65
6.5	Test Deliverables	66
6.5.1	Test Case 1	66
6.5.2	Test Case 2	67
6.5.3	Test Case 3	68
6.5.4	Test Case 4	69
6.5.5	Test Case 5	70
6.5.6	Test Case 6	71
7	Future Work Conclusion	73
	BIBLIOGRAPHY	74

List of Figures

Figure 1-Use case Diagram	23
Figure 2- Complete Diagrammatic view	34
Figure 3-High Level Design	36
Figure 4-State Machine Diagram (Backend)	37
Figure 5 State Machine Diagram (Frontend)	38
Figure 7-Screen 1: Login Screen	47
Figure 8-Screen 2: Registration Page	48
Figure 9-Screen 3: Home Screen.....	49
Figure 10-Screen 4: Report.....	50
Figure 11-Screen 5: Code Highlight	51
Figure 12 -Screen 5: History Tab	52
Figure 13-Usecase Diagram	53
Figure 14-Activity Diagram: Plagiarism Detection Application	54
Figure 15-Sequence Diagrams 1: Plagiarism Detection Application	55
Figure 16-Class Diagram	56

List of Tables

Table 1 Use Case 1: Sign Up.....	22
Table 2 Use Case 2: Login.....	24
Table 3 Use Case 3: Create Directory.....	25
Table 4 Use Case 4: Upload Files.....	25
Table 5 Use Case 5: Check for Plagiarism.....	25
Table 6 Use Case 6: View Results.....	26
Table 7 Use Case 7: Store Results.....	26
Table 8 Plagiarism Detection Module.....	41
Table 9 Report Generation Module.....	43
Table 10 Database Module.....	45
Table 11 Desktop Interface Module.....	47
Table 12 Test Case 1.....	66
Table 13 Test Case 2.....	67
Table 14 Test Case 3.....	68
Table 15 Test Case 4.....	69
Table 16 Test Case 5.....	70
Table 17 Test Case 6.....	71

CHAPTER 1
INTRODUCTION

1 INTRODUCTION

When it comes to plagiarism checking possibly, it is the most common part of Educational Life, Research as well as in thesis submission. Since if the Technology of Optical Character Recognition is combined with the plagiarism checker, it will defiantly decrease hugely (huge amount of hard work load) and will help to eliminate plagiarism from the academia. The main objective is to make use of the technology such that all the Hard copies (research) can be plagiarism checked (multiple to be implemented) on papers, codes, thesis, Assignments, research and many more.

1.1 Purpose

Some systems are available for plagiarism detection. Our system will be one of them will facilitate User (Teachers, Students) to get reports of similarity basing on the source code (Academic Tasks). The basic purpose of our system is to check the source code whether It is plagiarized from another source code or not. Our system is hope to be independent, easy to use and easily extensible for other document formats. The system will have source codes (.java) from the corresponding directories which is going to be made by the Users (instructors). Our System have an extended scope to get the source code from the User in the form of picture which is going to be later on read through OCR and after getting the text from the Source code we can check its plagiarism. This will be on the Android Interface of our System.

Furthermore, the System will generate Reports basing on the Similarity in the code.

The following subsections introduce the software design for a Text Analysis Tool, known as PLAGIARISM DETECTION APPLICATION, to its readers. This document contains architectural level diagrams accompanied by other low level design diagrams. The aim of this document is to enumerate the main components, and describe the relationships and dependencies between them to carry out the various functionalities of the system.

1.2 Scope

The project consists of two parts:

Desktop Application named as “Plagiarism Detection Software” will let the User create directories and upload the source code into those directories. It will compare the source codes according to the algorithms. After that it will provide the interface in the Desktop application for the Users to create directories and check for plagiarism and get a report.

1.2.1 Extended Scope:

Android Application named as “Plagiarism Detection Software” will let the User’s take pictures from their android device. And OCR Mechanism will extract the text (source code) from that picture and analyze computerized format codes. After that it will provide the same interface as the Desktop application for the Users to create directories and check for plagiarism and get a report.

1.2.2 Benefits:

The main advantages that could be achieved from this system are:

- It will provide the opportunity to check the originality and uniqueness of the source codes.
- User can see the percentage of similarity between the codes.
- Users (Teachers, Students) can get proof that their codes are not plagiarized.
- It can let the users develop more efficiently and with the similarity check report.
- Checking Hard copy of source codes are a new way for the users to use it easily.

CHAPTER 2
LITERATURE REVIEW

2 LITERATURE REVIEW

2.1 Introduction

A source code plagiarism detection engine **Plagiarism Detection Application** is presented. It is a stand-alone Java application that can be used to check Java programming exercises. Java is a common language for introductory programming courses. We find out that 56% of accredited undergraduate computer science programs in the Pakistan expected to use Java as their first language during the academic year 2015–2016. Several source code plagiarism detection engines exist (Section 2.2). After the proposal defence, we selected JPlag and MOSS out of seventeen engines to be tested more thoroughly. We found JPlag as being the most promising for the needs of our laboratory. However, JPlag could not make a difference between (a) program code that was programmed by students and (b) program code that was distributed to them as a part of an exercise. This was a problem because one of our laboratory's programming courses used heavily exercises where some program code was distributed to students. Therefore, we decided to program a JPlag-like detection engine. This shortcoming is solved in the current version of JPlag. Thus, the most important or the only contribution of "**Plagiarism Detection Application**" is that its source code is open (see Section 2).

2.2 Limitations of Prior Tools

JPlag or MOSS is probably the most suitable alternative for most computer science departments but **Plagiarism Detection Application** might be more suitable in the following situations:

(a) A computer science department does not want to take a slightest risk that confidential information about cheating is sent outside the institution by mistake. For example, a teaching assistant might forget to remove student names or identification numbers from the submissions before sending them to a web service.

(b) Someone wishes to develop **Plagiarism Detection Application** further or integrate it with other computer-assisted or computer-managed instruction software already in use.

(c) Someone wishes to compare different source code plagiarism detection engines in detail using white-box testing instead of black-box testing.

2.3 Related Work

Plagiarism Detection System is by far not the first system built for finding plagiarisms. This section discusses some of the previous ones.

Feature comparison. The early attempts at plagiarism detection are usually based on the notion of a feature vector. These systems compute for each program a number of different software metrics, so that each program is mapped to a point in an n -dimensional cartesian space. The systems then consider sets of programs that lie close to each other to be possible plagiarisms.

The earliest such system we could find is Ottenstein's from 1976. It uses only the basic Halstead metrics (number of unique operators U_1 , number of unique operands U_2 , total number of operators N_1 , total number of operands N_2) on Fortran programs and considers only programs to be plagiarisms where all four values coincide.

Later systems such as those of Donaldson and Lancaster introduce a much larger number of metrics and notions of similarity for the resulting feature vector in order to improve performance.

Unfortunately, these efforts are only moderately successful. Systems based on feature vectors can hardly have good performance, because summing up a metric across the whole program simply throws away too much structural information. Note that this deficiency cannot be removed by adding further dimensions to the comparison: Such systems will either be very insensitive (and hence easy to fool by simple program transformations) or will be sensitive and will come up with a large fraction of false positives.

CHAPTER 3

Software Requirement Specification

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Overview

The remaining sections of this document will cover an overall description of the project. The requirements and specifications for the software application are discussed in detail. The overall description of the software can be useful for the customers or users to determine whether the requirements specified fulfill their requirements and development team as well in the development of the product.

Here is an overview of the remaining section of this section:

- **Overall Description** has the detailed description of the product in general. This section covers what the product is and how it can be used for practical applications.
- **External Interface Requirements** specifies the requirements for external interfacing for the application.
- **System Features** mainly specifies the functional requirements. The system functionality depends on this section.
- **Non-Functional Requirements** covers the non-functional requirements like system reliability, portability etc.

3.2 Overall Description

This section has detailed description about the project. Basic features and constraints are discussed.

3.2.1 Product Perspective

The product “Plagiarism Detection Application” is aimed to provide information about the originality and uniqueness of the source code to the User through a Desktop or Android App.

There is also an Android App for targeted Users. Android App will be connected to Desktop Application using Interfacing. The main purpose of having an Android Application

is to provide the easiness to the user and being a handy tool for taking picture from the camera and Scan Hard Copies of source code. The process of taking picture and then extracting the text from it using OCR algorithms.

The plan to carry out this project consists of three main tasks:

1. Make an algorithm to get plagiarism reports from the source codes in directories.
2. Implementing OCR Algorithms to extract texts from pictures.
3. Interfacing Desktop and Android App.

3.2.2 User Characteristics

The Users having accounts will have access to earlier created directories and also to create new directories and upload source code into those directories. Our System will simply access those source code from the directory and later on apply certain algorithms written in Java Specifically using Greedy String Tiling GST Algorithm and Tokenization to get the similarity. This algorithm is actually based on semantics of those codes. After that System will show the results to the User in the form of Report showing a Similarity Index.

3.2.3 Constraints

Student names submitting the plagiarism is not on the end of the Instructor but it's on the end of the student himself. While submitting the code he should always submit a comment in the first line of code containing his name and ID.

3.2.4 Assumptions and Dependencies

An assumption is that the user has to submit **java** files to detect the plagiarism and the files must compile without any errors.

3.3 External Interface Requirements

This section covers the interfacing requirements for our project. Features of user interfaces are described.

3.4 Hardware Requirements

The application is intended to be a two-interface project with a Java Server to be run only while application is running. The application will be a desktop application. No further hardware devices or interfaces will be required.

3.4.1 Software Requirements

3.4.1.1 User Interfaces

Firstly, in order to view the real-time processing on Server, there shall be a desktop application based on Java 8.0. This application shall be compatible with Windows 7 and latest. The application will show the processing and it will be connected to database.

Secondly, the android based application shall be an android form based java application, which shall provide a graphical user interface for user friendly environment. All the data entry shall be done with keypad and the user can navigate between menus/forms/screens with the help of navigation keys.

Finally, the user interface for the android application of the System, shall be compatible to all android device but for best user experience the following versions are preferable

- Jelly beans 4.1.2
- ICS 4.1.1

3.4.1.2 Communication Interfaces

For communication between Server and Frontend Client Application or Android App, UDP and FTP Sockets protocols will be used. Each Android device will be identified by its unique IP address.

The android application shall use Google APIs or Android OCR Libraries to convert the Images into the text for User.

3.4.1.3 Output Interfaces

The output to the user would be displayed graphically in a clear and meaningful manner which the user can easily understand.

3.5 System Features

System features are discussed in detail. This heading mainly covers the functional requirements of the project.

3.5.1 Sign Up

Description	New Users are required to register as members before they are allowed to use Plagiarism Detection System.
Priority	High
Stimulus/Response	Once the user pressed “Sign Up” Button. A New Screen will be Shown in which users can enter their details and request for username and password.
Functional Requirement	<ol style="list-style-type: none">1. Enter the Details for any user2. Users are given the choice to create there username and passwords.3. Writing into the Database4. Once Signed Up successfully the user can login and use the plagiarism detection system.

Table 1 Use Case 1: Sign Up

3.5.2 Use Cases

Plagiarism Detection Application

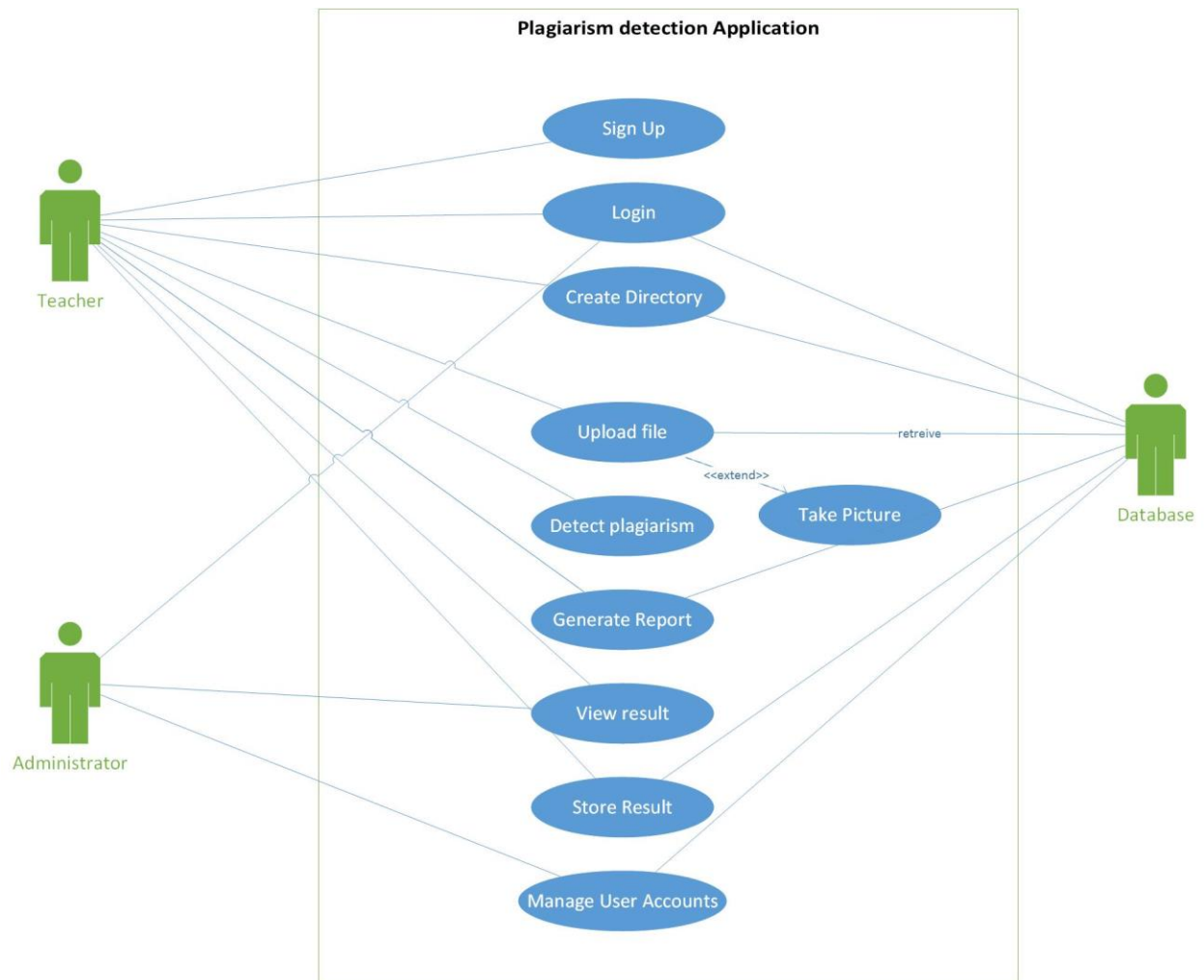


Figure 1-Use case Diagram

3.5.3 Login

Description	Users who have been already signed up, can login into the system by providing the username and password.
Priority	High
Stimulus/Response	The User will enter the details into “Username and Password” Text Fields and press the Login button and will be logged into the system.
Functional Requirement	User needs to provide the Username and password. The System will detect the origin of the username and password and verify from the database. The User providing wrong details will be shown an error message and user will have to re-enter the username and password again. User can logout, after logging in from the system and its settings and files will be saved against their account.

Table 2 Use Case 2: Login

3.5.4 Create Directory

Description	Users will be allowed to create, maintain and updates the directory in the system.
Priority	High
Stimulus/Response	Users will enter the location of created directory or Will choose the directory from File Chooser.
Functional Requirement	User will enter the Directory Path. User can add any directory in the system. User can delete any Directory in the system.

User can also update the directory.

Table 3 Use Case 3: Create Directory

3.5.5 Upload Files

Description	Users will have to upload the code into the directories.
Priority	High
Stimulus/Response	Users will choose the directory path and then upload the codes into that path and the Codes will be saved into the directory.
Functional Requirement	Users will choose the directory. Users will upload the files in the directory.

Table 4 Use Case 4: Upload Files

3.5.6 Check for Plagiarism

Description	The System will read the codes from the given directories of the user and Apply certain Algorithms “GST (Greedy String Tiling) and Tokenization” to check the plagiarism of the codes.
Priority	High
Stimulus/Response	The User will click on the “Check plagiarism” button.
Functional Requirement	The System will Read the Codes from directories. The System will then check the plagiarism using defined algorithms in Java. The System will generate the Similarity Index in the Reports of the Plagiarism.

Table 5 Use Case 5: Check for Plagiarism

3.5.7 View Results

Description	The System will read the codes from the given directories of the user and Apply certain Algorithms “GST (Greedy String Tiling) and Tokenization” to check the plagiarism of the codes.
Priority	High
Stimulus/Response	The User will click on the “Check plagiarism” button.
Functional Requirement	<p>The System will Read the Codes from directories.</p> <p>The System will then check the plagiarism using defined algorithms in Java.</p> <p>The System will generate the Similarity Index in the Reports of the Plagiarism.</p>

Table 6 Use Case 6: View Results

3.5.8 Store Results

Description	The System will save the results of the plagiarism in the database.
Priority	High
Stimulus/Response	Users can retrieve the results from the database and the system will show the report.
Functional Requirement	<p>The system will save the result into the database in the form of reports.</p> <p>The User can request to show the report against any plagiarism checked before.</p>

Table 7 Use Case 7: Store Results

3.6 Non-Functional Requirements

Non-Functional Requirements are very critical in success of a project. This section enlists the non-functional requirements for our project.

3.6.1 Response Time

Response time of Server on request from Android App must not exceed 30 sec.

Android App should not take more than 15sec in taking a picture and converting it into the text and send it to Server.

3.6.2 Reliability

- The system shall be available and online 24/7, with no more than 3 hours of downtime per week for maintenance.
- There should be a contractual agreement with an internet service provider for T3 access with 99.99% availability for the main server.
- There should be an alternate android device, such that in case of failure, within a time span of 2 hours alternate android device should be up and operational.
- Backup power source (external battery) shall be provided in order to ensure maximum availability of the system.

3.6.3 Security

The system shall use sockets in all transactions that include any confidential user information also images, directory information and files.

3.6.4 Maintainability

The software should be written clearly and concisely. The code will be well documented. Particular care will be taken to design the software modularly to ensure that maintenance is easy.

3.6.5 Portability

- The application shall be scalable for maximum possible Users.

- Data of users older than 5 years files and codes shall be discarded for better system performance.
- System shall be flexible enough to cater future modifications such as checking plagiarism for different codes .cpp,.cs,.php etc.
- System shall be modifiable if more features need to be added.

3.6.6 Usability

- The system shall provide a uniform look in all application pages.
- The system shall provide colored GUI for better visualization.
- The android application shall provide colored icons and menu bars for navigation and better visualization and user experience.
- Navigation between different menus and screens shall also be possible using keyboard commands and shortcuts.
- The system shall only provide single language support i.e. English

3.6.7 Availability

- The system shall be available and online 24/7, with no more than 3 hours of downtime per week for maintenance.
- There should be a contractual agreement with an internet service provider for T3 access with 99.99% availability for the main server.
- There should be an alternate android device, such that incase of failure, within a time span of 2 hours alternate android device should be up and operational.
- Backup power source (external battery) shall be provided in order to ensure maximum availability of the system.

CHAPTER 4

Software Design Specification

4 DESIGN AND DEVELOPMENT:-

4.1 Introduction

Some systems are available for plagiarism detection. Our system will be one of them will facilitate User (Teachers, Students) to get reports of similarity basing on the source code (Academic Tasks). The basic purpose of our system is to check the source code whether It is plagiarized from another source code or not. Our system is hope to be independent, easy to use and easily extensible for other document formats. The system will have source codes (.java) from the corresponding directories which is going to be made by the Users (instructors). Our System have an extended scope to get the source code from the User in the form of picture which is going to be later on read through OCR and after getting the text from the Source code we can check its plagiarism. This will be on the Android Interface of our System.

Furthermore, the System will generate Reports basing on the Similarity in the code.

The following subsections introduce the software design for a Text Analysis Tool, known as **Plagiarism Detection Application**, to its readers. This design document contains architectural level diagrams accompanied by other low level design diagrams. The aim of this document is to enumerate the main components, and describe the relationships and dependencies between them to carry out the various functionalities of the system.

4.2 Purpose of this document

This Software Design Document enumerates all the components of **Plagiarism Detection Application**. The functionality of each and every module is explained. The dependencies and relationships between different modules or components are also stated in the document. Conventional visual demonstrations from software industry are used wherever necessary.

The document plays the role of a reference guide to development of the entire project to its audience. The intended audience includes the following stakeholders:

- Development Team
- Supervisor
- Evaluating Team
- General Public
- Instructors

4.3 Scope of the project

The project consists of two parts:

Desktop Application named as “Plagiarism Detection Software” will let the User create directories and upload the source code into those directories. It will compare the source codes according to the algorithms. After that it will provide the interface in the Desktop application for the Users to create directories and check for plagiarism and get a report.

Extended Scope:

Android Application named as “Plagiarism Detection Software” will let the User’s take pictures from their android device. And OCR Mechanism will extract the text (source code) from that picture and analyze computerized format codes. After that it will provide the same interface as the Desktop application for the Users to create directories and check for plagiarism and get a report.

4.4 Benefits:

The main advantages that could be achieved from this system are:

- It will provide the opportunity to check the originality and uniqueness of the source codes.
- User can see the percentage of similarity between the codes.
- Users (Teachers, Students) can get proof that their codes are not plagiarized.
- It can let the users develop more efficiently and with the similarity check report.
- Checking Hard copy of source codes are a new way for the users to use it easily.

4.5 Objectives & Goals:

- Develop a cost-effective solution for plagiarism checking in existing system.
- Will Provide Easy to Use Interface for the Users (Desktop and Android)
- Get source codes from the Pictures and compare the similarities between the codes.
- System will check the plagiarism between uploaded files.
- Generate Reports basing on the similarity.
- Establish a connection with Users of Android App.

4.6 Definitions, Acronyms, and Abbreviations

Here are a list defining the difficult terms and abbreviations used in the document:

- **Plagiarism Detection:** Source Code Plagiarism Detection
- **SQL:** Structured Query Language
- **GUI:** Graphical User Interface
- **HTTP:** Hyper Text Transfer Protocol

4.7 Overview

This section gives an overview of the remaining sections in this document. Section 2 comprises of system architecture description. It contains the overview of working modules. Complete overview of the project is demonstrated using a complete diagrammatic view of the system. A high level design diagram is also used to visualize the structure within the system as well as individual modules. State machine diagrams are also used to show the basic workings in the project. Low level states using state machine diagrams are given with backend processing perspective as well as users' perspectives.

Section 3 has detailed description of each module in the system. Each submodule is described as well. Diagrams are used wherever necessary for better understanding. Sketches for output screens are included in the subsection of web interface. Section 4 covers low level design diagrams like use cases, ER diagrams, sequence diagrams, and activity diagrams. Section 5 contains information about reusable components in the system. There are two subsections in this section. One section has information about components that have been reused in the system while the other focuses on the reusability of components developed in the project. Section 5 has information that explains the design decisions taken in development of project. It also explains the tradeoff between two approaches to web interface development. Section 6 has pseudo code for modules and their interfaces.

4.8 System Architecture Description

This section covers the brief discussion of each module and its relationships with the other modules in the system. Graphical representations are used for better understanding as well.

4.8.1 Overview of Modules

Based on functionality, there are four main modules in the PLAGIARISM DETECTION APPLICATION project. Section 3 has detailed description of each module including definition of subcomponents and their functionality. This section briefly states the overall functionality of each module in the system.

The four modules are as follows:

- **Plagiarism Detection Module:** This component of the system makes use of different algorithms and checks for plagiarism in the source code.
- **Report Generation Module:** Report Generation Module uses the output from the **plagiarism detection module**. Different processing algorithms are then used to

generate reports with the results of uniqueness. Percentage is shown as the final result. The results are stored in the databases.

- **Database Module:** The database acts as a bridge between the backend processing modules and the frontend desktop and android interface. The files which are uploaded in the form of source code and the results are stored in the database.
- **Desktop Application Interface Module:** The web interface retrieves data from the database and updates the output screens. There will be at least one common welcome screen and different output screens depending upon the type of users.

Here is a diagram showing the complete diagrammatic view of the project using the modules, their sub modules and the relationships to show the flow of the system.

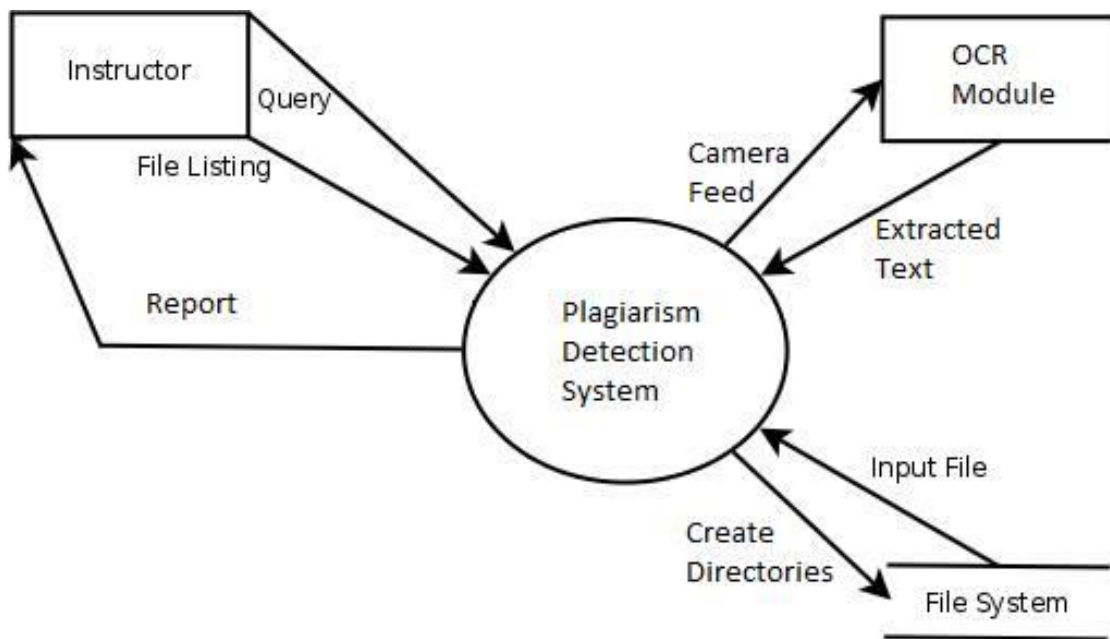


Figure 2- Complete Diagrammatic view

4.8.2 Structure and relationships

Each module in the system is properly interfaced for communication with other modules in the system. The modules are interconnected and depend upon each other for complete functionality. For example, Report generation module cannot start processing unless Plagiarism Detection module has passed some structured results to it. If Report generation module has not processed any source code, database will not have any entries and consequently, the desktop interface would have nothing to display. Hence, the functionality from each module is partially dependent on other modules.

The various modules and the submodules are displayed in a structural format using the following high level design diagram.

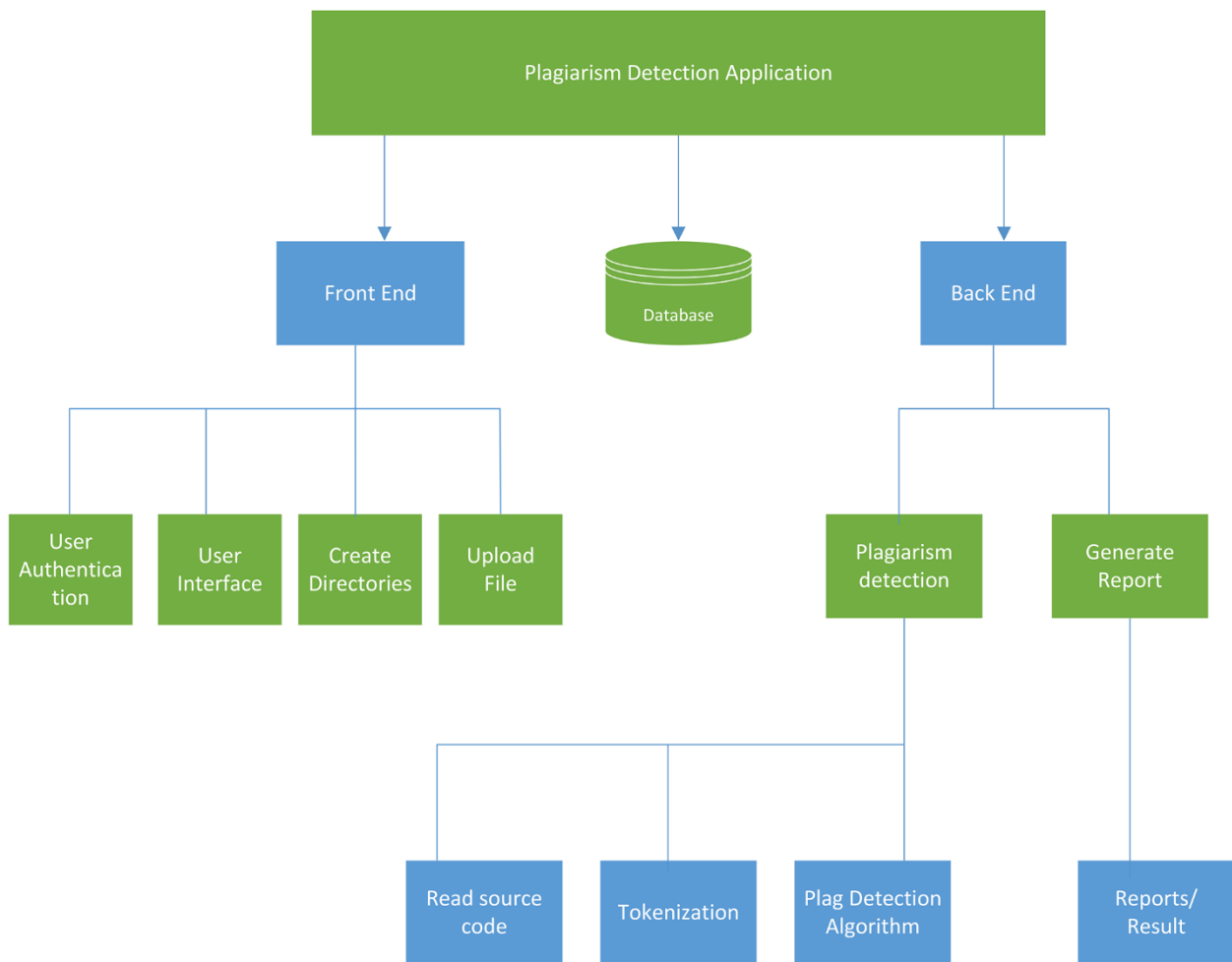


Figure 3-High Level Design

Following two subsections contain three state machine diagrams which shed some more light on the structure and relationships between components. For the sake of simplicity, backend processing and frontend interface have been demonstrated in separate diagrams.

4.8.3 State Machine Diagram (backend)

The main model of computations is constructed at the backend and involves the two modules, Plagiarism Detection module and Report generation module. Database module is also used for storing the results from report generation module.

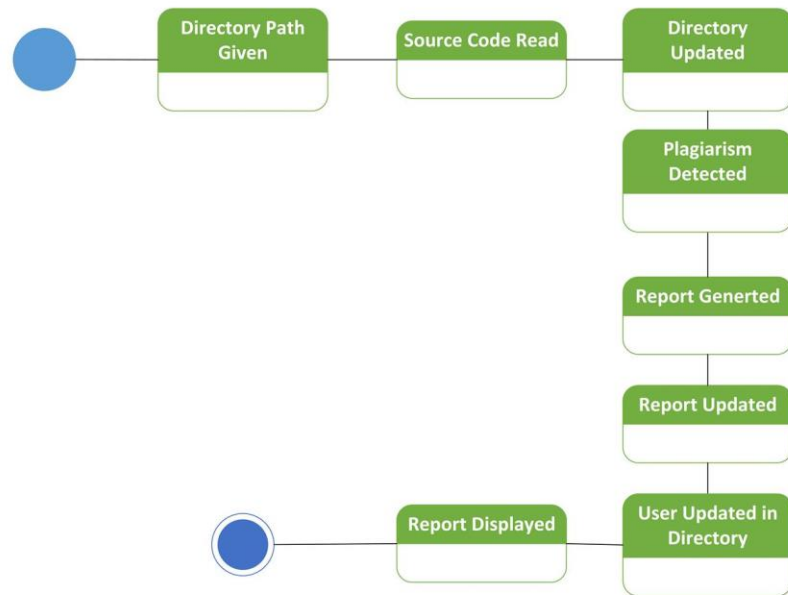


Figure 4-State Machine Diagram (Backend)

The state machine diagram in above figure demonstrates the transition of states to carry out the backend processing. Backend processing goes through each state to establish the model which will be used by desktop interface. Let us describe functionality with respect to each state:

4.8.4 State Machine Diagrams (frontend)

State machine diagrams are used to represent the states users (Instructor) go through when using PLAGIARISM DETECTION APPLICATION.

The following diagram shows the state machine diagram in case of authenticated user:

Here are the description for each state involved in the above diagram:

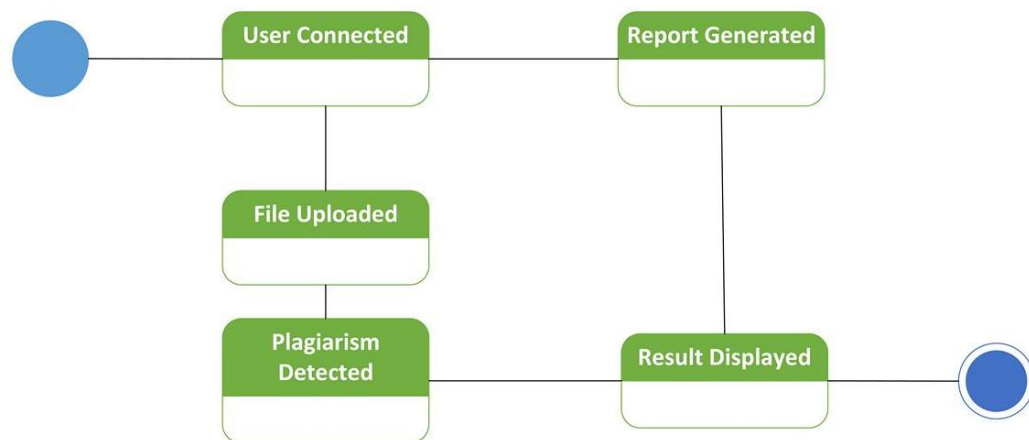


Figure 5 State Machine Diagram (Frontend)

4.8.5 User Interface Issues

Java GUI is used for interaction of users with the system. There are at least 4 screens involved for both types of users. The home screen is the login screen, which has two buttons and two text fields. The two buttons are used to login the user. Once, the user has entered the valid credentials, the authenticated user is directed to plagiarism detection page. Users are directed to the plagiarism detection screen where files are displayed along with their plagiarism report. Section 3.4 has detailed description for each of these output screens along with visual demonstrations.

The plagiarism detection module passes the results from its computations to report generation module. The database module is then populated with results from Report generation module. Java GUI extracts information from database and displays it to the users. Desktop Interface and Database modules also interact for authentication of users. Users share the same database for retrieving the required information on output screens.

4.9 Detailed Description of Modules

This section comprises of detailed descriptions of each module. The submodules of each module are described as well. The inputs and outputs associated with each module are discussed as well. Interfaces between the modules, however, have been demonstrated in the section 2.2.

4.9.1 Plagiarism Detection Module

Plagiarism Detection Module deals with the source code and uploading the source code from the user. Source code can be in Java language. The module identifies and uses the source code in Java and which is then used for getting results of plagiarism by Report generation module.

This module performs following major tasks:

1. Getting Source code from the database module.
2. Checking the source code format.
3. Extraction of tokens components like identifiers, keywords and structures.
4. Passing the results to Reports Generation Module.

To keep the functionality modular, the various functionalities of the module are divided into multiple subcomponents. The following subsections describe the subcomponents which work together for complete functionality of this module.

Identification	Plagiarism Detection Module
Type	A module
Purpose	The purpose of this module is to check the source code and its format. Tokenize it and then pass it to the algorithm which will result into the checking the similarity between the files and later on will pass the files to report generation module.
Function	This module checks for plagiarism by tokenizing the code and then applying the specific algorithm to get results of uniqueness.
Subordinates	Tokenization, Algorithm Development
Dependencies	Source code is taken as an input. The output of this module is used in the report generation module as input.
Interfaces	External Interfaces are required for interaction between the module and report generation module. An external interface is required for retrieving the source code from the files as well. Internal interfaces include the message passing between Tokenizer and the Plag Checker Classes and Results components for checking the uniqueness.
Resources	Resources required include the source code with user identifiers. Checking Plagiarism demands fast CPU execution times for efficiency.

Processing Pseudo code for the module is given in section 7.

Data Source code will be in the format of .java . The output from the module is the data in structured format.

Table 8 Plagiarism Detection Module

4.9.2 Tokenization

- Parsing & Pre-tokenization stages
- Read source code
- Remove strings
- Remove comments
- Separate statements
- Generate tokens which include:
 - 1) Identifiers or Symbols (variables, types, functions, and labels)
 - 2) Keywords (these are language reserve words e.g. for, while, if etc for Java language)
 - 3) Literals (these are constants)
 - 4) Operators (e.g. +, -, / etc)
 - 5) Punctuators (these have syntactic and semantic significance for the compiler e.g. and, not, bitand, xor etc for Java Languages)

4.9.3 Algorithm Development

This component is dedicated to the development of algorithm for evaluating the opinions contained in the messages. This algorithm is used for report generation. The inputs to the algorithm are the text messages in structured format and the outputs are scores indicating the level of sentiments involved in them. (An algorithm) application stages:

- Input Tokens
- Apply Certain Algorithms
- Obtain Results

Some other results are included which are to detect “false positive clone detection”. Method is to count the token if found consecutive e.g. for “consecutive if” the pseudo code given at the end.

4.9.4 Report Generation Module

This module carries out the core functionality of the project, sentiment analysis. The input into the system is the structured textual data in the form of report. The results are stored in the database.

Identification	Report Generation Module
Type	A Module
Purpose	To carry out the main functionality of the system by generating the reports of the plagiarism.
Function	Generating the reports in the forms of HTML from the algorithm in the plagiarism checking module.

Subordinates	Algorithm Development, Report Generation, and Database updating.
Dependencies	The module depends directly on the output from plagiarism detection module as the algorithms will work with structured data received from the text processing module.
Interfaces	External interfaces are required to receive input from plagiarism detection module and save the outputs in the database. Internal interfaces are developed for communications between the subordinates of the module.
Resources	Resources required for this module to work are only the outputs from text processing module.
Processing	There are different components for Plagiarism Detection. Pseudo code for the whole module is included in section 7.
Data	The input data is in structured format and the output data comprises of keywords and tokens. The outputs are saved in the database using proper interfaces.

Table 9 Report Generation Module

Following subsections cover the subcomponents which work together to complete the functionalities of this module.

4.9.4.1 Report Generation

This component uses the sentiments extracted and uses a threshold value to detect the messages that could be possible threats. The higher intensity of negativity is considered as threatening, and a separate dictionary can also be developed for detection of threats.

4.9.4.2 Update the database

This component plays the role of an interface from Report generation module to the database module. The results from computations are stored in the database. Entities like users, files, directories, reports, and others are updated.

4.9.5 Database Module

Database module is designed to act as a bridge model between the backend processing modules and front end graphical web interface. The information to be displayed is stored in the databases after backend processing. The database is accessible by the web interface for tasks like user authentication, data visualizations, and threat detection.

Identification	Database Module
Type	A module
Purpose	The output from sentiment analysis module has to be saved in a database, from where it can be uploaded onto the web interface.
Function	The core functionality of this module is to design a database solution for the application. Proper ER diagrams are used for designing the solution and interfaces are provided for interaction with the modules used for inputs and outputs.
Subordinates	Entities and their attributes along with the relationships.

Dependencies	The database is populated with results from sentiments and threats identified from sentiment analysis module. Hence, there is a direct between these two modules. The web interface is dependent upon this module as the database acts as an information model.
Interfaces	External interfaces are required for interactions with web interface and sentiment analyzer. Internal interfaces are simpler as database solution has already been devised as represented through the ER diagrams in section 4.2.
Resources	Resources required for this module are an SQL database server.
Processing	Pseudo code for the module is given in section 7.
Data	Data is stored in tables and are accessible through web interface.

Table 10 Database Module

The database will comprise of following main entities:

- Users
- Raw source Code
- Reports

Authenticated users have login details which have to be stored in the database. The desktop interface queries this entity for validation of users. New entries can only be entered manually by the system administrators.

Users are the users whose messages are being used in the development of working model. It will have attributes like user ID, username, and location. Each message will be associated with two users, one as sender, and the other one as receiver.

4.9.6 Desktop Interface Module

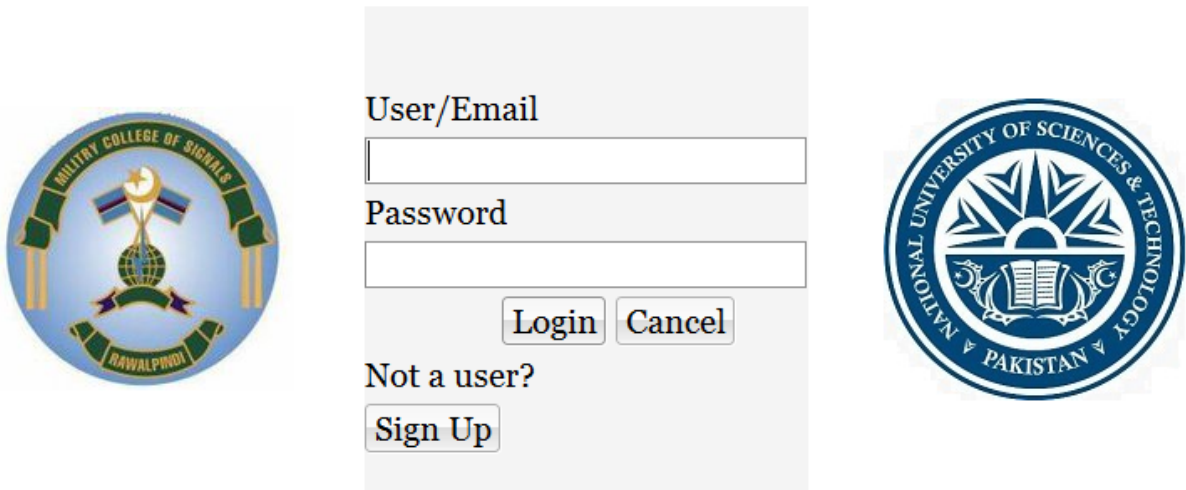
This module is the only module that interacts with the users directly. The web interface comprises of a website that can be accessed by public users and authenticated users. Website is the frontend module in the system, which is used to visualize the output from the two backend modules, Plagiarism Checker and Report Generation via the Plagiarism Checker Module. This is demonstrated in the overall architecture of the system diagram as well.

Identification	Desktop Interface Module
Type	A module
Purpose	This module is for the users to interact with the application.
Function	Develops a desktop interface which displays the output from backend processing to the users via a central database.
Subordinates	Multiple pages or screens.
Dependencies	This module directly uses the database to access the computational results. Database is updated by the two backend processing modules.
Interfaces	External Interfaces include the procedures for extracting information from databases and the webpages for interacting with the users for inputs and outputs. Internal Interfaces include how different webpages are interconnected.
Resources	Resources required for this module are a sql server that can access our database module.
Processing	Pseudo code for this module is given in section 7.4.
Data	The data required by the module is contained in the database. The database has to be populated before the application is

uploaded. User inputs taken at runtime such as search queries and getting threat details also interact with the database.

Table 11 Desktop Interface Module

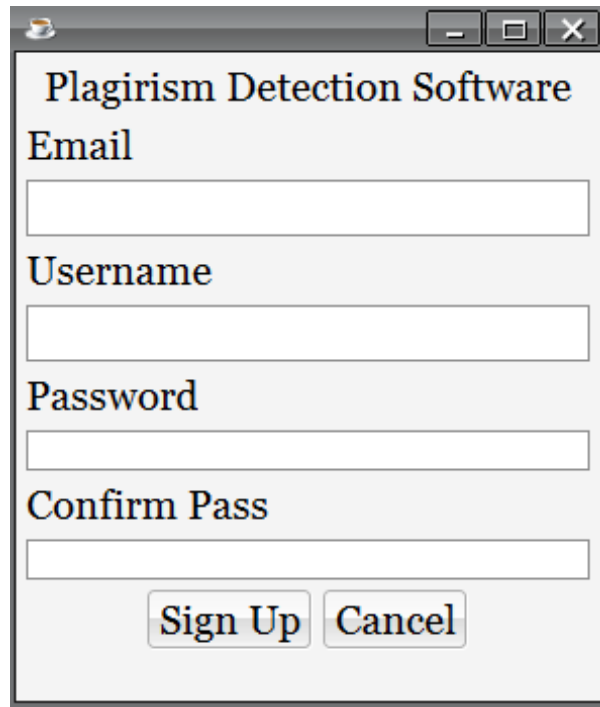
There are at least four tabs involved for users to complete the functionality of desired system. However, more pages may be added in the future. Following figures represent the sketch for various screens involved in the system. First of all, all users are taken to the login page. The user type is selected by user using the available choices on the page and are redirected to different pages depending upon their choice.



The image shows a login screen interface. On the left is the logo of the Military College of Signals, Rawalpindi, featuring a globe and a crescent moon. On the right is the logo of the National University of Sciences & Technology, Pakistan, featuring a book and a sun. The central form contains the following elements:

- A label "User/Email" above a text input field.
- A label "Password" above a text input field.
- Two buttons: "Login" and "Cancel".
- A label "Not a user?" above a "Sign Up" button.

Figure 6-Screen 1: Login Screen



A registration dialog box titled "Plagirism Detection Software". The dialog box has a standard Windows-style title bar with minimize, maximize, and close buttons. It contains four text input fields, each preceded by a label: "Email", "Username", "Password", and "Confirm Pass". At the bottom of the dialog box, there are two buttons: "Sign Up" and "Cancel".

Figure 7-Screen 2: Registration Page

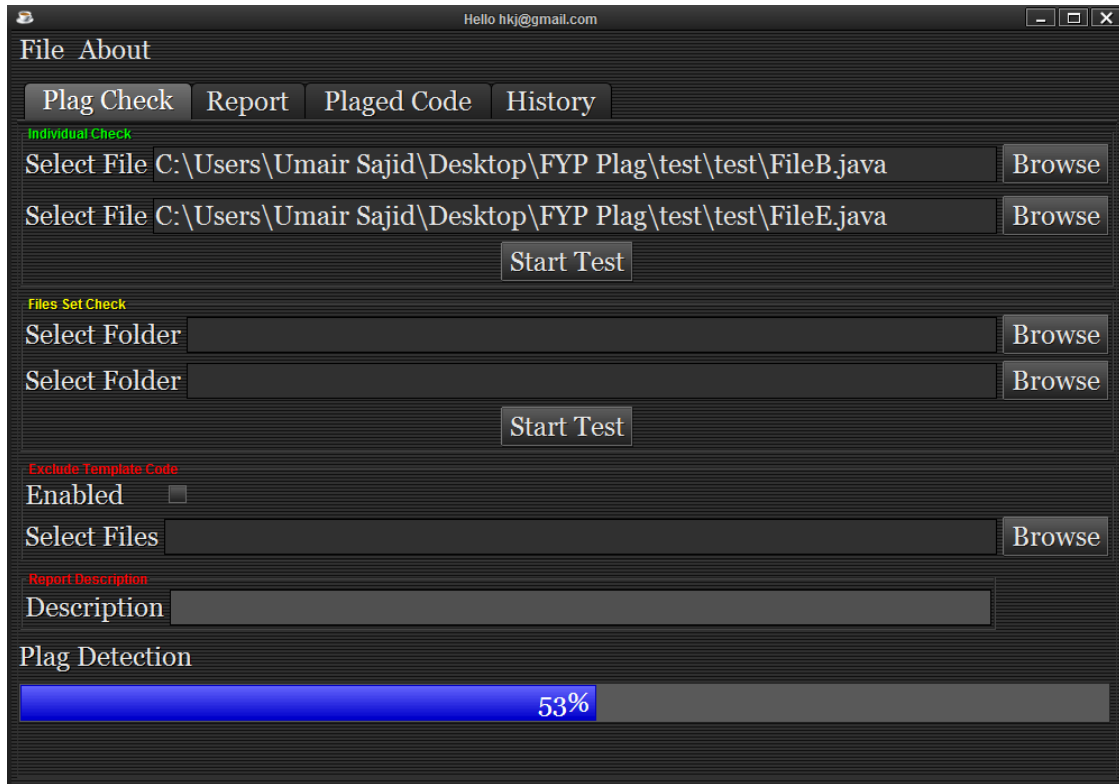


Figure 8-Screen 3: Home Screen

File About

Plag Check Report Plaged Code History

Plagiarism Detection Report

[\[Configuration\]](#) [\[Statistics\]](#) [\[Results\]](#)

Configuration

Minimum match length used	11
Recurse subdirectories	true
Html report directory	dir4/html
Minimum submission similarity to report	0.5
Maximum number of detection results to report	200
Minimum file similarity to report	0.05
Code tokenizer used	plag.parser.java.JavaTokenizer
Filename filter used	plag.parser.java.JavaFilenameFilter
Checked submissions in subdirectory	round1
Excluded files	ExistingCode1.java,ExistingCode2.java
Excluded subdirectories	webstore,webstore/server,webstore/remote,webstore/servlets
Excluded code in template files	
Interface code excluded	true

Statistics

Distribution of similarities, Average: 53%

Figure 9-Screen 4: Report

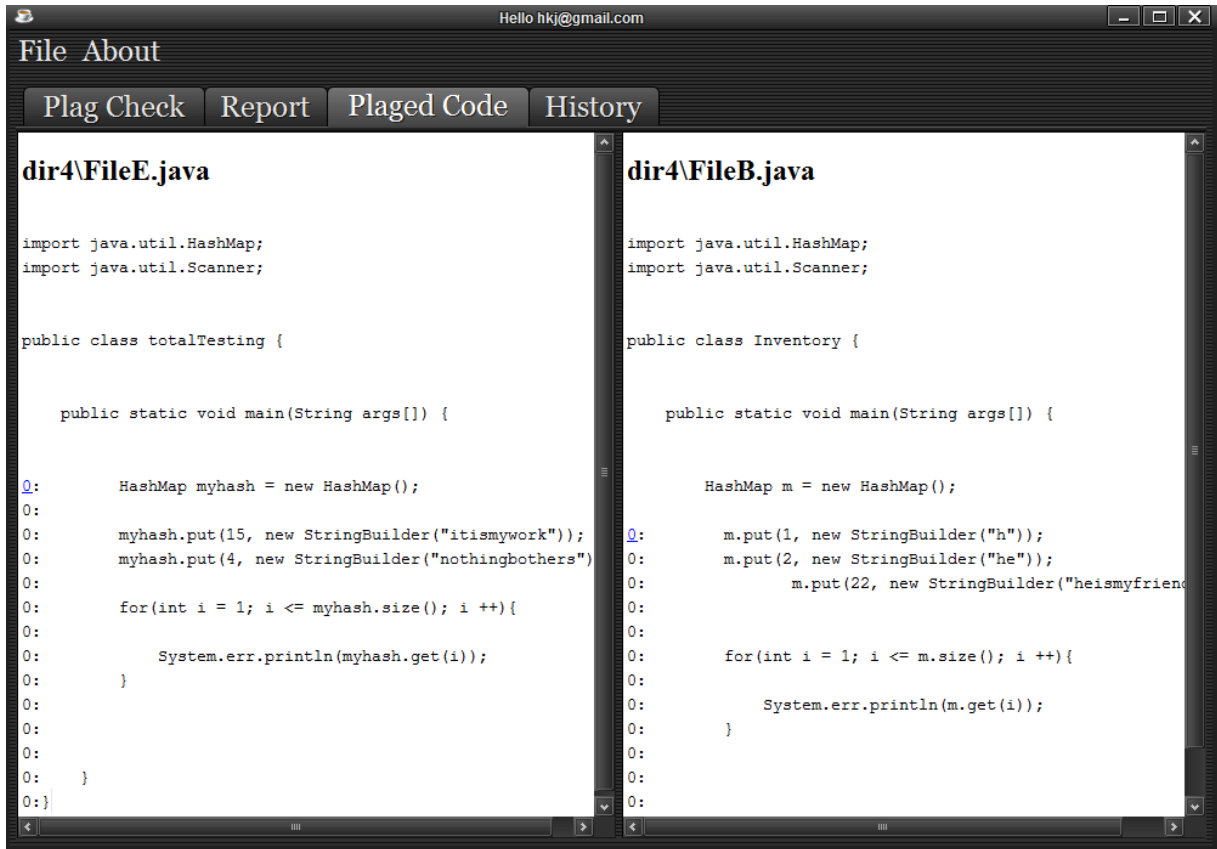


Figure 10-Screen 5: Code Highlight

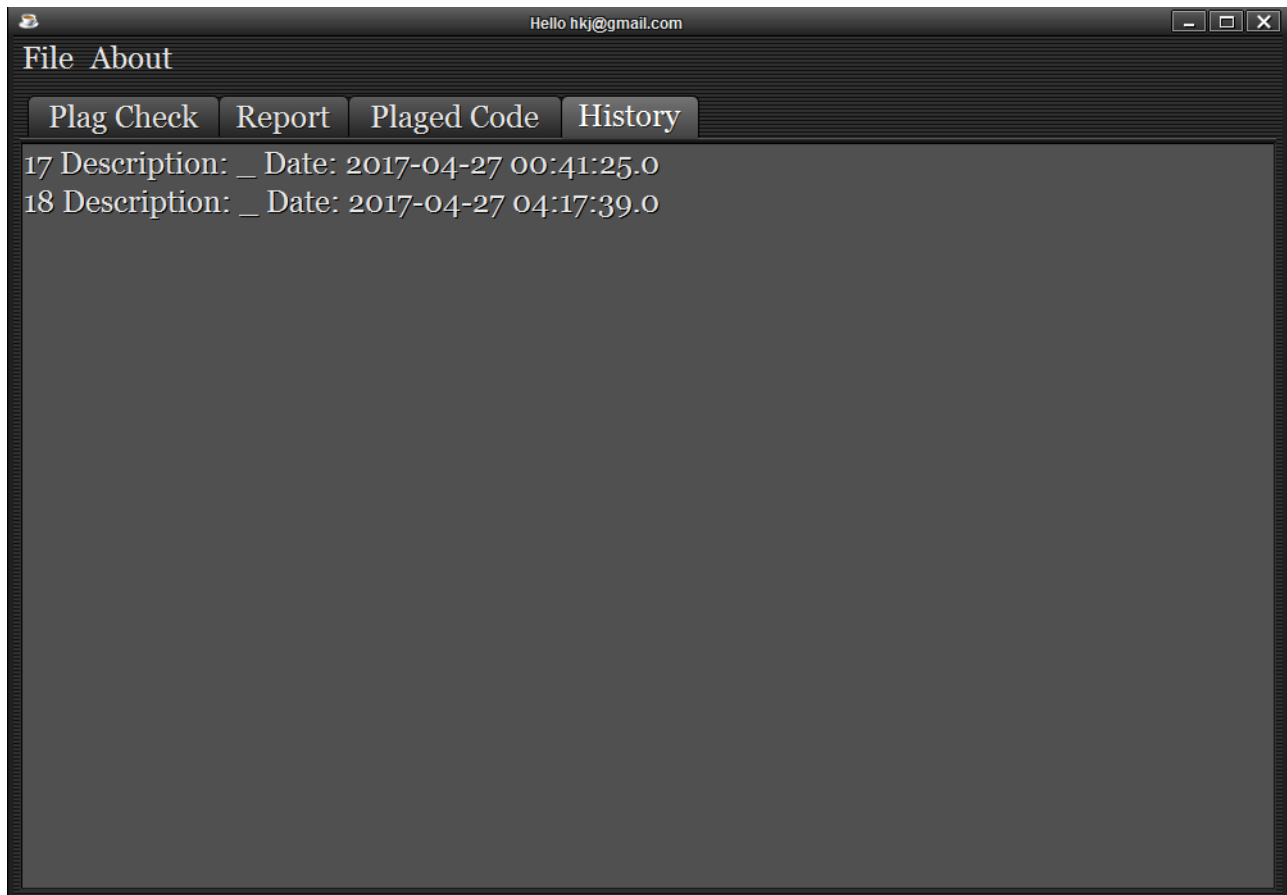


Figure 11 -Screen 5: History Tab

4.10 Detailed Design

4.10.1 Use Cases

This section encompasses the use cases of the system. Depending upon the users of the system, there can be two main use cases. These are described in the following subsections.

4.10.1.1 Use Case Diagram

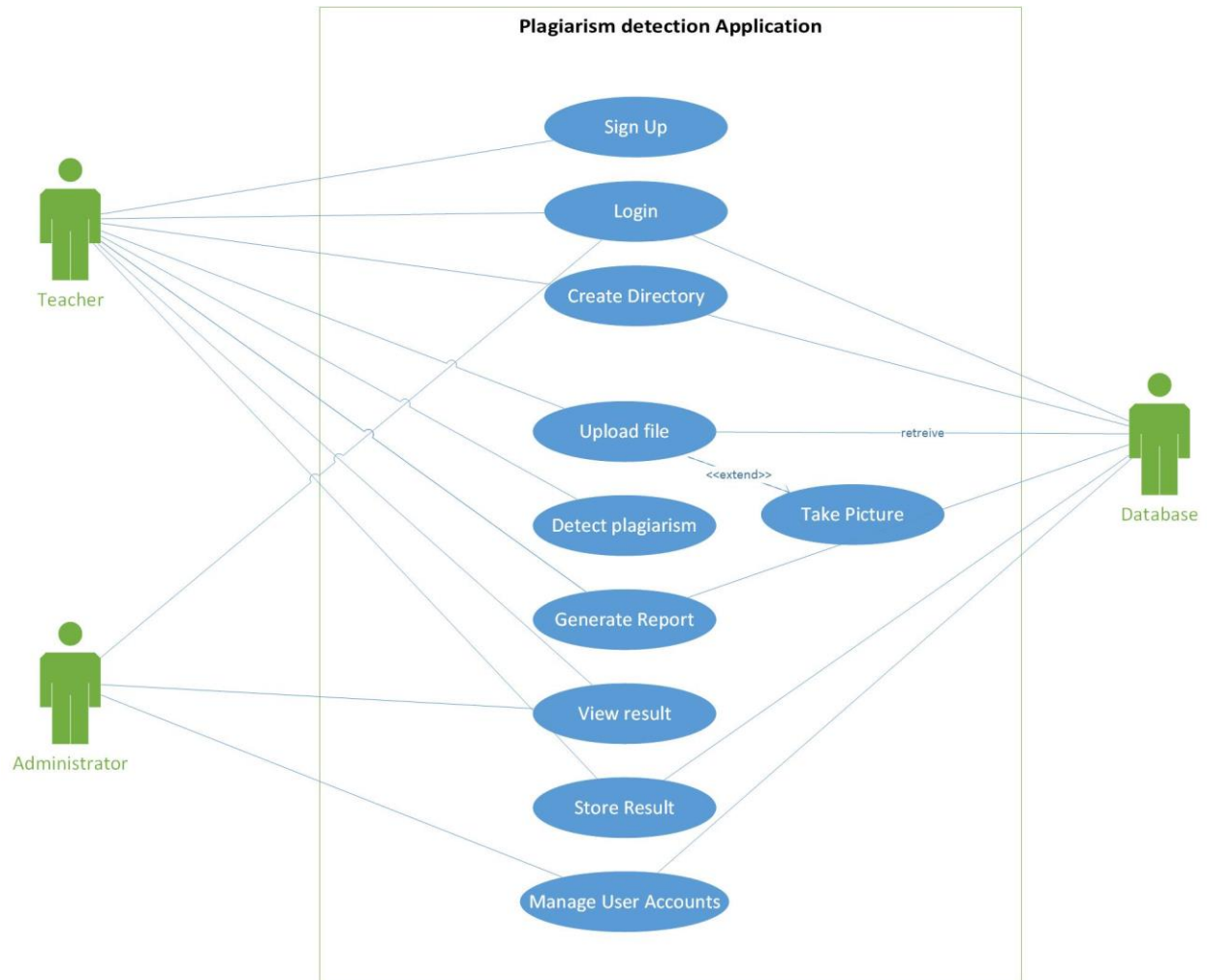


Figure 12-Usecase Diagram

This use cases are aimed for the public users which checks the plagiarism of source codes and results in reports. The following tables has further description about the use cases.

4.10.2 Activity Diagrams

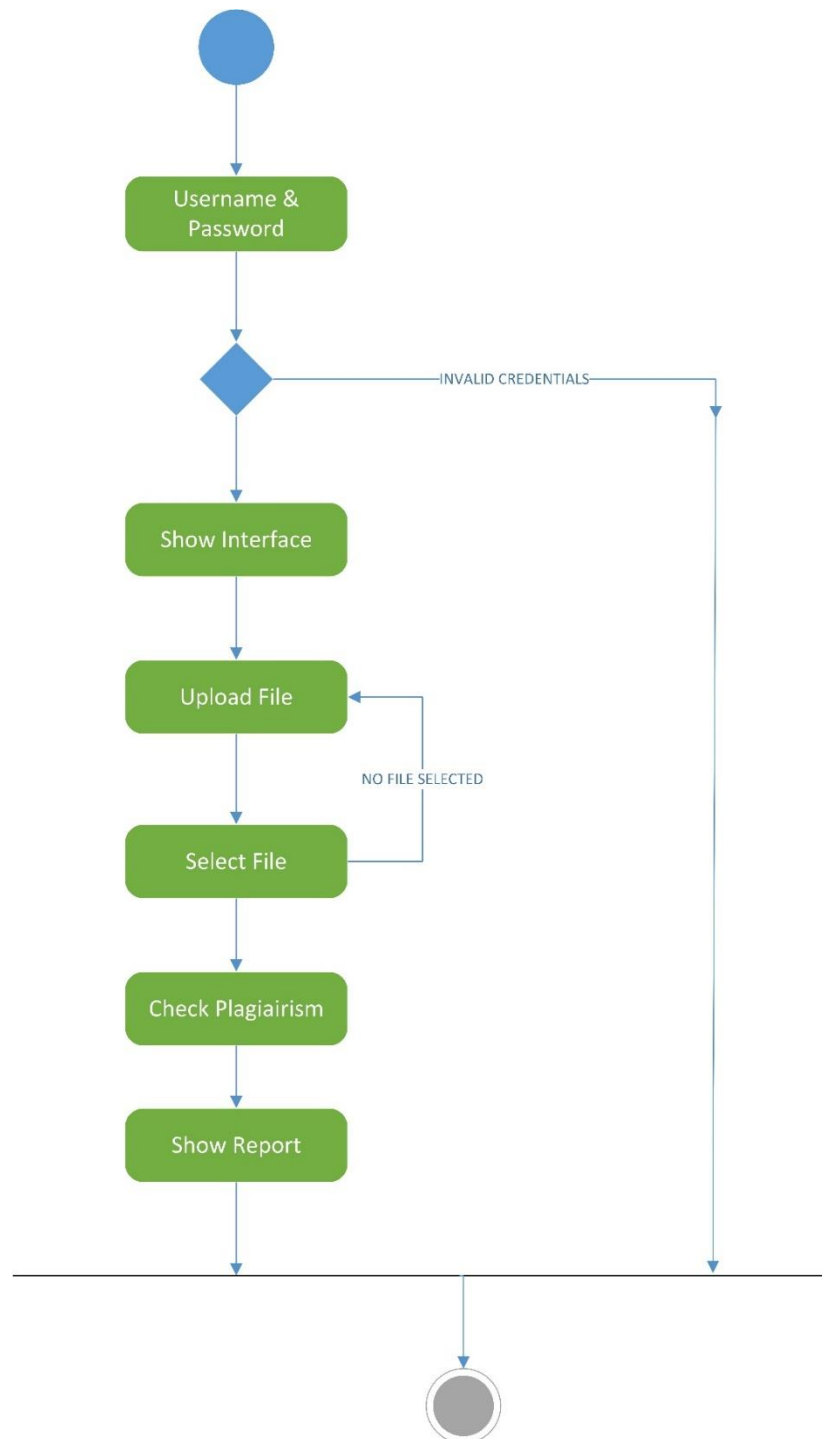


Figure 13-Activity Diagram: Plagiarism Detection Application

4.10.3 Sequence Diagrams

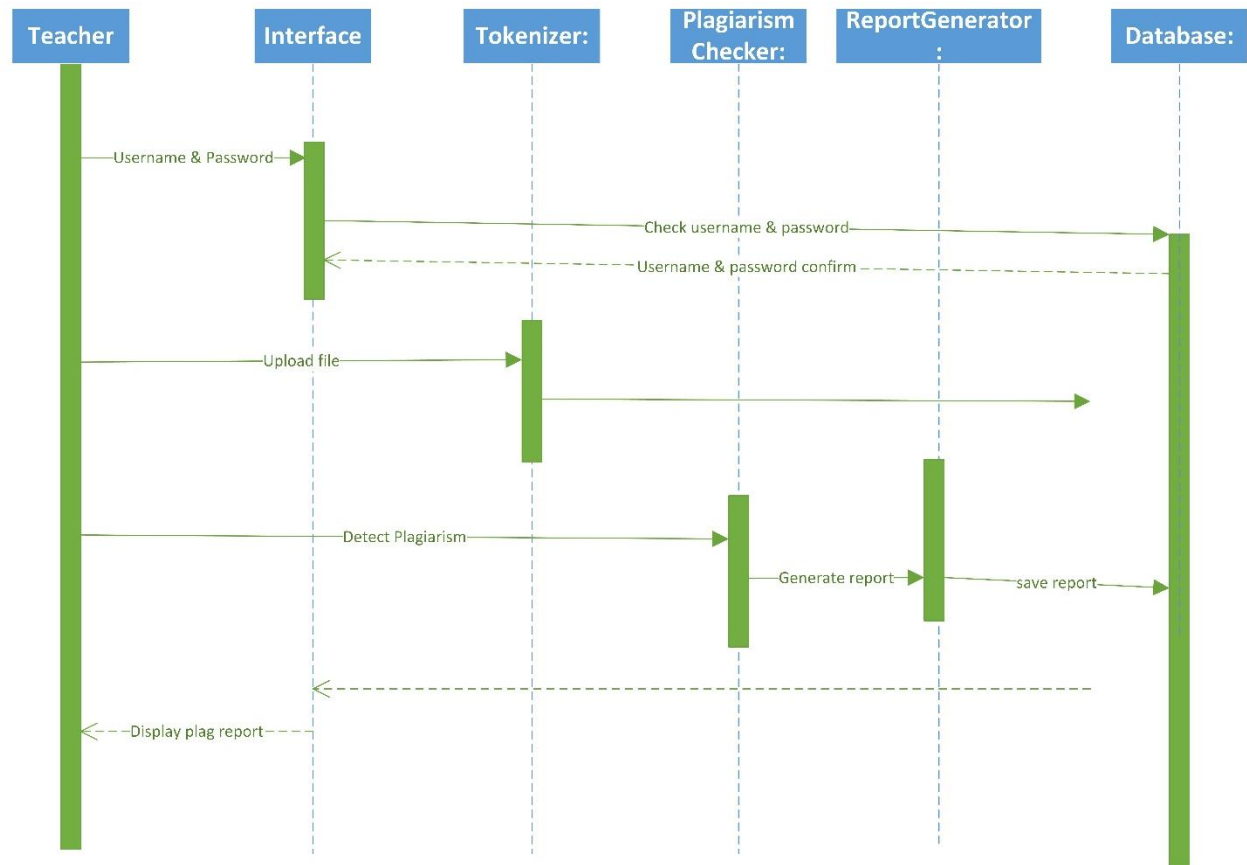


Figure 14-Sequence Diagrams 1: Plagiarism Detection Application

4.10.4 Class Diagram

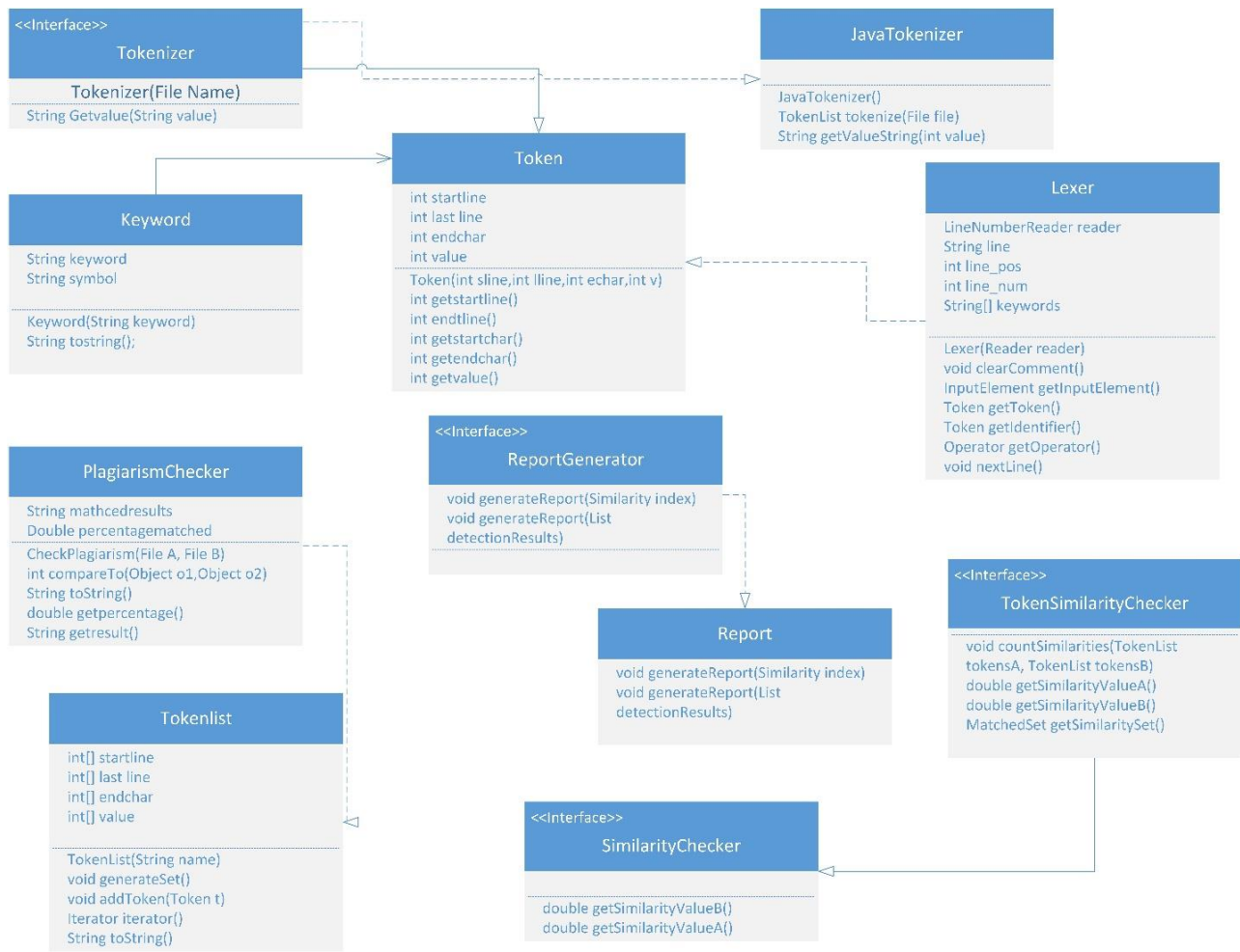


Figure 15-Class Diagram

4.11 Reusability

There has been a focus on the reusability of components developed in the project as well. Therefore, functionality is kept modular. Functionality for each module is defined with as much clarity as possible. Dependencies between different components of the system have also been demonstrated in this document. Inputs and outputs for all modules are clearly stated and each module properly interfaced so that it can be reused within other projects easily.

4.12 Design Decisions and tradeoffs

The design decision to divide the system up into a four-module system was made to promote modularity by keeping the functionality perspective in mind. Modularity provides encapsulation for the important pieces of the system, which allows the developers to make changes with minimized effect on the entire project.

The backend modules, Plagiarism Checker and Report generation are designed separately to support the modularity in the system. Plagiarism Checker module makes use of existing software and is designed in a manner so that it can be reused in future projects as well. Sentiment Analysis module provides the core functionality of the system. Reusability factor is kept in mind for designing the core functionality as a separate module.

The database module is kept separate and acts as a bridge between the backend processing and frontend interface. This is desirable as it divides the overall functionality into two halves. Natural Language Processing are performed at the backend, and results are stored in the database. The web interface simply uses the database for extracting only the desired information. If there is no database between the backend modules and the web interface, we face problems like huge bandwidth requirements and decreased efficiency.

Client Server architecture is used in the development of web interface as there is a centralized database server for communications. Local machines need very little processing capabilities as most of the work is done on the server side. The users only need to have a stable internet connection. This allows the application to work at good speeds on PCs and laptops as well as mobile devices such as tablets and phones.

4.13 Pseudo Code for Modules

4.13.1 Plagiarism Detection Module

```
GETCONSECUTIVEIF (TOKENFILE)
```

```
{  
LOOP INDEX = 1 TO TOKENFILE.LENGTH  
  
{  
IF(TOKENFILE[INDEX] = "if" THEN  
ADD 1 IN CONSECITIVE_IF  
}  
  
RETURN CONSECITIVE_IF
```

4.13.2 Report Generation Module

4.13.3 Database Module

```
Create_database ()  
  
Open DBconnection ()  
  
InsertIntoDB (user conversations )  
  
//Create Trigger for finding popular trends from discussion  
topics, based on the frequency of nouns.
```

4.13.4 Desktop Interface Module

```
//Update UI (topics) from database.  
  
//Update threat from database  
  
//Data Visualization of user sentiments  
  
//Data Visualization of threat details.
```

CHAPTER 5
IMPLEMENTATION

5 System Implementation

5.1 Programming Language:

Java is used as programming language to develop the application and for maintaining the database **MySQL** language is used.

Using the Java Cup Library we have implemented the Parser.

5.2 Development Tools:

Eclipse IDE with GUI Libraries and MySQL Connector JDBC is used for the development of the application and MySQL Database is used for database development.

5.3 Database:

Databases were developed and managed in MySQL.

5.4 Operating System:

On server side, Apache is used while the server backend has been tested with windows as an operating environment, while reports are tested on all browsers of Microsoft Windows.

5.5 Complete System Implementation:

5.5.1 Java Server Module

Java Server is implemented using Socket Programming and with HTTP protocol we are transferring the files and plagiarism detection results generated from the files. Along with that it's also responsible for the Interfacing which will be done in Android. Using the same server on a single port we will implement the Client Side on an Android as we have done for the Desktop Application.

5.5.2 Client Module

This module is responsible for registering itself with the Java Server. Each instance of this module is responsible for one user. It receives a Request from the Java Server and is responsible for handling the commands which are being issued such as sending files, detecting plagiarism and generating reports and transferring the result file back to the Java Server.

CHAPTER 6

TESTING AND EVALUATION

6 System Testing

This test plan document describes the appropriate process and methods used to plan testing of the Plagiarism Detection System. The test plan will ensure that Plagiarism Detection System works as intended without any failure.

We are performing manual testing of the project i.e.; no tool or script is being utilized for testing purpose. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. First of all, each unit (module) will be tested separately and then whole project will be tested after integration of all the units.

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed.

6.1 Test Items

Based on the requirements of Plagiarism Detection System, design description, modules of web application, SQL database and non-functional scenario will be tested. The requirements defined in Software Requirements Specification and the design entities as explained in Software Design Document will be tested.

6.2 Features to be tested

Following features are to be tested:

- Ability to Authenticate and Sign in of each Users
- Ability to Sign Up a new User
- Ability to Upload Files
- Ability to Check Plagiarism
- Ability to Generate Reports
- Ability to View History

6.3 Approach

The project is using waterfall approach, with producing modules and integrating them. All the modules are tested individually and then integrated with system and integration tests are applied.

Black Box testing technique will be used for testing functionality of each module.

6.3.1 Unit Testing

Unit testing is that part of testing which requires a thorough check of each module of the project. In our project, there are 6 modules which we have to check if they are functioning normally or not. For this we will start from the module which is least dependent on other modules for its functions and then work our way through to the module which is dependent on other modules to function and test.

6.3.2 Integration Testing

Integration testing is the part where we will test all the previous tested modules in a way that they are functioning normally when they are combined together.

6.3.3 System Testing

In the end, system testing will ensure that all the modules are working, separately and together combined. Then only the final outcome of the program will decide the correctness of whole system.

6.3.4 Item Pass/Fail Criteria

Details of the test cases are specified in the section Test Deliverables. Following the principles outlined below, a test item would be judged as pass or fail.

- Preconditions are met
- Inputs are carried out as specified
- The actual output is as specified in the expected output for the test to be passed
- The system does not work or the actual output is different from the expected output for the test to be failed

6.4 Suspension Criteria and Resumption Requirements

Testing will be suspended on the occurrence of any defect/bug in system which cannot allow further testing of the system. Testing will be resumed when the defect/bug is removed.

6.5 Test Deliverables

Following are the test cases performed on the system for testing purpose:

6.5.1 Test Case 1

Test Case Name	Authenticating Users (Signing in of different Users)
Test case no	1
Description	This module will authenticate the users. Different Users such as Different Instructors and Administrator can log into the System using login Email/Username and password.
Testing Technique Used	Black Box testing
Preconditions	Database should be running
Input Values	Fill the login details such as Email and Password
Steps	Fill the Email and Password Click "Login" button
Expected Output	System will take the User to Main Screen of the System
Actual Output	System authenticates each sign in using the Database Entity "Users"
Status	Pass.

Table 12 Test Case 1

6.5.2 Test Case 2

Test Case Name	Sign Up
Test case no	2
Description	This module will let any new User to sign up for the system.
Testing Technique Used	White box testing
Preconditions	All files must be filled and unique username should be selected to sign up for system
Input Values	Email Address, User Name, Password, Confirm Password
Steps	All the required fields are filled
Expected Output	The User Credentials will be saved in Database.
Actual Output	User will be generated in database which can login to System.
Status	Pass.

Table 13 Test Case 2

6.5.3 Test Case 3

Test Case Name	Upload Files
Test case no	3
Description	This module will let the Users to upload files to be checked for Plagiarism.
Testing Technique Used	Black Box testing
Preconditions	Files must be Readable and present in Directory.
Input Values	Filled Values in the Text Box of Files.
Steps	Click Upload Button
Expected Output	Getting the name of Files in Text Box.
Actual Output	Show the name of files in Text Box.
Status	Pass.

Table 14 Test Case 3

6.5.4 Test Case 4

Test Case Name	Check Plagiarism
Test case no	4
Description	This module will allow the User to detect Plagiarism Against Uploaded Files in the Directories.
Testing Technique Used	Black Box testing
Preconditions	Files Should be uploaded in the Directories.
Input Values	Click on Check Plagiarism Button.
Steps	Upload Files, Click on Check Plagiarism Button.
Expected Output	System will Check the Plagiarism For the given Files.
Actual Output	Plagiarism will be Checked for the Files and results will be passed to Report Generator.
Status	Pass

Table 15 Test Case 4

6.5.5 Test Case 5

Test Case Name	Generate Reports
Test case no	5
Description	This feature allows the user to generate report.
Testing Technique Used	Black Box testing
Preconditions	Files are Uploaded, Plagiarism is detected.
Input Values	Click on Generate report Button.
Steps	Upload files in directories, Check for plagiarism, Generate Report.
Expected Output	Report is Generated in the form of html and is shown to the user.
Actual Output	Report is Generated in the form of html and is shown to the user.
Status	Pass.

Table 16 Test Case 5

6.5.6 Test Case 6

Test Case Name	View History
Test case no	6
Description	This feature allows the user to view the history of their submissions and reports.
Testing Technique Used	Black Box testing
Preconditions	Data should be present in the database
Input Values	Click on view History Tab.
Steps	Click on History Tab.
Expected Output	Previous Generated reports and submissions will be shown to the user.
Actual Output	Previous Generated reports and submissions will be shown to the user.
Status	Pass.

Table 17 Test Case 6

CHAPTER 7

CONCLUSION AND FUTURE WORK

7 Future Work Conclusion

Basically **Plagiarism Detection Application** is going to be a revolutionary step towards the customer satisfaction and eliminating plagiarism from Academics. When it comes to the Information technology and Software Practices this is highly Unethical that you copy someone's code. However, this thing is already being used in the foreign countries. But In Pakistan, HEC (Higher Education Commission) have no policy in this regard.

Plagiarism Detection Application can be extended by having a complete set of new algorithms which can be used to check source code plagiarism even more efficiently.

Its use case and market can be extended not just to freelancing agencies and outsourcing agencies but to all the article publishers around the world who wants to sell their customer an accurate source code to the community.

Right now, if we just extend from the Teachers to the University level. At every quiz, at every assignment this should be used. Teachers should have complete information of each student's history to know more about the code itself.

BIBLIOGRAPHY

- [1] A. Sheild, "UML 2 Use Case Diagrams: An Agile Introduction", *Agilemodeling.com*, 2017. [Online]. Available: <http://www.agilemodeling.com/artifacts/useCaseDiagram.htm>. [Accessed: 04- May- 2017].
- [2] S. Amber, "UML 2 Activity Diagrams: An Agile Introduction", *Agilemodeling.com*, 2017. [Online]. Available: <http://agilemodeling.com/artifacts/activityDiagram.htm>. [Accessed: 04- May- 2017].
- [3] S. System, "Sparx Systems - UML 2 Tutorial - Activity Diagram", *Sparxsystems.com*, 2017. [Online]. Available: http://www.sparxsystems.com/resources/uml2_tutorial/uml2_activitydiagram.html. [Accessed: 04- May- 2017].
- [4] D. Bell, "UML basics: An introduction to the Unified Modeling Language. IBM Global Services", *UML basics*, 2017. [Online]. Available: http://www.nyu.edu/classes/jcf/g22.2440-001_sp06/handouts/UMLBasics.pdf. [Accessed: 04- May- 2017].
- [5] M. MSDN, "UML Class Diagrams: Reference", *Msdn.microsoft.com*, 2017. [Online]. Available: <https://msdn.microsoft.com/en-us/library/dd409437.aspx/>. [Accessed: 04- May- 2017].

APPENDIX A-1
USER MANUAL

Reading Instructions

This user manual will describe the way in which **PLAGIARISM DETECTION APPLICATION** can be used. It contains the instructions needed to utilize this software. This system provides a user-friendly interface which allows you to efficiently interact with the system.

This User Manual should be read in the order given.

Installation

The front end of this software is a desktop application which is built in JAVA. Therefore, no particular Operating System is required. Internet Explorer 9 or above, Mozilla Firefox or Google Chrome and XAMPP MySQL are required for the Desktop Application to work properly.

How to Use the Plagiarism Detection System

Hardware

Plagiarism Detection Application does not require its users to use specific hardware. System with an Internet connection is enough to carry out the job.

Software Use

1. Login

User will need to register using his email and get a username and password. He can only login using these credentials. If in case of wrong credentials, he will need to enter them again.

2. Register

User can register himself using a proper email and a strong password. Which is being validated for the field. If he inputs wrong email. Error Message will be shown and he will need to enter it again.

3. Check Plagiarism

User will need to upload the files in the files parameter. In case of directory he can copy paste the path or Upload the directory itself. Later onwards he will need to Check the Plagiarism by clicking on the Check Plagiarism Button. It must be assumed by the system that the user has input his name in the code at the first line.

4. Report Generation

Report is generated right after when the plagiarism is checked by the system. The user can view the report either in the browser or in the Report Tab of the system.

5. Code Highlighted

Code highlight is being shown in the Code Highlight Tab, User should assume that the Code having 0: is basically plagiarized code and this one should be removed or changed in order to correct. Other code without 0: in the prefix of the statements are not plagiarized and can be considered correct by the instructor.

6. Check History

History Tab have the following History of the user submitted codes or directories. He can click on the History of each and can view the reports generated of the previous submissions.