MINI - PROJECT REPORT ON

"Mini Project Plagiarism Detector"

BY

Burhanuddin Unwalla

&

Rohan Bhange

Under the Guidance of :

Prof. Suresh Kapare

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MIT SCHOOL of Engineering

Loni Kalbhor Pune

M.I.T. SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

LONI – KALBHOR PUNE

CERTIFICATE

MIT-ADT UNIVERSITY
PUNE, INDIA
A leap towards World Class Education

This is to certify that the Mini- Project report entitled

"*MINI PROJECT PLAGIARISM DETECTOR*"

submitted by

Burhanuddin Unwalla MITU19BTML0106

Rohan Bhange MITU19BTML0023

is a record of Bonafede work carried out by them, under my guidance, in partial fulfilment of the requirement for the Second Year of Engineering (Computer) at M.I.T. School of Engineering, Pune under MIT Art, Design & Technology University.

Date: Place:

Prof. Suresh kapare

Guide,

Department of CSE
M.I.T. School of Engineering
Loni Kalbhor, Pune

Dr. RajneeshKaur Sachdeo

Dean Engineering,

Head, Department of CSE
MIT School of Engineering
Loni-Kalbhor, Pune

ACKNOWLEDGEMENT

We would like to express our humble gratitude towards our mentor Prof. Suresh kapare as well as our principal Dr. Kishore Ravande who gave this this golden opportunity to work on this interesting project This project helped us gain in depth knowledge about natural language processing which is the future of this fast-paced world. It also helped analyse the Pros and Cons of a particular application and its market viability. We provide valuable information through our analysis which will help upcoming app developers plan meticulously as to how their application has to be designed. Both members contributed greatly to the timely completion of this project which would have been difficult without the assistance of our guide.

TABLE OF CONTENTS

SR NO	CONTENTS	PAGE NO
1	ABSTRACT	5
2	LIST OF TABLES	
3	LIST OF FIGURES	
4	1. INTRODUCTION	6
	2. PROBLEM DEFINITION	7
	3. FEATURES OF PROJECT	7
	4. PLATFORM / TECHNOLOGY	8
	5. FLOWCHART	9
	6. OUTPUT	10
	7. CODE	24
	8. GLOSSARY	
	9. REFERENCES	48
	10. CONCLUSION AND FUTURE ENHANCEMENTS	49
5	GLOSSARY	
6	REFERENCE AND BIBLIOGRAPHY	48
7	ANNEXURE A	50

ABSTRACT

The problem of plagiarism is one that has long plagued educational and various other institutions. This has inevitably led to a decrease in the quality of education and effort of pupils. Our project endeavours specifically to deter and stop plagiarism by students in the domain of mini projects. We have implemented a method called SVDplag that has shown to be better than other methods. Results show that this method along with other pre processing steps can effectively detect plagiarism.

INTRODUCTION

The rise of digitized education and widespread and easy access to the internet and computers had led to an explosion of plagiarism which at times is as simple as copying and pasting. As a result there has been a deterioration in the quality of education an increase in complacency in students. In order to stop this worrisome trend effective and efficient methods to detect plagiarism are the need of the hour.

our project attempts to detect plagiarism attempts in mini project. A database of all project reports is pre-processed, converted into mathematical models and finally a similarity metric is applied to detect the similarity.

PROBLEM STATEMENT

Processing a large database of documents to discover the presence of any plagiarism present in an efficient and effective manner.

FEATURES OF THE PROJECT

The project has the following basic features:

1. The user can choose any folder on which to run plagiarism analysis.
2. The user can also choose a file to add to the folder already chosen.
3. Display plagiarism results only for the selected file that is added later.
4. Display plagiarism results for all files in the folder with each other
5. Display a bar chart of plagiarism scores of the selected file with all other files.
6. User login system for security.

PLATFORM / TECHNOLOGY USED IN THE PROJECT

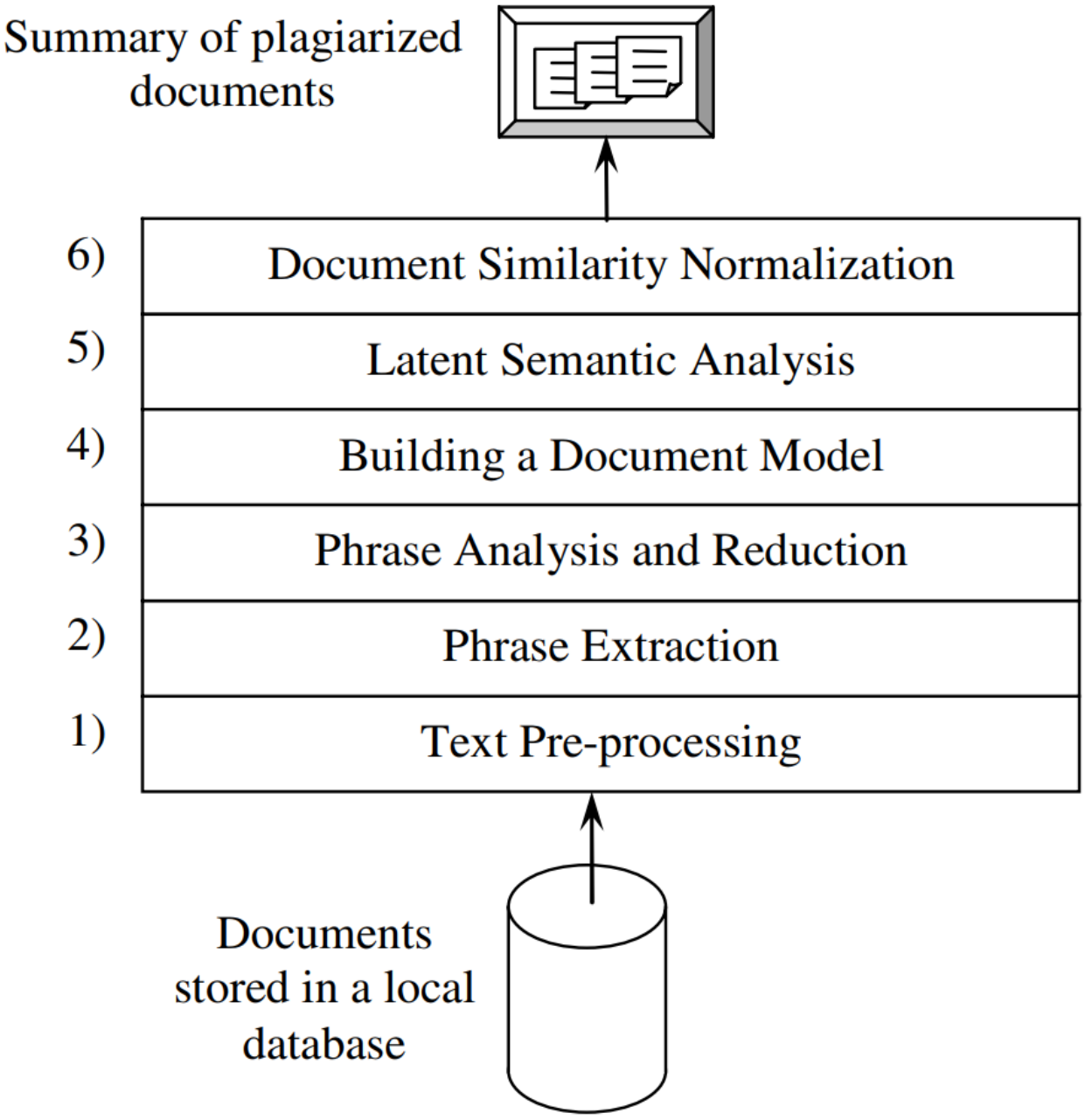
1. JAVA

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture.

Libraries used:

1.commons math3 3.6.1
2.mysql connector java 8.0.25
3.opennlp 1.9.3
4.pdfbox 2.0.24

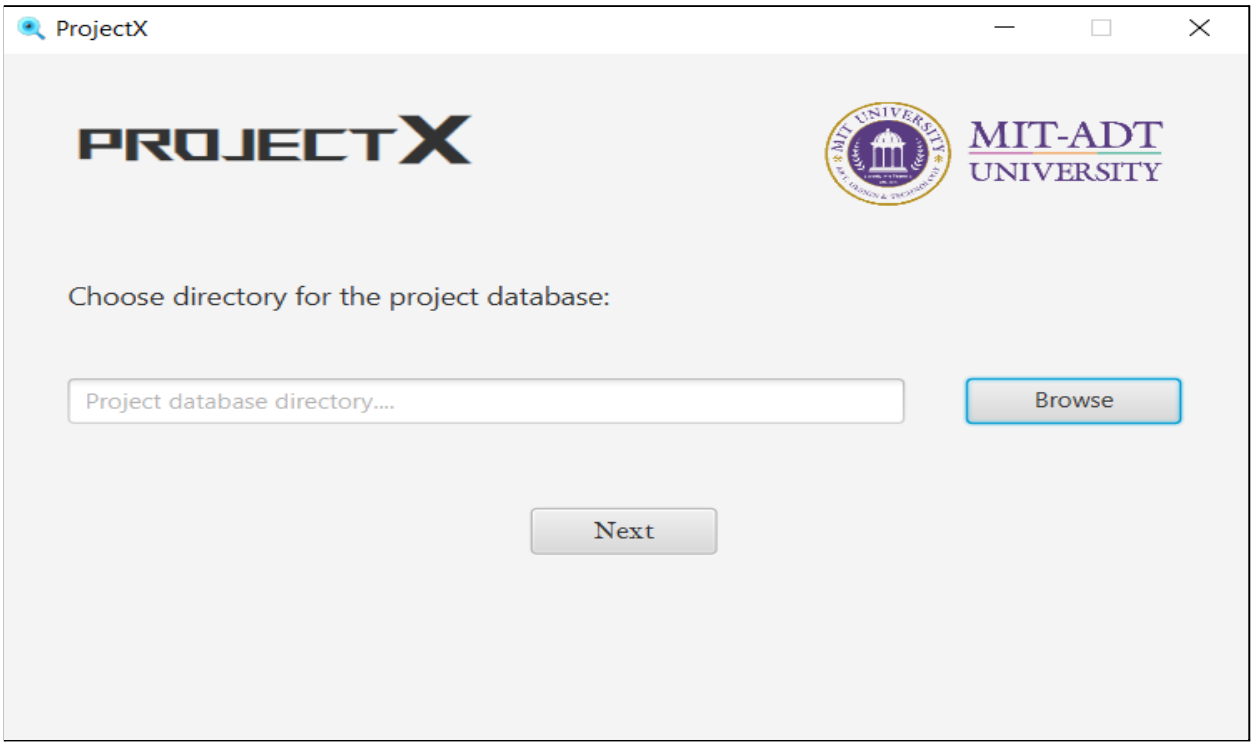
FLOWCHART:


Summary of plagiarized documents
6)
Document Similarity Normalization
5)
Latent Semantic Analysis
4)
Building a Document Model
3)
Phrase Analysis and Reduction
2)
Phrase Extraction
1)
Text Pre-processing
Documents stored in a local database

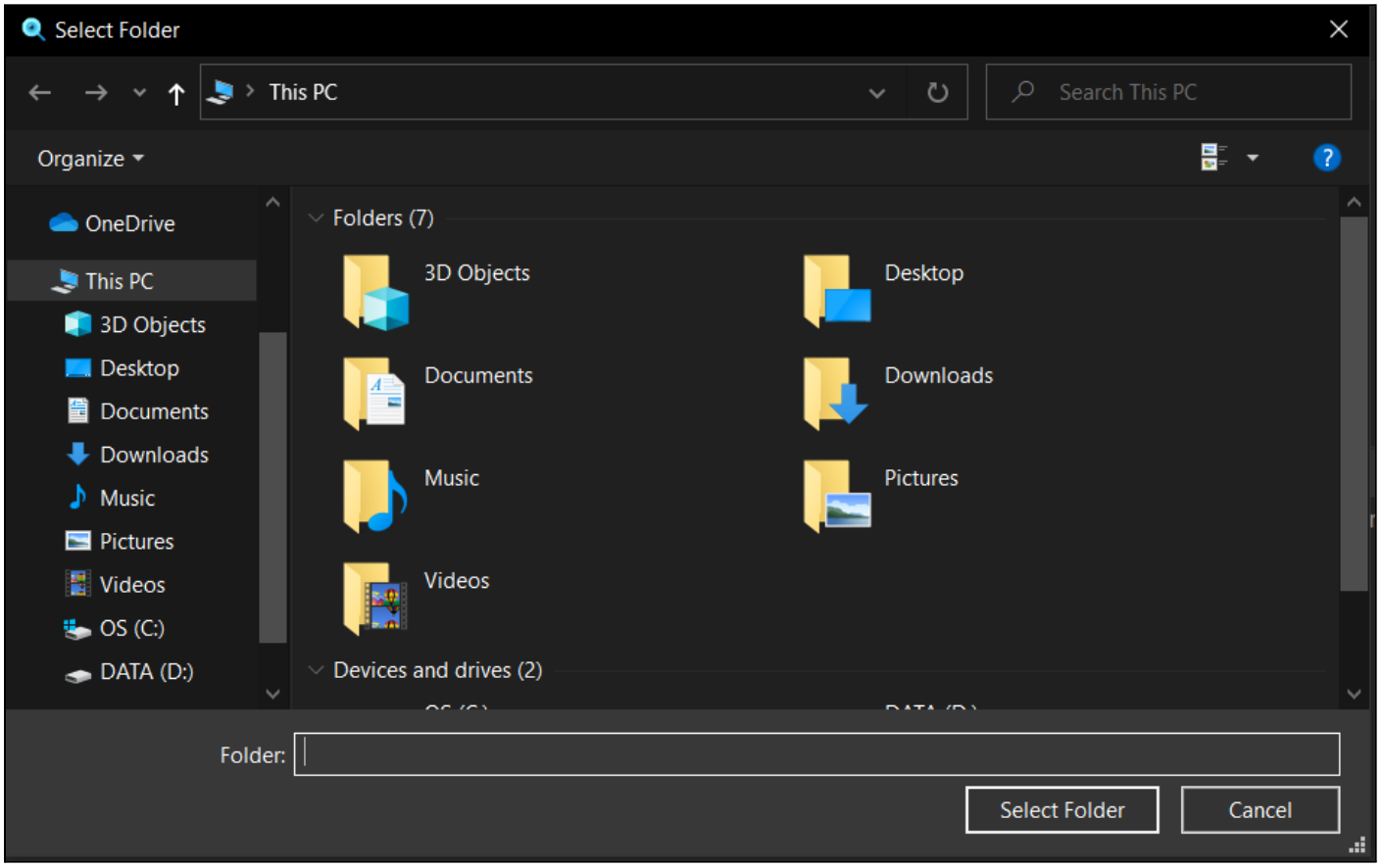
OUTPUT

The Login Screen.

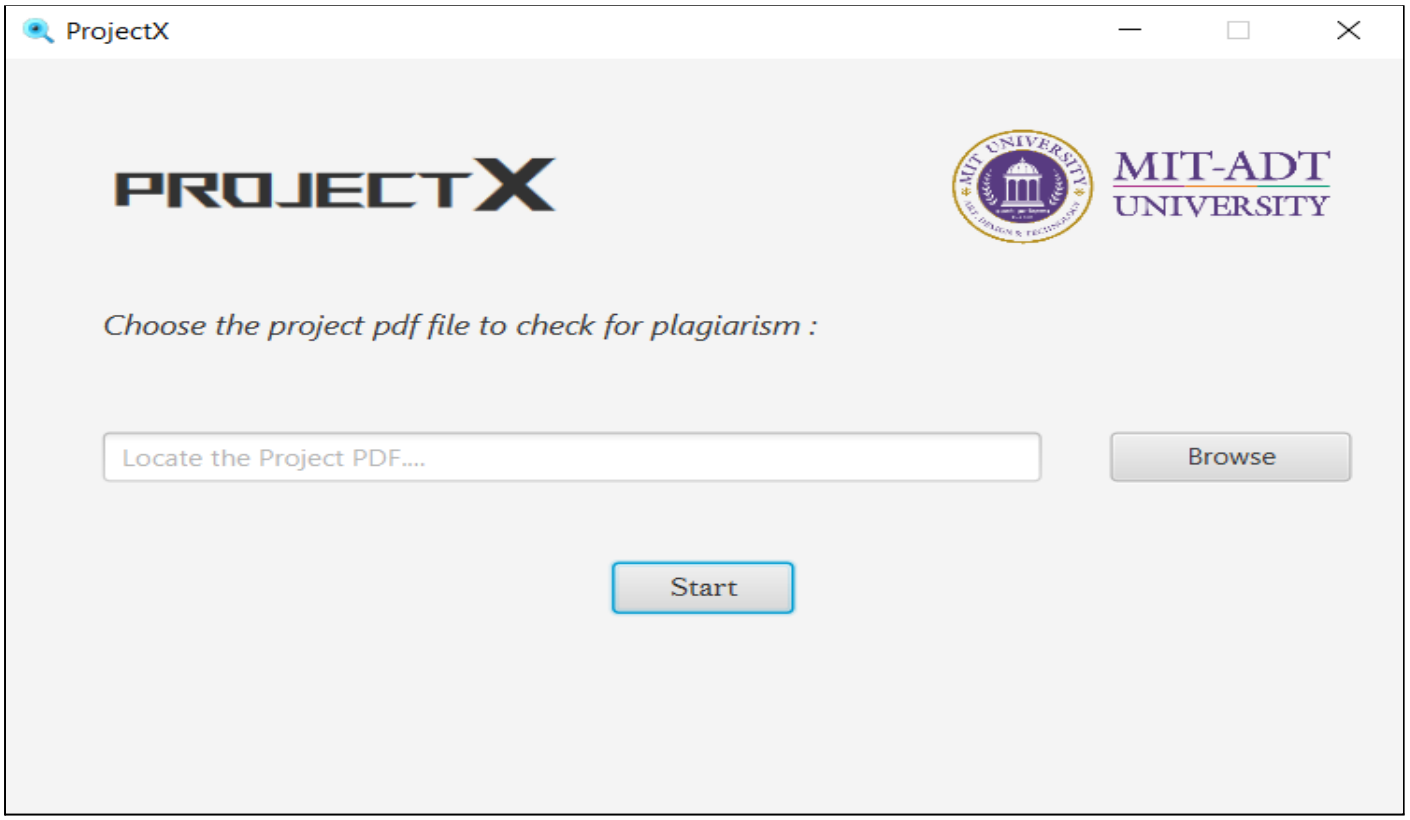
Choose Directory Window:



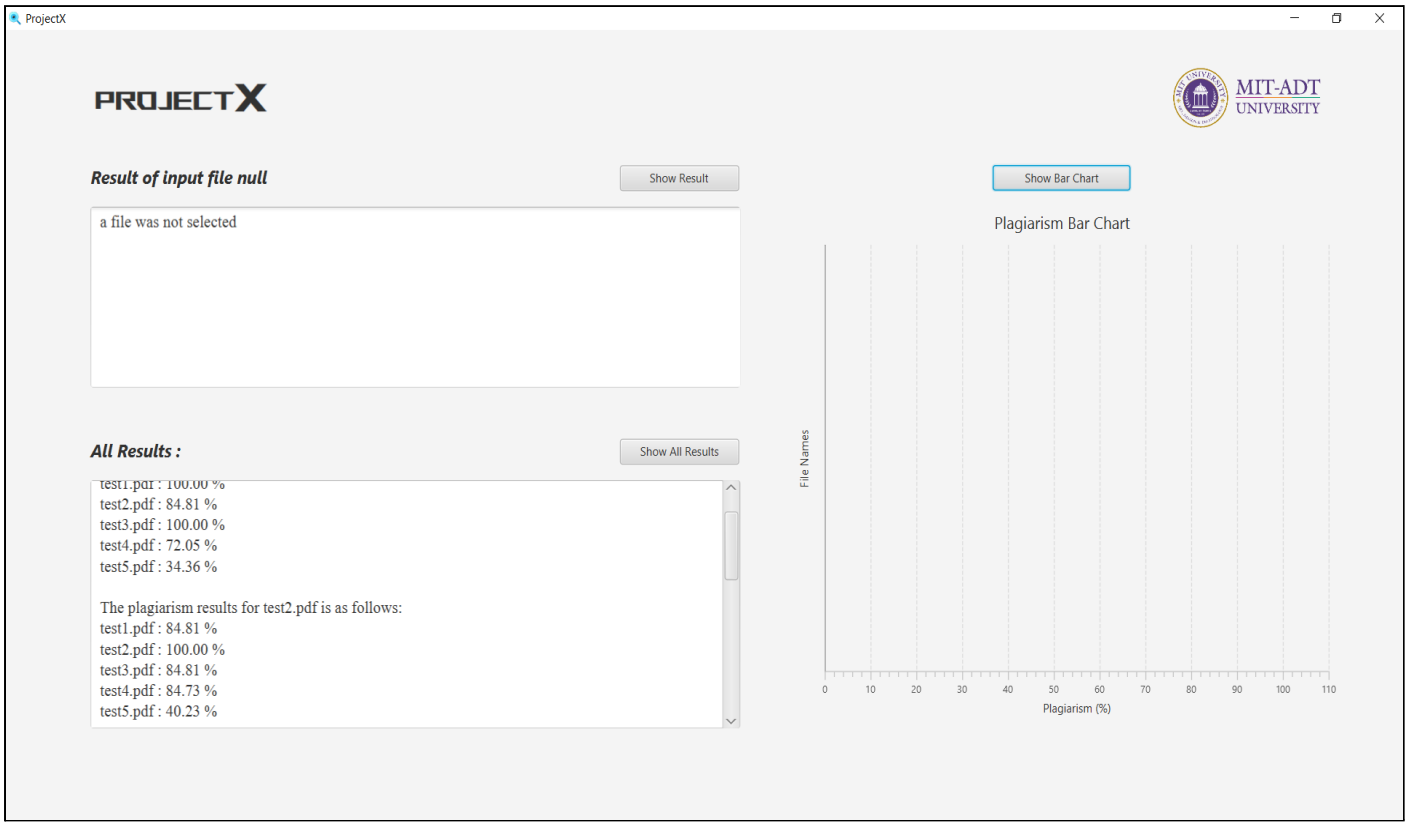
Select folder Popup:(Opens after clicking Browse button in previous window)



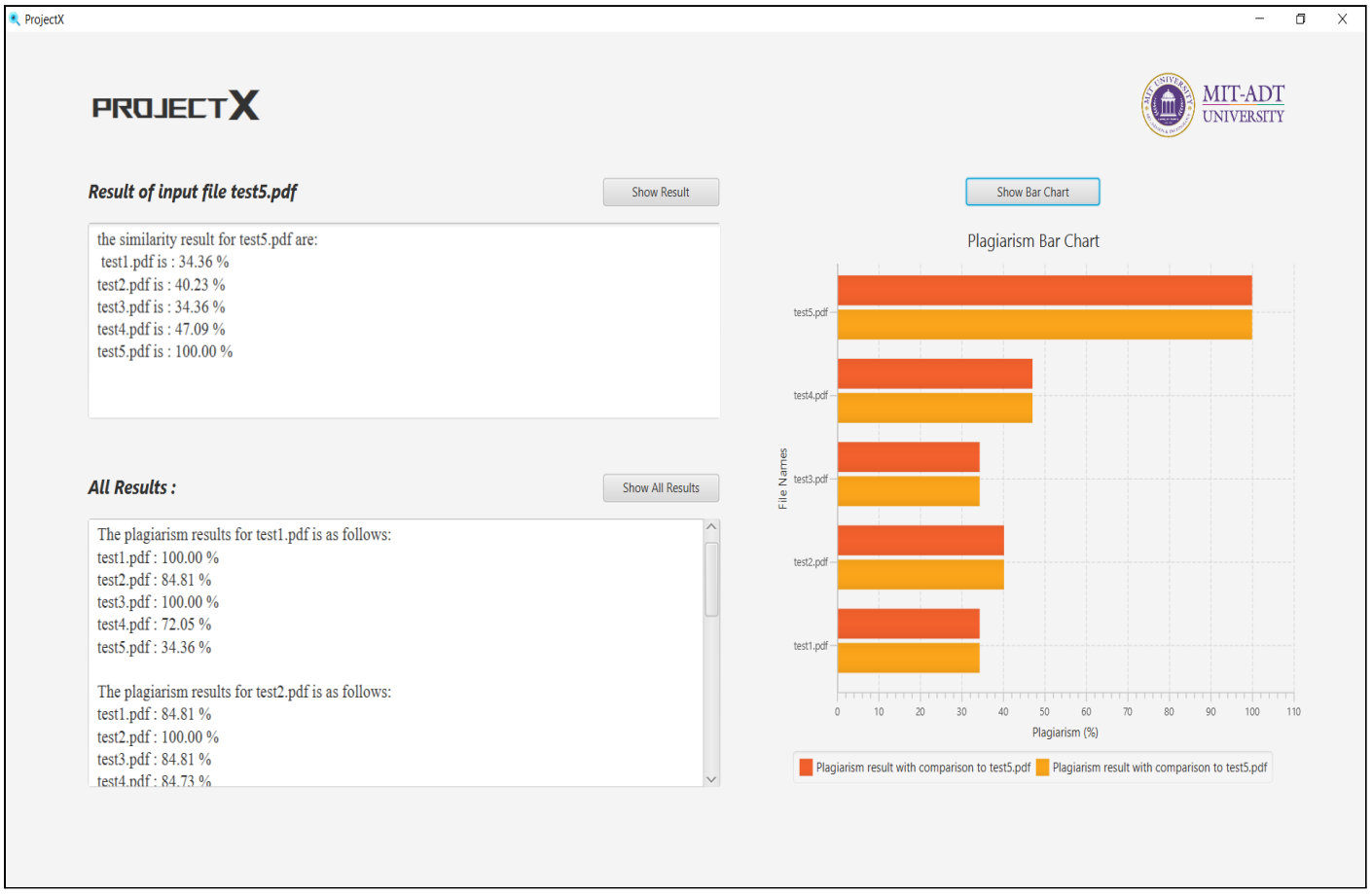
Choose pdf file window:(Opens after selection database directory and clicking next button in previous window)



Result Window 1:(Opens if no pdf file is chosen for plagiarism detection)



Result Window 2:(Opens if a pdf file is chosen for plagiarism detection)



BACKEND CODE PROCESSES:

Step by step description.

Text pre-processing.

text pre-processing involves the following steps:

Punctuation removal:

In this step all punctuation marks are removed from the text. The remove_punc function handles this process.

```
static String remove_punc(String content) {
    content = content.replaceAll("\\p{Punct}", "");
    return content;
}
```

here, \\p{Punct} is the regex for punctuation marks which are replaced by the empty string.

Stopword Removal:

stop words are words which are filtered out before or after processing of natural language data (text). These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc) and *does not add much information to the text*. Examples of a few stop words in English are “the”, “a”, “an”, “so”, “what”. By removing these words, we remove the low-level information from our text in order to give more focus to the important information.

```
static String[] remove_stopwords(String content) throws IOException {
    ArrayList<String> stopwords;
    stopwords = (ArrayList<String>) Files.readAllLines(Paths.get("src/sample/stopwords.txt"));
    ArrayList<String> allWords1 = Stream.of(content.toLowerCase().split(" "))
            .collect(Collectors.toCollection(ArrayList<String>::new));
    allWords1.removeAll(stopwords);
    return allWords1.toArray(new String[0]);
}
```

Here the String the holding the content read from a file is split along the whitespace character and then converted to an ArrayList object. The stopwords are read from a file in the directory and is stored in another arrayList object. The

removeAll method is finally called and all stopwords are removed, the final arrayList is returned as an Array of Strings.

Lemmatization:

Lemmatization is an algorithmic process of determining and reducing the word to its lemma based on its meaning and context the word is used in. All inflected forms of a word are reduced to their lemma so that they can be analysed as one word. In order to perform lemmatization the context and meaning of the words need to be understood which is done via Part Of Speech tagging. The tags for words are nouns, verbs, adjectives etc. Words along with their tags are then used for lemmatization.


```
static String[] POS_tagging(String[] tokens) throws IOException {
   InputStream inputStreamPOSTagger = new FileInputStream("src/sample/en-pos-maxent.bin");
   POSModel posModel = new POSModel(inputStreamPOSTagger);
   POSTaggerME posTagger = new POSTaggerME(posModel);
   return posTagger.tag(tokens);
}
```

this function accepts the Array of words contained in the file and determines their POS tag using a file that contains English words with their corresponding tags. The tags for the corresponding words are then returned by the function.

```
static String[] lemmatization(String[] tags, String[] tokens) throws IOException {
   String[] lemmas;
   InputStream DictLemmatizer = new FileInputStream("src/sample/lemmatizer-dict.txt");
   DictionaryLemmatizer lemmatizer = new DictionaryLemmatizer(DictLemmatizer);
   lemmas = lemmatizer.lemmatize(tokens, tags);
   for (int i = 0; i < lemmas.length; i++) {
      if (lemmas[i].equals("O")) {
         lemmas[i] = tokens[i];
      }
   }
   return lemmas;
}
```

This function accepts the words and corresponding tags and the lemmatize method of DictionaryLemmatizer object to generate the required lemmas using the lemmatizer-dict text file that contains a list of words, POStags and the corresponding lemmas. The method however returns “O” for words (proper

nouns) whose lemmas don't exist. Therefore the for loop replaces the "O" with the corresponding original proper nouns.

Model Building

Building a mathematical model from the data that can mathematically represent the data is the in these steps:

Ngrams Generation:

Ngrams of a list of words consist of all possible combinations of n consecutive words from that list. For e.g. 2-grams of the sentence *"This project detects plagiarism"* are "*this project", "project detects", "detects plagiarism"*. Experiments show the most suitable n-grams length lies between 2 and 7 **[1]**. Smaller values of n are unable to detect over lapping text, while larger values of n cannot detect fine grained modifications.

```
static String[] n_gram(String[] tokens) {
   return NGramGenerator.generate(Arrays.asList(tokens), 2, " ").toArray(new String[0]);
}
```

this function takes the array of words that have been lemmatized and generates bi-grams and returns the bi-grams as an array.

Vocabulary Building:

Vocabulary is a tool that enables us to build mathematical models to represent our documents numerically. The vocabulary is a set of all unique terms (bi-grams in this case) present in all the documents. The vocabulary is an unordered set and has no repeating elements. After the n-grams for each document are generated they are added to the vocabulary.

```
      static void create_bow(String[] ngrams) {
   bag_of_words.addAll(Arrays.asList(ngrams));
}
```

This function adds the bi-grams from each document to the bag_of_words Which is a static variable representing our vocabulary.

Phrase Extraction:

Since we are dealing with a large number of documents, a *feature selection* method must be employed to reduce the number of phrases to reduce processing time. The most simple and from our point of view the best method for plagiarism detection is a Document Frequency (DF) feature selection. According to the document frequency we can simply determine if the given phrase is important or not. The phrases existing just in one document are removed right away since they cannot be plagiarized in any other document.

```
static void document_frequency() {

    double df;
    doc_freq = new double[bag_of_words_arr.length];
    for (int i = 0; i < bag_of_words_arr.length; i++) {
        doc_freq[i] = 0.0;
        df = 0.0;
        for (String[] doc : all_ngrams) {
            for (String s : doc) {
                if (s.equals(bag_of_words_arr[i])) {
                    doc_freq[i] += 1;
                    break;
                }

            }
        }
    }
```

This function generates the document frequency for all terms in the vocabulary. The document frequencies are stored in a global variable called doc_freq.

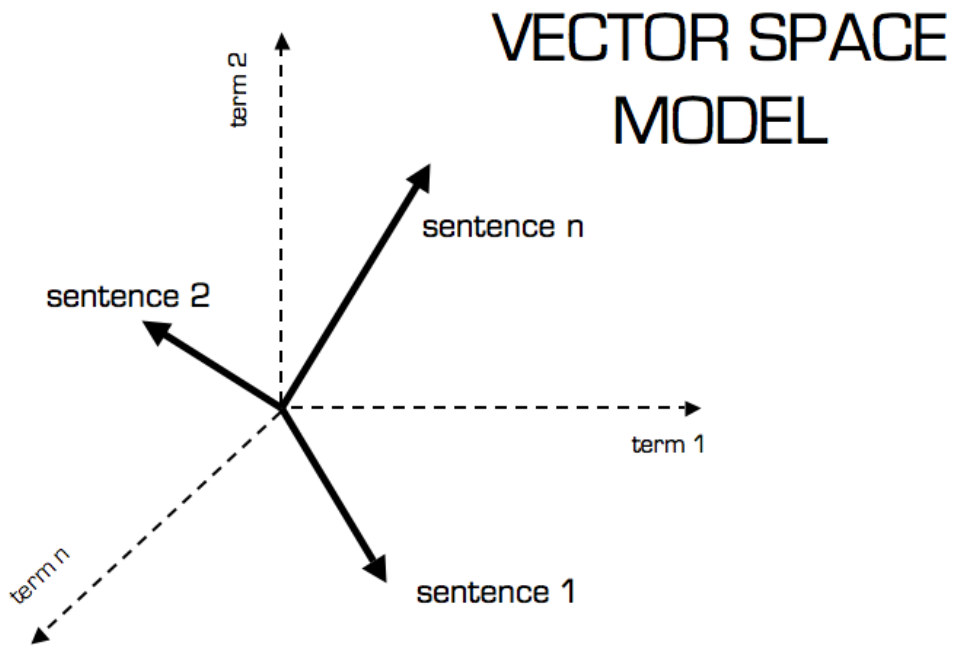
```
static void phrase_extraction() {
    StandardDeviation std = new StandardDeviation();
    Mean m = new Mean();
    System.out.println("printing doc frq");
    for(double j:doc_freq){
        System.out.println(j);
    }
    double std_dev = std.evaluate(doc_freq);
    double mean = m.evaluate(doc_freq, 0, doc_freq.length - 1);
    ArrayList<String> tmp_arr= new ArrayList<>(Arrays.asList(bag_of_words_arr));
    for(int i= tmp_arr.size()-1;i>=0;i--)
    {
        if(doc_freq[i]==1){//|| doc_freq[i]>std_dev+mean
            tmp_arr.remove(i);
        }
```

```
    }
    // to calculate phrases removed for normalization at end.
    double removed;
    for(int i=0;i<all_ngrams.size();i++){
        removed=0.0;
        for(String s: all_ngrams.get(i)){
            if(!tmp_arr.contains(s)){

                removed++;
            }
        }
        phr_red[i]=ph_org[i]-removed;
    }
    bag_of_words_arr=tmp_arr.toArray(new String[0]);


}
```

This function uses the document frequency to remove all words from the vocabulary that occur in only one document. The number of terms removed from each document because of feature selection is computed and stored as well for later use in normalizing the results.

Vector Space Model:

The vector space model is an algebraic model for representing text documents as vectors of terms. Each dimension of the vector space corresponds to a term. In this case the terms are the elements of our vocabulary.



Our documents will be represented as n dimensional vectors where n is the size of the vocabulary.

Term Frequency-Inverse Document Frequency:

The values along each dimension for a document vector are determined using TF-IDF weighting. TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. The formula used in this project is a light variation of the traditional one.

$$a_{i,j} = \begin{cases} \frac{1}{2} + \frac{PF_{i,j} \cdot \log\left(\frac{|N|}{DF_j}\right)}{2 \cdot \max_i\left(PF_{i,j}\right) \cdot \log\left(|N|\right)} & if\ phrase\ \ j\ occurs\ in\ \ document\ i \\ 0 & otherwise \end{cases}$$

PF i,j represents the occurrence frequency of phrase *j* in document *i*, *DFj* represents the number of documents where phrase *j* occurs.

```
static double[] phrase_frequency(String[] doc,int k) {   // divide by new length jeesus christ.
   double count = 0.0;

   double[] pf = new double[bag_of_words_arr.length];
   for (int j = 0; j < bag_of_words_arr.length; j++) {

      count = 0.0;
      for (int i = 0; i < doc.length; i++) {

         if (bag_of_words_arr[j].equals(doc[i])) {
            count++;
         }
      }
      pf[j] = count;
   }
```

Pharse frequency generates PF mentioned above.

```
static double[] tfidf_vectorization(double[] doc_pf) {
   double[] tfidf = new double[bag_of_words_arr.length];
   double max=Arrays.stream(doc_pf).max().getAsDouble();
   for (int i = 0; i < bag_of_words_arr.length; i++) {

      if (doc_pf[i] == 0.0){
         tfidf[i] = 0.0;
      }
      else{
         tfidf[i] = 0.5 + ((doc_pf[i] * Math.log(N / doc_freq[i])) / (2 * Math.log(N)*max));
      }

   }
   return tfidf;
}
```

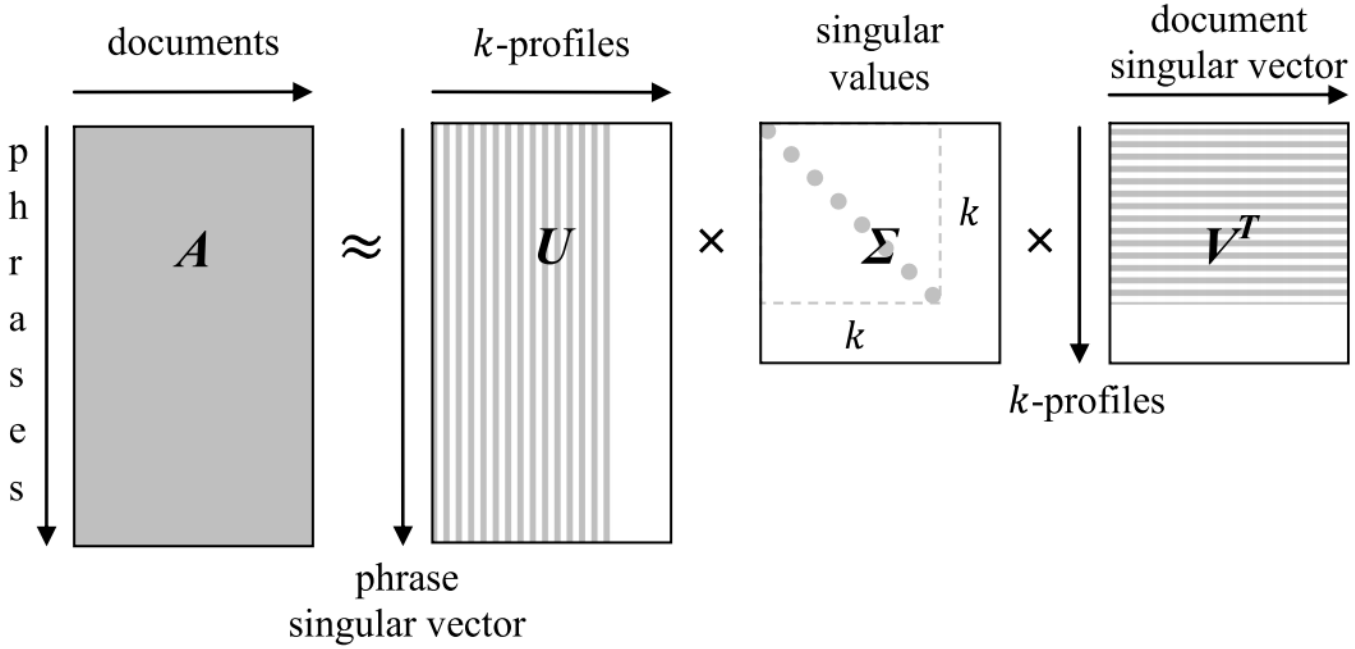
The tfidf_vecotization function then calculates and returns the document vector using the formula mentioned above and stores it in a two dimensional matrix for further use.

Similarity Calculation:

The final part involves processes to calculate the similarity between document vectors.

Latent semantic analysis:

Latent semantic analysis (LSA) is a technique in natural language processing, in particular distributional semantics, of analysing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. matrix containing word counts per document (rows represent unique words and columns represent each document) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns.



Document Similarity Normalization:

Before we use matrix VT, single elements of all document profiles must be rescaled with the corresponding singular values.

$B = \Sigma \times V^T$

Finally, we compute the mutual pairwise correlation according to Equation where the columns of matrix B are length-normalized. The resulting matrix is a symmetric matrix where each pair of documents is evaluated by a score representing the percentage similarity.

$$sim_{SVD} = \|B\|^T \times \|B\|$$

simSVD obtains much higher score for such documents where vast majority of phrases are removed as meaningless. Therefore, we need to normalize the results with the following formula.

$$sim(R,S) = sim_{SVD}(R,S) \cdot \sqrt{\frac{|ph_{orig}(R)|}{|ph_{red}(R)|} \cdot \frac{|ph_{orig}(S)|}{|ph_{red}(S)|}}$$

where the evaluation of documents R and S is weighted by the ratio between the number of original phrases |phorig| and the number of phrasesafter reduction |phred|.

```
static RealMatrix SVD(RealMatrix m) {
   SingularValueDecomposition svd = new SingularValueDecomposition(m.transpose());
   return svd.getS().multiply(svd.getVT());
}


static RealMatrix length_normalize(RealMatrix b) {
   int col_d = b.getColumnDimension();
   int row_d = b.getRowDimension();
   for (int i = 0; i < col_d; ++i) {
      double sum = 0.0;
      double length;
      for (int j = 0; j < row_d; j++) {
         //System.out.println(B.getEntry(j,i));
         sum += b.getEntry(j, i) * b.getEntry(j, i);
      }
      length = Math.sqrt(sum);
      System.out.print("length is ");
      System.out.println(length);

      for (int j = 0; j < row_d; j++) {
         b.multiplyEntry(j, i, 1.0 / length);
      }

   }
   return b;
```

```
}
```

svd gives the singular value decomposition of the document model matrix and length_normalize function length normalizes the columns of matrix b as per the formula.

```
for(int j=0;j< tmp.length;j++)
{
   if(phr_red[i]!=0 && phr_red[j]!=0) {
      tmp[i][j]=tmp[i][j] * Math.sqrt((ph_org[i] * ph_org[j]) / (phr_red[i] * phr_red[j]));
```

this piece of code then normalizes all the values in the similarity to get the final similarity score between two documents.

```java
// IntelliJ declarations start
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
        primaryStage.initStyle(StageStyle.UNDECORATED); //removes the borders, title and close, minimise options from frame
        Image icon = new Image("https://img.icons8.com/fluent/48/000000/detective.png");
        primaryStage.getIcons().add(icon);

        primaryStage.setScene(new Scene(root, 520, 400));
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);
    }
}
// IntelliJ declarations end (Only changes in line15: login.fxml and entire line 16)
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.text.Font?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="520.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.LoginController">
   <left>
      <AnchorPane prefHeight="407.0" prefWidth="228.0" BorderPane.alignment="CENTER">
         <children>
            <ImageView fx:id="brandingImageView" fitHeight="389.0" fitWidth="224.0" layoutX="7.0" layoutY="9.0" pickOnBounds="true" preserveRatio="true">
               <image>
                  <Image url="@../../../MiniProjectSem4/Images/projxlogin3.png" />
               </image>
            </ImageView>
         </children>
      </AnchorPane>
   </left>
   <right>
      <AnchorPane prefHeight="400.0" prefWidth="332.0" style="-fx-background-color: #ECECE7;" BorderPane.alignment="CENTER">
         <children>
            <ImageView fx:id="lockImageView" fitHeight="47.0" fitWidth="46.0" layoutX="121.0" layoutY="44.0" pickOnBounds="true" preserveRatio="true">
               <image>
                  <Image url="@../../../MiniProjectSem4/Images/lock.png" />
               </image>
            </ImageView>
            <Label layoutX="29.0" layoutY="126.0" text="Username">
               <font>
                  <Font size="13.0" />
               </font>
            </Label>
            <TextField fx:id="usernameTextField" layoutX="96.0" layoutY="123.0" prefHeight="26.0" prefWidth="173.0" promptText="Username" />
            <Label layoutX="29.0" layoutY="176.0" text="Password">
               <font>
                  <Font size="13.0" />
               </font>
            </Label>
            <PasswordField fx:id="enterPasswordField" layoutX="96.0" layoutY="173.0" prefHeight="26.0" prefWidth="173.0" promptText="Password" />
            <Label fx:id="loginMessageLabel" layoutX="34.0" layoutY="226.0" prefHeight="19.0" prefWidth="237.0" textFill="RED">
               <font>
                  <Font size="13.0" />
               </font>
            </Label>
            <Button fx:id="logInButton" layoutX="29.0" layoutY="277.0" mnemonicParsing="false" onAction="#loginButtonOnAction" prefHeight="26.0" prefWidth="240.0" text="LogIn" />
            <Button fx:id="cancelButton" layoutX="29.0" layoutY="316.0" mnemonicParsing="false" onAction="#cancelButtonOnAction" prefHeight="26.0" prefWidth="240.0" text="Cancel" />
         </children>
      </AnchorPane>
   </right>
</BorderPane>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.chart.BarChart?>
<?import javafx.scene.chart.CategoryAxis?>
<?import javafx.scene.chart.NumberAxis?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="768.0" prefWidth="1024.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Result">
   <children>
      <TextArea fx:id="resultAreaView" editable="false" layoutX="93.0" layoutY="179.0" prefHeight="183.0" prefWidth="714.0">
         <font>
            <Font name="Times New Roman" size="17.0" />
         </font></TextArea>
      <Label fx:id="resultView" layoutX="93.0" layoutY="136.0" prefHeight="27.0" prefWidth="564.0">
         <font>
            <Font name="System Bold Italic" size="18.0" />
         </font>
      </Label>
      <ImageView fx:id="mitView1" fitHeight="91.0" fitWidth="171.0" layoutX="1283.0" layoutY="38.0" pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@../../Images/mitadtlogo.png" />
         </image>
      </ImageView>
      <Button fx:id="show_result_view" layoutX="675.0" layoutY="137.0" mnemonicParsing="false" onAction="#showResultOnAction" prefHeight="26.0" prefWidth="131.0" text="Show Result" />
      <TextArea fx:id="resultAreaView1" editable="false" layoutX="93.0" layoutY="456.0" prefHeight="251.0" prefWidth="714.0">
         <font>
            <Font name="Times New Roman" size="17.0" />
         </font></TextArea>
      <Label fx:id="allResultView" layoutX="93.0" layoutY="413.0" prefHeight="27.0" prefWidth="106.0" text="All Results :">
         <font>
            <Font name="System Bold Italic" size="18.0" />
         </font>
      </Label>
      <Button fx:id="show_all_result_view" layoutX="675.0" layoutY="414.0" mnemonicParsing="false" onAction="#showAllResultOnAction" prefHeight="26.0" prefWidth="131.0" text="Show All Results" />
      <ImageView fx:id="projx" fitHeight="48.0" fitWidth="200.0" layoutX="93.0" layoutY="53.0" pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@../../Images/projectx.PNG" />
         </image>
      </ImageView>
      <BarChart fx:id="bc" layoutX="853.0" layoutY="179.0" prefHeight="529.0" prefWidth="615.0" title="Plagiarism Bar Chart">
        <xAxis>
          <NumberAxis label="Plagiarism (%)" side="BOTTOM" />
        </xAxis>
        <yAxis>
          <CategoryAxis label="File Names" side="LEFT" />
        </yAxis>
      </BarChart>
      <Button fx:id="barChartButton" layoutX="1085.0" layoutY="137.0" mnemonicParsing="false" onAction="#showBarChartOnAction" prefHeight="25.0" prefWidth="151.0" text="Show
```

```
54 Bar Chart" />
55     </children>
56 </AnchorPane>
57
```

```java
package sample;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;

import javafx.scene.chart.BarChart;
import javafx.scene.chart.CategoryAxis;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

import java.io.File;
import java.net.URL;
import java.util.ResourceBundle;

import static sample.Plagiarism.pdf_name;
import static sample.backend.*;

public class Result  implements Initializable{

    @FXML
    private TextArea resultAreaView1;
    @FXML
    private ImageView mitView1;
    @FXML
    private  ImageView projx;
    @FXML
    private TextArea resultAreaView;
    @FXML
    private Label resultView;
    @FXML
    private BarChart<String,Number> bc;
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        File brandingFile = new File("Images/mitadtlogo.png");
        Image brandingImage = new Image(brandingFile.toURI().toString());
        mitView1.setImage(brandingImage);
        File projxFile = new File("Images/projectx.PNG");
        Image projxImage = new Image(projxFile.toURI().toString());
        projx.setImage(projxImage);
        resultView.setText("Result of input file "+pdf_name);
    }

    public void showResultOnAction(ActionEvent event){
        resultAreaView.setText(ans_string);


    }
    public void showAllResultOnAction(ActionEvent event){
        resultAreaView1.setText(all_ans_string);
    }

    public void showBarChartOnAction(ActionEvent event){
        XYChart.Series series1= new XYChart.Series();
        series1.setName("Plagiarism result with comparison to " + my_file_name);
            for (int p = 0; p < tmp.length; p++) {
                series1.getData().add(new XYChart.Data(tmp[our_file_inx][p] * 100,
file_names.get(p) ));

            }

        bc.getData().addAll(series1);
    }

```

```
67
68
69
70
71 }
72
```

```java
package sample;

import javafx.fxml.FXML;
import javafx.scene.control.Label;
import opennlp.tools.lemmatizer.DictionaryLemmatizer;
import opennlp.tools.ngram.NGramGenerator;
import opennlp.tools.postag.POSModel;
import opennlp.tools.postag.POSTaggerME;
import org.apache.commons.math3.linear.MatrixUtils;
import org.apache.commons.math3.linear.RealMatrix;
import org.apache.commons.math3.linear.SingularValueDecomposition;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;
import java.io.*;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.LinkedHashSet;
import java.util.stream.Collectors;
import java.util.stream.Stream;
import java.lang.Runtime;
import org.apache.commons.math3.stat.descriptive.moment.StandardDeviation;
import org.apache.commons.math3.stat.descriptive.moment.Mean;

public class backend {
    static LinkedHashSet<String> bag_of_words = new LinkedHashSet<>();
    static ArrayList<String[]> all_ngrams = new ArrayList<>();
    static ArrayList<double[]> model = new ArrayList<>();
    static String[] bag_of_words_arr;
    static double N;
    static double[] doc_freq;
    static double[] ph_org;
    static double[] phr_red;
    static ArrayList<String> file_names= new ArrayList<>();
    static String my_file_name;
    static String ans_string;
    static String all_ans_string;
    static double[][] tmp;
    static int our_file_inx;

    public static void backend_process(String path,String pdf_name) throws IOException,
URISyntaxException {

        System.out.println("enetring main");
        my_file_name=pdf_name;

        System.out.println("starting preprocessing");
        preprocessing(path);
        System.out.println("preprocess done");
        System.out.println("printing bow\n ");
        for (String s : bag_of_words.toArray(new String[0])) {
            System.out.print(s + "        ||         ");
        }
        System.out.println();

        model_building();
        if(bag_of_words_arr.length!=0){
            similarity_cal();
        }
        else{
            ans_string="Results cannot be computed";
            all_ans_string="Results cannot be computed";
        }
```

```
 67         System.out.println("printing total memory");
 68         Runtime rt = Runtime.getRuntime();
 69         System.out.println(rt.totalMemory());
 70         System.out.println(rt.maxMemory());
 71
 72
 73     }
 74
 75     static void preprocessing(String folder_path) throws IOException, URISyntaxException
    {
 76         System.out.println("printing path"+folder_path);
 77         File folder = new File(folder_path);
 78         N = folder.listFiles().length;
 79         ph_org=new double[(int)N];
 80         phr_red=new double[(int)N];
 81         PDDocument document;
 82         int ptr=0;
 83         String content;
 84         String[] tokens;
 85         String[] tags;
 86         String[] ngrams;
 87         PDFTextStripper stripper = new PDFTextStripper();
 88         for (File file : folder.listFiles()) {
 89             file_names.add(file.getName());
 90             System.out.println("**********"+file.getName()+"**********");
 91             document = PDDocument.load(file);
 92             content = stripper.getText(document);
 93             document.close();
 94             content = remove_punc(content);
 95             System.out.println("AFTER REMOVING PUNCTUAUTION");
 96             System.out.println(content);
 97             System.out.println("before removing newline\n");
 98             System.out.println(content.length());
 99             System.out.println(content);
100             content = content.replaceAll("\\s{2,}", " ").trim();
101             System.out.println("after removing newline");
102
103             System.out.println(content);
104
105             tokens = remove_stopwords(content);
106             System.out.println("tokens are");
107             //System.out.println(tokens.length);
108             for (String s : tokens) {
109                 System.out.print(s+"   ");
110             }
111             System.out.println();
112             tags = POS_tagging(tokens);
113             tokens = lemmatization(tags, tokens);
114             System.out.println("printing lemmas");
115             for (String s : tokens) {
116                 System.out.print(s+"   ");
117             }
118             System.out.println();
119
120             ngrams = n_gram(tokens);
121             ph_org[ptr]= ngrams.length;
122             all_ngrams.add(ngrams);      // required later for model building
123             System.out.println("printng trigrams");
124             for (String s : ngrams) {
125                 System.out.println(s);
126             }
127             create_bow(ngrams);
128
129             ptr++;
130         }
131
132
```

```
133     }
134
135     static String remove_punc(String content) {
136         content = content.replaceAll("\\p{Punct}", " ");
137         return content;
138     }
139
140     static String[] remove_stopwords(String content) throws IOException {
141         ArrayList<String> stopwords;
142         stopwords = (ArrayList<String>) Files.readAllLines(Paths.get("src/sample/
    stopwords.txt"));
143         ArrayList<String> allWords1 = Stream.of(content.toLowerCase().split(" "))
144                 .collect(Collectors.toCollection(ArrayList<String>::new));
145         allWords1.removeAll(stopwords);
146         return allWords1.toArray(new String[0]);
147     }
148
149     static String[] POS_tagging(String[] tokens) throws IOException {
150         InputStream inputStreamPOSTagger = new FileInputStream("src/sample/en-pos-maxent
    .bin");
151         POSModel posModel = new POSModel(inputStreamPOSTagger);
152         POSTaggerME posTagger = new POSTaggerME(posModel);
153         return posTagger.tag(tokens);
154     }
155
156     static String[] lemmatization(String[] tags, String[] tokens) throws IOException {
157         String[] lemmas;
158         InputStream DictLemmatizer = new FileInputStream("src/sample/lemmatizer-dict.txt
    ");
159         DictionaryLemmatizer lemmatizer = new DictionaryLemmatizer(DictLemmatizer);
160         lemmas = lemmatizer.lemmatize(tokens, tags);
161         for (int i = 0; i < lemmas.length; i++) {
162             if (lemmas[i].equals("O")) {
163                 lemmas[i] = tokens[i];
164             }
165         }
166         return lemmas;
167     }
168
169     static String[] n_gram(String[] tokens) {
170         return NGramGenerator.generate(Arrays.asList(tokens), 2, " ").toArray(new String
    [0]);
171     }
172
173     static void create_bow(String[] ngrams) {
174         bag_of_words.addAll(Arrays.asList(ngrams));
175     }
176
177
178     static void model_building() {
179
180         System.out.println("ENTERING MODEL BUILDING");
181         bag_of_words_arr = bag_of_words.toArray(new String[0]);
182        // System.out.println("INITIAL BOW SIZE BEFORE REDUCTION");
183        // System.out.println(bag_of_words_arr.length);
184         //System.out.println("OLD DOC FREQ, USING OLD BOW");
185         document_frequency();// doc_freq after ph extraction., nned doc freq for
    reduction then we recompute doc_freq.
186         //for(double i: doc_freq){
187        //     System.out.print(i+"  ");
188        // }
189         phrase_extraction();
190         document_frequency();
191         if(bag_of_words_arr.length==0){
192             return;
193         }
194         //System.out.println("FINAL BOW AFTER REDUCTION.");
```

```
195         // System.out.println(bag_of_words_arr.length);
196         System.out.println("PRINTING NEW BOW");
197         for (String s : bag_of_words_arr) {
198             System.out.print(s + "        ||         ");
199         }
200         System.out.println();
201         System.out.println("printing NEW DOC FREQ");
202         for (double i : doc_freq) {
203             System.out.print(i + "  ");
204         }
205         System.out.println();
206 
207         double[] pf;
208         String[] doc;
209         double[] tfidf;
210         for (int k=0;k< all_ngrams.size();k++) {
211             doc=all_ngrams.get(k);
212             pf = phrase_frequency(doc,k);
213             // to normalize the raw phrase frequency
214             //double max= Arrays.stream(pf).max().getAsDouble();
215             //dviding raw count by max
216             //for(int i=0;i<pf.length;i++){
217             //  pf[i]=pf[i]/max;
218             // }
219             System.out.println("printing PHRASE FREQUENCY VECTOR");
220             for (double i : pf) {
221                 System.out.print(i + " ");
222             }
223             System.out.println();
224 
225             tfidf = tfidf_vectorization(pf);
226             System.out.println("printing TF-IDF");
227             for (double i : tfidf) {
228                 System.out.print(i + " ");
229             }
230             System.out.println();
231 
232             model.add(tfidf);
233 
234         }
235     }
236 
237     static double[] phrase_frequency(String[] doc,int k) {   // divide by new length
    jeesus christ.
238         double count = 0.0;
239 
240         double[] pf = new double[bag_of_words_arr.length];
241         for (int j = 0; j < bag_of_words_arr.length; j++) {
242 
243             count = 0.0;
244             for (int i = 0; i < doc.length; i++) {
245 
246                 if (bag_of_words_arr[j].equals(doc[i])) {
247                     count++;
248                 }
249             }
250             pf[j] = count;
251         }
252 
253 
254         return pf;
255     }
256 
257     static void document_frequency() {
258 
259         double df;
260         doc_freq = new double[bag_of_words_arr.length];
```

```
261         for (int i = 0; i < bag_of_words_arr.length; i++) {
262             doc_freq[i] = 0.0;
263             df = 0.0;
264             for (String[] doc : all_ngrams) {
265                 for (String s : doc) {
266                     if (s.equals(bag_of_words_arr[i])) {
267                         doc_freq[i] += 1;
268                         break;
269                     }
270 
271                 }
272             }
273         }
274 
275 
276     }
277 
278     static double[] tfidf_vectorization(double[] doc_pf) {
279         double[] tfidf = new double[bag_of_words_arr.length];
280         double max=Arrays.stream(doc_pf).max().getAsDouble();
281         for (int i = 0; i < bag_of_words_arr.length; i++) {
282 
283             if (doc_pf[i] == 0.0){
284                 tfidf[i] = 0.0;
285             }
286             else{
287                 tfidf[i] = 0.5 + ((doc_pf[i] * Math.log(N / doc_freq[i])) / (2 * Math.
    log(N)*max));
288             }
289 
290         }
291         return tfidf;
292     }
293 
294     static void similarity_cal() {
295 
296         RealMatrix m = MatrixUtils.createRealMatrix(model.toArray(new double[0][]));
297         RealMatrix b;
298         b = SVD(m);
299         System.out.println();
300         System.out.println("pritning B INITIAL");
301         tmp = b.getData();
302         for (double[] arr : tmp) {
303             for (double i : arr) {
304                 System.out.print(i + "  ");
305             }
306             System.out.println();
307         }
308         System.out.println();
309         b = length_normalize(b);
310         System.out.println("PRINTING B LENGTH NORMALIZED");
311         tmp = b.getData();
312         for (double[] arr : tmp) {
313             for (double i : arr) {
314                 System.out.print(i + "  ");
315             }
316             System.out.println();
317         }
318         RealMatrix sim = b.transpose().multiply(b);
319 
320 
321         System.out.println("printing og and removed phrases");
322         for(double d: ph_org){
323             System.out.print(d);
324             System.out.print("  ");
325         }
326         System.out.println();
```

```
327         for(double d: phr_red){
328             System.out.print(d);
329             System.out.print("  ");
330         }
331         System.out.println();
332         System.out.println("printing similarity matrix");
333         tmp = sim.getData();
334         for (double[] arr : tmp) {
335             for (double i : arr) {
336                 System.out.print(i + "  ");
337             }
338             System.out.println();
339         }
340         System.out.println("after normalization");// something wrong in normalization
341         //only slected file
342         our_file_inx= file_names.indexOf(my_file_name);
343
344
345
346
347
348         for(int i=0;i<tmp.length;i++)
349         {
350             all_ans_string=all_ans_string+"\nThe plagiarism results for "+file_names.get
    (i)+" is as follows:\n";
351             for(int j=0;j< tmp.length;j++)
352             {
353                 if(phr_red[i]!=0 && phr_red[j]!=0) {
354                     tmp[i][j]=tmp[i][j] * Math.sqrt((ph_org[i] * ph_org[j]) / (phr_red[i
    ] * phr_red[j]));
355                     all_ans_string=all_ans_string+file_names.get(j)+ " : "+String.format
    ("%.2f",tmp[i][j]*100)+" %\n";          //+tmp[i][j]*100+" % \n"
356                     System.out.print("similarity between "+file_names.get(i)+" and "+
    file_names.get(j)+" is ");
357                     System.out.println(tmp[i][j]*100);
358                     //System.out.print(" ");
359                 }
360                 else{
361                     System.out.print(0+" ");
362                 }
363             }
364            // System.out.println();
365         }
366
367         ans_string= "";
368         System.out.println("\n\n\nprinting pnly important file\n\n\n");
369        if(our_file_inx!=-1) {
370            ans_string="the similarity result for "+file_names.get(our_file_inx)+" are:\n
     ";
371            for (int k = 0; k < tmp.length; k++) {
372
373                System.out.print(  file_names.get(k) + " is : ");
374                System.out.println(tmp[our_file_inx][k] * 100 + "%");
375                ans_string = ans_string +  file_names.get(k) + " is : " + String.format(
    "%.2f",tmp[our_file_inx][k] * 100) + " %\n";
376            }
377            System.out.println("printing ans string \n");
378            System.out.println(ans_string);
379        }
380        else{
381            ans_string="a file was not selected";
382        }
383     }
384
385     static RealMatrix SVD(RealMatrix m) {
386         SingularValueDecomposition svd = new SingularValueDecomposition(m.transpose());
387         return svd.getS().multiply(svd.getVT());
```

```
388     }
389
390     static RealMatrix length_normalize(RealMatrix b) {
391         int col_d = b.getColumnDimension();
392         int row_d = b.getRowDimension();
393         for (int i = 0; i < col_d; ++i) {
394             double sum = 0.0;
395             double length;
396             for (int j = 0; j < row_d; j++) {
397                 //System.out.println(B.getEntry(j,i));
398                 sum += b.getEntry(j, i) * b.getEntry(j, i);
399             }
400             length = Math.sqrt(sum);
401             System.out.print("length is ");
402             System.out.println(length);
403
404             for (int j = 0; j < row_d; j++) {
405                 b.multiplyEntry(j, i, 1.0 / length);
406             }
407
408         }
409         return b;
410
411     }
412
413     static void phrase_extraction() {
414         StandardDeviation std = new StandardDeviation();
415         Mean m = new Mean();
416         System.out.println("printing doc frq");
417         for(double j:doc_freq){
418             System.out.println(j);
419         }
420         double std_dev = std.evaluate(doc_freq);
421         double mean = m.evaluate(doc_freq, 0, doc_freq.length - 1);
422         ArrayList<String> tmp_arr= new ArrayList<>(Arrays.asList(bag_of_words_arr));
423         for(int i= tmp_arr.size()-1;i>=0;i--)
424         {
425             if(doc_freq[i]==1){//|| doc_freq[i]>std_dev+mean
426                 tmp_arr.remove(i);
427             }
428
429         }
430         // to calculate phrases removed for normalization at end.
431         double removed;
432         for(int i=0;i<all_ngrams.size();i++){
433             removed=0.0;
434             for(String s: all_ngrams.get(i)){
435                 if(!tmp_arr.contains(s)){
436
437                     removed++;
438                 }
439             }
440             phr_red[i]=ph_org[i]-removed;
441         }
442         bag_of_words_arr=tmp_arr.toArray(new String[0]);
443
444
445
446     }
447
448     static void printing_fn(){
449
450     }
451
452
453 }
454
```

455
456
457
458

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.Cursor?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane fx:id="anchorView" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Homescreen">
   <children>
      <Button fx:id="browseButton" layoutX="464.0" layoutY="188.0" mnemonicParsing="false" onAction="#handleButtonAction" prefHeight="26.0" prefWidth="103.0" text="Browse" />
      <TextField fx:id="directoryView" layoutX="33.0" layoutY="188.0" prefHeight="26.0" prefWidth="401.0" promptText="Project database directory...." />
      <Label layoutX="33.0" layoutY="127.0" prefHeight="26.0" prefWidth="281.0" text="Choose directory for the project database:">
         <font>
            <Font size="14.0" />
         </font>
      </Label>
      <ImageView fx:id="mitView" fitHeight="91.0" fitWidth="171.0" layoutX="396.0" layoutY="27.0" pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@../../../MiniProjectSem4/Images/mitadtlogo.png" />
         </image>
      </ImageView>
      <Button fx:id="nextButton" layoutX="255.0" layoutY="263.0" mnemonicParsing="false" onAction="#nextViewOnAction" prefHeight="26.0" prefWidth="89.0" text="Next">
         <font>
            <Font name="Bell MT" size="14.0" />
         </font>
      </Button>
      <ImageView fx:id="projx" fitHeight="48.0" fitWidth="200.0" layoutX="33.0" layoutY="34.0" pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@../../Images/projectx.PNG" />
         </image>
      </ImageView>
      <Label fx:id="folderErrorLabel" layoutX="224.0" layoutY="325.0" prefHeight="18.0" prefWidth="150.0" textFill="#fa0202d7">
         <font>
            <Font name="System Bold" size="12.0" />
         </font>
      </Label>
   </children>
   <cursor>
      <Cursor fx:constant="DEFAULT" />
   </cursor>
</AnchorPane>

```

```java
package sample;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;

import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.VBox;
import javafx.stage.DirectoryChooser;
import javafx.stage.Stage;

import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

public class Homescreen implements Initializable {
    static String d;
    @FXML
    private ImageView mitView;
    @FXML
    private ImageView projx;
    @FXML
    private AnchorPane anchorView;
    @FXML
    private TextField directoryView;
    @FXML
    private Button nextButton;
    @FXML
    private Label folderErrorLabel;


    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        File brandingFile = new File("Images/mitadtlogo.png");
        Image brandingImage = new Image(brandingFile.toURI().toString());
        mitView.setImage(brandingImage);
        File projxFile = new File("Images/projectx.PNG");
        Image projxImage = new Image(projxFile.toURI().toString());
        projx.setImage(projxImage);
    }

    @FXML
    private void handleButtonAction(ActionEvent event) {
        final DirectoryChooser dirchooser = new DirectoryChooser();
        Stage stage2 = (Stage) anchorView.getScene().getWindow();
        File file = dirchooser.showDialog(stage2);
        if (file != null) {
            System.out.println("Path : " + file.getAbsolutePath());
            directoryView.setText(file.getAbsolutePath());
            d= file.getAbsolutePath();
        }
    }
    public String getStr(){
        return d;
    }
    @FXML
    private void nextViewOnAction(ActionEvent event) {
        if(d!=null)
```

```
68          {
69              Stage stage2 = (Stage) nextButton.getScene().getWindow();
70              stage2.close();
71
72              try {
73                  FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("
    plagiarism.fxml"));
74                  Parent root = (Parent) fxmlLoader.load();
75                  Stage stage3 = new Stage();
76                  Image icon = new Image("https://img.icons8.com/fluent/48/000000/
    detective.png");
77                  stage3.getIcons().add(icon);
78                  stage3.setTitle("ProjectX");
79                  stage3.setScene(new Scene(root, 600, 400));
80                  stage3.setResizable(false);
81                  stage3.show();
82              } catch (IOException e) {
83                  e.printStackTrace();
84              }
85          }
86          else{
87              System.out.println("choose folder");
88              folderErrorLabel.setText("Please choose a directory");
89          }
90      }
91  }
92
93
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane fx:id="anchor3View" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Plagiarism">
   <children>
      <ImageView fx:id="mitView1" fitHeight="91.0" fitWidth="171.0" layoutX="406.0" layoutY="37.0" pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@../../Images/mitadtlogo.png" />
         </image>
      </ImageView>
      <Label layoutX="43.0" layoutY="131.0" text="Choose the project pdf file to check for plagiarism :">
         <font>
            <Font name="System Italic" size="14.0" />
         </font>
      </Label>
      <Button fx:id="startButton" layoutX="261.0" layoutY="267.0" mnemonicParsing="false" onAction="#startButtonOnAction" prefHeight="26.0" prefWidth="77.0" text="Start">
         <font>
            <Font name="Bell MT" size="14.0" />
         </font>
      </Button>
      <Button fx:id="browsePdfView" layoutX="474.0" layoutY="198.0" mnemonicParsing="false" onAction="#browsePdfOnAction" prefHeight="26.0" prefWidth="103.0" text="Browse" />
      <TextField fx:id="pdfDirectoryView" layoutX="43.0" layoutY="198.0" prefHeight="26.0" prefWidth="401.0" promptText="Locate the Project PDF...." />
      <ImageView fx:id="projx1" fitHeight="48.0" fitWidth="200.0" layoutX="43.0" layoutY="51.0" onMouseClicked="#goHome" pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@../../Images/projectx.PNG" />
         </image>
      </ImageView>
      <Label fx:id="waitLabel" layoutX="210.0" layoutY="324.0" prefHeight="20.0" prefWidth="253.0" textFill="#ff0000ae">
         <font>
            <Font name="System Bold Italic" size="14.0" />
         </font>
      </Label>
   </children>
</AnchorPane>

```

```java
package sample;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Group;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.AnchorPane;
import javafx.stage.DirectoryChooser;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ResourceBundle;

import static java.nio.file.StandardCopyOption.REPLACE_EXISTING;
import static sample.backend.ans_string;

public class Plagiarism implements Initializable {
    static String  pdf_name;



    @FXML
    private ImageView mitView1;
    @FXML
    private AnchorPane anchor3View;
    @FXML
    private ImageView projx1;
    @FXML
    private TextField pdfDirectoryView;
    @FXML
    private Button startButton;
    @FXML
    private Label waitLabel;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        File brandingFile = new File("Images/mitadtlogo.png");
        Image brandingImage = new Image(brandingFile.toURI().toString());
        mitView1.setImage(brandingImage);
        File projxFile = new File("Images/projectx.PNG");
        Image projxImage = new Image(projxFile.toURI().toString());
        projx1.setImage(projxImage);
    }
    private Path to;
    private Path from;
    private File selectedFile;


    public void browsePdfOnAction(ActionEvent event) throws IOException{
        FileChooser fc = new FileChooser();
        fc.setTitle("Attach a file");
```

```
 68         fc.getExtensionFilters().addAll(
 69                 new FileChooser.ExtensionFilter("PDF Files", "*.pdf"));
 70         selectedFile = fc.showOpenDialog(null);
 71 
 72         if (selectedFile != null) {
 73             pdfDirectoryView.setText(selectedFile.getAbsolutePath());
 74             from = Paths.get(selectedFile.toURI());
 75             Homescreen boo = new Homescreen();
 76             pdf_name=selectedFile.getName();
 77             if(boo.getStr()!=null) {
 78                 to = Paths.get(boo.getStr() + System.getProperty("file.separator") +
    selectedFile.getName());
 79                 System.out.println(to);
 80                 System.out.println(from);
 81                 Files.copy(from, to, REPLACE_EXISTING);
 82             }
 83 
 84         }
 85     }
 86 
 87 
 88 
 89     public void startButtonOnAction(ActionEvent event) throws IOException,
    URISyntaxException {
 90 
 91         Stage stage3 = (Stage) startButton.getScene().getWindow();
 92 
 93         Homescreen b= new Homescreen();
 94 
 95         waitLabel.setText("Processing please wait");
 96         System.out.println("printing labbel content"+waitLabel.getText());
 97         backend.backend_process(b.getStr(),pdf_name);
 98             if(to!=null){
 99             File new_location = new File(to.toUri());
100             new_location.delete();
101             }
102 
103 
104 
105 
106         stage3.close();
107 
108 
109         try {
110             FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("result.fxml"
    ));
111             Parent root = (Parent) fxmlLoader.load();
112             Stage stage4 = new Stage();
113             Image icon = new Image("https://img.icons8.com/fluent/96/000000/detective.
    png");
114             stage4.getIcons().add(icon);
115             stage4.setScene(new Scene(root, 1024, 768));
116             stage4.setTitle("ProjectX");
117             stage4.setMaximized(true);
118             stage4.show();
119         } catch (IOException e) {
120             e.printStackTrace();
121         }
122         }
123         public void goHome(){
124         System.out.println("home button printed");
125            Stage stage3 = (Stage) projx1.getScene().getWindow();
126            stage3.close();
127 
128             try {
129                 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("
    homescreen.fxml"));
```

```
130                 Parent root = (Parent) fxmlLoader.load();
131                 Stage stage2 = new Stage();
132 
133                 stage2.setTitle("ProjectX");
134 
135                 stage2.setScene(new Scene(root, 600, 400));
136                 stage2.show();
137             } catch (IOException e) {
138                 e.printStackTrace();
139             }
140         }
141 
142 
143 
144 }
```

```java
// this file has code

package sample;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.stage.StageStyle;

import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ResourceBundle;
import java.net.URL;

public class LoginController implements Initializable {

    @FXML
    private  Button cancelButton;
    @FXML
    private Label loginMessageLabel;
    @FXML
    private ImageView brandingImageView;
    @FXML
    private ImageView lockImageView;
    @FXML
    private TextField usernameTextField;
    @FXML
    private PasswordField enterPasswordField;


    @Override
    public void initialize(URL url, ResourceBundle resourceBundle){
        File brandingFile= new File("Images/projxlogin3.png");
        Image brandingImage= new Image(brandingFile.toURI().toString());
        brandingImageView.setImage(brandingImage);

        File lockFile= new File("Images/lock.png");
        Image lockImage= new Image(lockFile.toURI().toString());
        lockImageView.setImage(lockImage);


    }

    public void loginButtonOnAction(ActionEvent event){

        if (usernameTextField.getText().isBlank() == false && enterPasswordField.getText
().isBlank() == false) {
            validateLogin();
        } else {
            loginMessageLabel.setText("Please enter username and password");
        }
    }
    public void cancelButtonOnAction(ActionEvent event) {
        Stage stage = (Stage) cancelButton.getScene().getWindow();
```

```
 67         stage.close();
 68     }
 69 
 70     public void validateLogin(){
 71         DatabaseConnection connectNow = new DatabaseConnection();
 72         Connection connectDb = connectNow.getConnection();
 73 
 74         String verifyLogin = "SELECT count(1) FROM user_account WHERE username = '" +
    usernameTextField.getText() + "' AND password = '" + enterPasswordField.getText() + "' "
    ;
 75 
 76         try {
 77             Statement statement = connectDb.createStatement();
 78             ResultSet queryResult = statement.executeQuery(verifyLogin);
 79 
 80             while(queryResult.next()){
 81                 if (queryResult.getInt(1)==1) {
 82                     loginMessageLabel.setText("Congratulations");
 83                     Stage stage = (Stage) cancelButton.getScene().getWindow();
 84                     stage.close();
 85                     try {
 86                       FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("
    homescreen.fxml"));
 87                       Parent root = (Parent) fxmlLoader.load();
 88                       Stage stage2 = new Stage();
 89                       Image icon = new Image("https://img.icons8.com/fluent/48/000000/
    detective.png");
 90                       stage2.getIcons().add(icon);
 91                       stage2.setTitle("ProjectX");
 92                       stage2.setScene(new Scene(root, 600, 400));
 93                       stage2.setResizable(false);
 94                       stage2.show();
 95                     } catch (IOException e) {
 96                         e.printStackTrace();
 97                     }
 98                 }
 99                 else {
100                     loginMessageLabel.setText("Invalid login. Please try again");
101 
102                 }
103 
104             }
105 
106         } catch (Exception e) {
107             e.printStackTrace();
108             e.getCause();
109         }
110     }
111 
112 
113 }
114 
```

```java
// This file deals with database connection
package sample;
import java.sql.Connection;
import java.sql.DriverManager;

public class DatabaseConnection {
    public Connection databaseLink;

    public Connection getConnection(){
        String databaseName= "proj";
        String databaseUser= "revolt";
        String databasePassword= "Kingreyno@24";
        String url = "jdbc:mysql://localhost/" + databaseName;

        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            databaseLink = DriverManager.getConnection(url,databaseUser,databasePassword
);

        }catch (Exception e){
            e.printStackTrace();
            e.getCause();
        }
        return  databaseLink;
    }
}

```

References:

[1] Ceska Z. (2008) Plagiarism Detection Based on Singular Value Decomposition. In: Nordström B., Ranta A. (eds) Advances in Natural Language Processing. GoTAL 2008. Lecture Notes in Computer Science, vol 5221. Springer, Berlin, Heidelberg.

Conclusion:

Mini Project Plagiarism Detector is a robust plagiarism detection software.
The key concept is to detect the plagiarism between the inserted documents to prevent malpractices.
The user with minimum knowledge about computers can be able to operate the software easily.
We learnt a lot of new things about the NLP(Natural Language Processing), Java and Software Development, and were able to successfully implement the project in limited time using team and time management.

Future Scope:

More accurate backend algorithm can be implemented.
GUI can be enhanced.
Client Server Architecture can be implemented, and the project can be modified to the entire Mini Project Management System.

Mini Project Tracker (Group.: 08) Guide: Prof. Suresh Kapare

Task in Progress Task Completed

PROJECT START DATE	21-Mar-21	SCROLL TO WEEK #	1

Sprint	Priority	Start	Finish	Duration	Status	% Complete
Project Summary		**21-Mar-21**	**26-Jun-21**	**70 day(s)**		**100%**
Rohan Bhange (Front End)		**21-Mar-21**	**26-Jun-21**	**70 day(s)**		
Research	High	21-Mar-21	10-Apr-21	15 day(s)	Green	100%
Learning (Language and Platform)	Medium	11-Apr-21	25-Apr-21	10 day(s)	Green	100%
Coding	High	26-Apr-21	16-May-21	15 day(s)	Green	100%
Implementation	High	17-May-21	31-May-21	11 day(s)	Green	100%
Front End Testing	High	1-Jun-21	9-Jun-21	7 day(s)	Green	100%
Integration	High	10-Jun-21	20-Jun-21	7 day(s)	Green	100%
Final Testing	High	21-Jun-21	26-Jun-21	5 day(s)	Green	100%
Burhanuddin Unwalla (Back End)		**21-Mar-21**	**26-Jun-21**	**70 day(s)**		
Research	High	21-Mar-21	10-Apr-21	15 day(s)	Green	100%
Learning (Language and Platform)	Medium	11-Apr-21	25-Apr-21	10 day(s)	Green	100%
Coding	High	26-Apr-21	16-May-21	15 day(s)	Green	100%
Implementation	High	17-May-21	31-May-21	11 day(s)	Green	100%
Back End Testing	High	1-Jun-21	9-Jun-21	7 day(s)	Green	100%
Integration	High	10-Jun-21	20-Jun-21	7 day(s)	Green	100%
Final Testing	High	21-Jun-21	26-Jun-21	5 day(s)	Green	100%