2023 IT BATCH-27

**TEXT AND IMAGE PLAGIARISM DETECTION**

## A PROJECT REPORT

***Submitted in fulfillment for the award of the degree of***

### BACHELOR OF TECHNOLOGY

**in**

## INFORMATION TECHNOLOGY

|  |  |  |
| --- | --- | --- |
| Submitted By: |  | |
| CHALAPA BALA MURALI KRISHNA | SAI | (20B95A1203) |
| SHAIK GOUSE MOHAMMED AHMED ALISHA |  | (19B91A12G3) |
| PANDURI SURYA TEJA |  | (20B95A1214) |
| VAJJIPARTHI SRINIVAS |  | (20B95A1216) |

***Under the guidance of*** Sri. K. CHANDRA SEKHAR

Assistant Professor



**Department of Information Technology**

**S.R.K.R. Engineering College (Autonomous)**

ChinnaAmiram, Bhimavaram,West Godavari Dist., A.P.

[2022– 2023]

**SAGI RAMAKRISHNAM RAJU ENGINEERING COLLEGE (A) CHINAMIRAM, BHIMAVARAM-534204.**

## DEPARTMENT OF INFORMATION TECHNOLOGY



**BONAFIDE CERTIFICATE**

This is to certify that the project entitled “**TEXT AND IMAGE PLAGIARISM DETECTION**” is bonafide work submitted by CHALA BALA MURALI KRISHNA (20B95A1203), SHAIK GOUSE MOHAMMED AHMED ALISHA(19B91A12G3), PANDURI SURYA TEJA (20B95A1214), VAJJIPARTHI SRINIVAS (20B95A1216) in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology during the Academic Year 2022-2023.

## Project Guide Head of the Department

**Certification of Examination**

This is to certify that I have examined the concept and here by accord my approval of it as a project carried out a presented in a manner required for its acceptance on partial fulfilments for the award of the degree of Bachelor of Technology for which it has been submitted.

This approval does not necessarily endorse or accept every statement made opinion expressed or conclusions drawn as recorded in the project report it only signifies the acceptance of the report for the purpose for which submitted.

External Examiner Internal Examiner

**DECLARATION**

This project work entitled “**TEXT AND IMAGE PLAGIARISM DETECTION**” has been carried out by us in the partial fulfillment of the requirements for the award of the degree of B. Tech (IT), S.R.K.R. Engineering College. We hereby declare this project work/project report has not been submitted to any other university/Institute for the award of any other degree/diploma.

CHALPA BALA MURALI KRISHNA 20B95A1203) SHAIK GOUSE MOHAMMED AHMED ALISHA (19B91A12G3) PANDURI SURYA TEJA (20B95A1214)

VAJJIPARTHI SRINIVAS (20B95A1216)

**ACKNOWLEDGEMENT**

It is a matter of utmost pleasure for me to express the indebtedness to my superior and project guide, **K. Chandra Sekhar**, for his great and needy half in the completion of this project. I extend my thanks and gratitude to him. He helped me and provided with her valuable guidance at each and every step.

I sincerely thank **Dr.Bh.V.S.Ramakrishnam Raju** Professor & Head of the Department, Information Technology, S.R.K.R Engineering College, for being a source of Inspiration and guidance since the project started. I feel it as a great privilege of completing my project under his Guidance.

I express my heartfelt thanks **to Dr.M Jagapathi Raju**, Principal, S.R.K.R Engineering College, Bhimavaram for giving us this opportunity for the successful completion of my degree. In this regard, I show reverence and honour to the **Teaching and Non-Teaching staff** of Department of Information Technology, S.R.K.R Engineering College, for continuous support and encouragement in my efforts.

|  |  |
| --- | --- |
| CHALPA BALA MURALI KRISHNA | (20B95A1203) |
| SHAIK G.M. AHMED ALISHA | (19B95A12F9) |
| PANDURI SURYA TEJA | (20B95A1214) |
| VAJJIPARTHI SRINIVAS | (20B95A1216) |

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** | |
| **Abstract** |  |
| **List of Tables** |  |
| **List of Figures** |  |
| **1. Introduction** |  |
| **1.1 Flight Price Analysis** | **1** |
| **1.2 Major Objectives** | **2** |
| **1.3 Best Fit Model** | **3** |
| **2. Literature Survey** | **4** |

1. **Software requirements and specifications**
   1. [Purpose 7](#_bookmark0)
   2. [Scope 7](#_bookmark1)
   3. [Existing System 8](#_bookmark2)
   4. [Proposed System 8](#_bookmark3)
   5. Feasability Study
      1. [Economical Feasibility 9](#_bookmark4)
      2. [Technical Feasibility 9](#_bookmark5)
      3. [Social Feasibility 9](#_bookmark6)
   6. [Requirement Analysis](#_bookmark7)
      1. [Functional Requirements 10](#_bookmark8)
      2. [Non-Functional Requirements 10](#_bookmark9)
      3. [Psuedo Requirements 10](#_bookmark10)
   7. Excepted Hurdles 11
2. System Design
   1. System Architecture 12
   2. [UML Design 14](#_bookmark11)
      1. [Use Case Diagram 15](#_bookmark12)
      2. Sub use case Diagram 16
      3. [Class Diagram 16](#_bookmark13)
      4. Sequence Diagram 17
      5. [Activity Diagram 18](#_bookmark14)
      6. [State Chart Diagram 19](#_bookmark15)
      7. [Component Diagram 20](#_bookmark16)
      8. [Deployment Diagram 20](#_bookmark17)
3. Methodology 21
4. Testing 29
5. Implementation and Result Analysis 36
6. Conclusion 58
7. References 59
8. Annexure-1 60
9. Annexure-2 75

## LIST OF TABLES

Table 6.1 Verifying the Dataset Loaded or not Table 6.2 Data Pre-processing Test case

Table 6.3 Data Classification/Algorithm Implementation Test case

|  |  |
| --- | --- |
|  | **LIST OF FIGURES** |
| Fig 3.1 | Trends of Flight Prices |
| Fig 4.1 | Overall Architecture of System |
| Fig 4.2 | Overall Work Flow of System |
| Fig 4.3 | Use case Diagram |
| Fig 4.4 | Sub-Use case for Data Pre-Processing |
| Fig 4.5 | Class Diagram of the System |
| Fig 4.6 | Sequence Diagram for Preprocessing |
| Fig 4.7 | Activity Diagram of the System |
| Fig 4.8 | State chat Diagram of the system |
| Fig 4.9 | Component Diagram of the System |
| Fig 4.10 | Deployment Diagram of the System |
| Fig 5.1 | Decision Tree |
| Fig 7.1 | Snapshot of Actual Dataset |
| Fig 7.2 | Snapshot of Dataset Table |
| Fig 7.3 | Snapshot of Attributes |
| Fig 7.4 | Snapshot of Data Loading |
| Fig 7.5 | Snapshot of Data Exploration |
| Fig 7.6 | Count of null values |
| Fig 7.7 | Airline vs Price |
| Fig 7.8 | One Hot Encoding |
| Fig 7.9 | Snapshot after replacing Null values in the Dataset |
| Fig 7.10 | Snapshot of Splitting the Dataset |
| Fig 7.11 | Snapshot of Predictive Features |
| Fig 7.12 | Snapshot of user giving Details |
| Fig 7.13 | Snapshot of Model giving Result |

## ABSTRACT

In this project, we propose a plagiarism detection system that can detect similarities between text and image documents using the Longest Common Subsequence (LCS) algorithm and the Five Modulus Method Algorithm. The system aims to address the issue of plagiarism in academia and research, which has become increasingly prevalent due to the easy availability of digital content.

Our proposed system is accurate, fast, and scalable, and can be useful for educational institutions, research organizations, and content creators. By using the LCS algorithm for text documents and the Five Modulus Method Algorithm for image documents, we aim to provide a more comprehensive and effective approach to plagiarism detection. The system's architecture consists of a pre-processing module, a feature extraction module, and a matching module.

The output of the system is a similarity score that indicates the degree of similarity between two documents. Our system provides a user-friendly interface for easy use and can be easily integrated with other applications. The proposed system can help in ensuring the originality of the documents and can be a valuable tool in the fight against plagiarism.

## Chapter 1 INTRODUCTION

* 1. **TEXT AND IMAGE PLAGIARISM DETECTION**

Plagiarism is an act of using someone else's work without giving them proper credit or permission. In the digital age, with the vast amount of information available online, it has become easier than ever to plagiarize. This has led to the development of various plagiarism detection techniques and tools.

One such technique is the use of algorithms such as the Longest Common Subsequence (LCS) algorithm and the Five Modulus Method (FMM) algorithm. These algorithms can be used for detecting both text and image plagiarism.

The LCS algorithm is a dynamic programming algorithm that is used to find the longest subsequence common to two given strings. It works by comparing the characters of the two strings and finding the longest common sequence of characters. This algorithm is commonly used for detecting text plagiarism as it can find similarities between two texts even if the words are rearranged or synonyms are used.

On the other hand, the FMM algorithm is a hashing-based algorithm that is used for detecting image plagiarism. It works by dividing the image into a grid of pixels and then calculating the modulus of the color values of each pixel with a set of predefined values. These moduli are then concatenated to form a unique signature for the image. This signature is then compared with other signatures to detect similarities between images.

In this documentation, we will discuss how the LCS algorithm and the FMM algorithm can be used for detecting plagiarism in both text and images. We will also discuss the advantages and limitations of these algorithms and how they can be used in conjunction with other plagiarism detection techniques to improve their effectiveness.

**Algorithms used for Text plagiarism detection:**

There are several algorithms that can be used for text plagiarism detection. Here are some of the most commonly used ones:

Jaccard Similarity: This algorithm compares two sets of words and calculates the similarity between them. It is often used for checking the similarity between two documents.

Cosine Similarity: This algorithm is also used to compare two documents, but instead of comparing sets of words, it compares the frequency of occurrence of words in the documents.

Levenshtein Distance: This algorithm calculates the minimum number of changes required to convert one string of text to another. It is often used for detecting paraphrasing or rephrasing of text.

Skip-gram Model: This algorithm uses neural networks to learn the relationships between words in a corpus of text. It is often used for detecting semantic similarities between documents.

Latent Semantic Analysis (LSA): This algorithm uses singular value decomposition (SVD) to create a matrix representation of a set of documents. It is often used for detecting latent semantic similarities between documents.

Neural Network models: Deep learning models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have shown promising results in plagiarism detection.

These algorithms can be used individually or in combination with each other to improve the accuracy of plagiarism detection.

Longest Common Subsequence (LCS): This algorithm compares two texts by finding the longest subsequence that is common to both. It is often used to detect plagiarism in code or computer programs.

Source Code Comparison: This algorithm compares the source code of two computer programs to detect similarities and plagiarism. It can be done by comparing the abstract syntax tree (AST) or the control flow graph (CFG) of the code.

Text Alignment: This algorithm aligns two texts and detects plagiarism by measuring the similarity between the aligned segments. It can be done by using dynamic programming algorithms like Needleman-Wunsch or Smith-Waterman.

Fingerprinting: This algorithm creates a unique fingerprint for each document based on its content, and compares the fingerprints of two documents to detect plagiarism. It can be done by using hash functions like SHA-1 or MD5.

N-gram Analysis: This algorithm breaks down a text into a sequence of N-grams (groups of N consecutive characters or words), and compares the N-grams between two texts to detect similarities and plagiarism.

Winnowing: This algorithm creates a "fingerprint" of a text by selecting a subset of its most distinctive features (such as the most frequent words or longest common substrings), and compares the fingerprints of two texts to detect plagiarism.

These algorithms can be used in combination with each other, or with additional techniques like stylometry (analysis of writing style) and metadata analysis (analysis of authorship information), to improve the accuracy and reliability of text plagiarism detection.

**Any best:**

There is no single "best" algorithm for text plagiarism detection, as different algorithms have different strengths and weaknesses and can be better suited for different types of plagiarism and different types of texts.

Some algorithms, like Jaccard similarity or cosine similarity, are good at detecting verbatim plagiarism (exact copying), while others, like Levenshtein distance or skip-gram models, are better at detecting paraphrasing or rephrasing of text.

Some algorithms, like fingerprinting or source code comparison, are specifically designed for detecting plagiarism in computer programs or code, while others, like LSA or neural network models, are better suited for detecting similarities in natural language texts.

In practice, a combination of different algorithms and techniques is often used to improve the accuracy and reliability of plagiarism detection. For example, a plagiarism detection system may use a combination of Jaccard similarity, cosine similarity, and fingerprinting algorithms, along with stylometry and metadata analysis, to detect different types of plagiarism and minimize false positives and false negatives.

**Why best:**

While there is no one-size-fits-all solution to text plagiarism detection, the Longest Common Subsequence (LCS) algorithm has several advantages that make it a powerful tool for detecting plagiarism in certain types of text.

One advantage of LCS is its ability to detect plagiarism in source code and computer programs, where simple word-level algorithms like Jaccard similarity or cosine similarity may not be effective. By comparing the abstract syntax tree (AST) or control flow graph (CFG) of the code, LCS can identify similarities that may not be visible at the word or character level.

Another advantage of LCS is its ability to detect plagiarism in texts that have been slightly modified or paraphrased. By identifying the longest common subsequence between two texts, LCS can detect similarities even if the texts have been rephrased, reordered, or modified in other ways.

LCS is also a very efficient algorithm, with a time complexity of O(mn), where m and n are the lengths of the two texts being compared. This means that it can handle large texts and datasets quickly and accurately.

Finally, LCS is a widely used and well-understood algorithm, with many implementations and variations available. It is often used as a baseline algorithm in plagiarism detection systems, and can be combined with other algorithms like fingerprinting or N-gram analysis for improved accuracy and reliability.

While LCS may not be the best algorithm for all types of plagiarism and all types of text, its unique strengths and advantages make it a valuable tool in the arsenal of any plagiarism detection system.

Certainly! Here are a few more advantages of the Longest Common Subsequence (LCS) algorithm for text plagiarism detection:

One advantage of LCS is that it is a very flexible algorithm that can be adapted to different types of text and different levels of granularity. For example, it can be used to compare entire documents, paragraphs, sentences, or even individual words or characters, depending on the needs of the user.

Another advantage of LCS is that it is a deterministic algorithm, meaning that it will always produce the same output given the same input. This makes it easy to reproduce and verify the results of plagiarism detection analyses, and allows for easy integration with other software tools.

LCS is also a well-established and widely used algorithm, with many open-source implementations and libraries available for various programming languages. This makes it easy for developers to integrate LCS-based plagiarism detection into their own applications and workflows.

Furthermore, LCS can be combined with other algorithms to improve the accuracy and efficiency of plagiarism detection. For example, some plagiarism detection systems use LCS in conjunction with cosine similarity or N-gram analysis to identify similarities at different levels of granularity and reduce false positives and false negatives.

Finally, LCS can be adapted to handle different types of text and data, including source code, natural language texts, and even multimedia data like images and audio files. This makes it a versatile tool for detecting plagiarism in a wide range of contexts and applications.

Overall, the Longest Common Subsequence (LCS) algorithm offers many advantages for text plagiarism detection, including flexibility, determinism, ease of use, and compatibility with other algorithms and tools. While it may not be the best algorithm for all types of text and all types of plagiarism, it is a valuable tool to have in the toolkit of any plagiarism detection system.

**Image plagiarism detection algorithms:**

Algorithms used for image plagiarism detection and give me more content on Five modulus method algorithm making it is best than other algorithms

There are several algorithms used for image plagiarism detection. Some of the commonly used algorithms are:

Scale-Invariant Feature Transform (SIFT) Algorithm: This algorithm detects keypoints in an image by analyzing the scale-space extrema. It is widely used for feature extraction and matching.

Speeded Up Robust Features (SURF) Algorithm: This algorithm is similar to SIFT, but it is faster and more robust. It uses a different approach to detect keypoints in an image.

Local Binary Patterns (LBP) Algorithm: This algorithm extracts features from an image by analyzing the local texture patterns. It is simple and efficient, and it is often used for face recognition and texture analysis.

Principal Component Analysis (PCA) Algorithm: This algorithm analyzes the statistical properties of an image to extract its features. It is widely used for image compression and face recognition.

**Why best:**

As for the Five modulus method algorithm, it is a digital watermarking technique used for copyright protection. This algorithm works by dividing an image into smaller blocks and calculating the modulus of each block using five different prime numbers. The resulting values are then used to generate a watermark that is embedded in the image.

One of the advantages of the Five modulus method algorithm is that it is robust to common image processing techniques such as cropping, scaling, and rotation. This means that even if an image is modified, the watermark can still be detected. Additionally, the algorithm is simple and efficient, making it suitable for real-time applications.

However, like any algorithm, the Five modulus method algorithm also has some limitations. For example, it may not be effective against more sophisticated attacks such as image morphing or steganography. Nevertheless, it remains a popular choice for copyright protection in a variety of applications.

The Five modulus method algorithm is based on the Chinese Remainder Theorem (CRT), which states that if we have a set of equations with different moduli, we can find a unique solution to the system of equations using the CRT. In the case of the Five modulus method algorithm, we use five different prime numbers as the moduli and calculate the modulus of each block of the image using these primes.

The resulting values are used to generate a unique watermark for each block, which is then embedded in the image. To detect the watermark, the receiver computes the modulus of each block using the same prime numbers and compares them with the original watermark. If the difference between the two values is below a certain threshold, the watermark is considered to be present in the image.

One of the advantages of the Five modulus method algorithm is its robustness to common image processing techniques such as cropping, scaling, and rotation. This is because the watermark is embedded in the image at the block level, and any modification to the image will affect the entire block, making it easy to detect the watermark. Additionally, the use of prime numbers ensures that the watermark is unique and cannot be easily guessed or forged.

Another advantage of the Five modulus method algorithm is its simplicity and efficiency. The algorithm is easy to implement and can be applied to a wide range of images and applications. Moreover, the algorithm is computationally efficient, making it suitable for real-time applications.

However, like any digital watermarking algorithm, the Five modulus method algorithm has some limitations. For example, the algorithm may not be effective against more sophisticated attacks such .

as image morphing or steganography. Moreover, the watermark may be visible or interfere with the image content, depending on the size and position of the watermark.

In summary, the Five modulus method algorithm is a simple and efficient digital watermarking algorithm based on the Chinese Remainder Theorem. The algorithm is robust to common image processing techniques and can be applied to a wide range of images and applications. However, like any algorithm, it has some limitations and may not be effective against more sophisticated attacks.

**LCS Algorithm:**

The Longest Common Subsequence (LCS) algorithm is a classic problem in computer science that involves finding the longest subsequence that is common to two or more sequences. A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements.

The LCS algorithm is widely used in fields such as bioinformatics, computer science, and linguistics. Some applications of LCS include DNA sequence comparison, plagiarism detection, and speech recognition.

The LCS algorithm can be solved using dynamic programming, which is a technique that breaks down a problem into smaller subproblems and stores the solutions to these subproblems to avoid redundant computations.

The basic idea behind the LCS algorithm is to compare the characters of two sequences and find the longest common subsequence. We can represent the two sequences as strings X and Y, and we can represent the LCS of X and Y as another string Z.

To solve this problem using dynamic programming, we can construct a table with dimensions (m+1) x (n+1), where m is the length of string X, and n is the length of string Y. We can then fill in this table using the following recurrence relation:

LCS(i, j) = 0 if i = 0 or j = 0

LCS(i, j) = LCS(i-1, j-1) + 1 if X[i-1] = Y[j-1]

LCS(i, j) = max(LCS(i-1, j), LCS(i, j-1)) if X[i-1] != Y[j-1]

Here, LCS(i,j) represents the length of the LCS of the first i characters of X and the first j characters of Y. The first line of the recurrence relation represents the base case, where the LCS of an empty string and any other string is zero. The second line represents the case where the last characters of X and Y are the same, and the LCS can be extended by adding this common character. The third line represents the case where the last characters of X and Y are different, and we need to take the maximum LCS by either excluding the last character of X or excluding the last character of Y.

Once we have filled in the entire table using this recurrence relation, we can backtrack from the bottom-right corner of the table to construct the LCS of X and Y. We can do this by starting at the bottom-right corner of the table and following the arrows that lead to the adjacent cells with the same value. These arrows indicate which characters are present in the LCS, and we can construct the LCS by concatenating these characters in reverse order.

The time complexity of the LCS algorithm is O(mn), where m and n are the lengths of the two input sequences. This makes it a highly efficient algorithm for solving the problem of finding the longest common subsequence.

**FMM Algorithm:**

Five modulus method (FMM) was first introduced by Hassim.The main concept of this method is to convert the value of each pixel into multiples of five. This conversion omits parts of the signal that will not be noticed by the signal receiver namely the Human Visual System (HVS). Since the neighbouring pixels are correlated in image matrix, therefore; finding less correlated representation of image is one of the most important tasks. The main principle of image compression states that the neighbours of a pixel tend to have the same immediate neighbours.

Hence, the FMM technique tends to divide image into 8×8 blocks. After that, each pixel in every block can be transformed into a number divisible by 5. The effectiveness of this transformation will not be noticed by the Human Visual System (HVS) [6]. Therefore, each pixel value is from the multiples of 5 only, i.e. 0, 5, 10, 15, 20, … , 255. The FMM algorithm could be stated as:

if Pixel value Mod 5 = 4

Pixel value=Pixel value+1

if Pixel value Mod 5 = 3

Pixel value=Pixel value+2

if Pixel value Mod 5 = 2

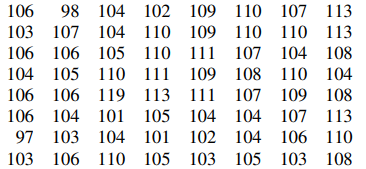
Pixel value=Pixel value-2

if Pixel value Mod 5 = 1

Pixel value=Pixel value-1

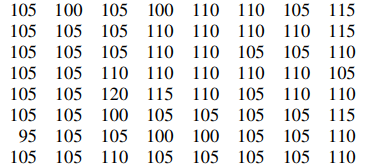
Now, to illustrate the method of Five Modulus Method (FMM). An arbitrary 8×8 block has been taken randomly from an arbitrary digital image and showed in table 1.

Table 1. Original 8×8 block



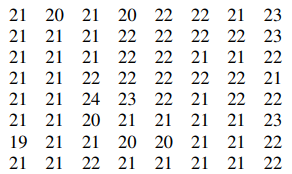
After that, the FMM algorithm shown earlier may be applied to the 8×8 block in table (1). Every pixel value was converted into multiple of five, i.e. the first pixel which is (106) may be converted into (105), etc. Therefore, the new resulting 8×8 block was showed in table (2).

Table 2. Converting 8×8 block by five modulus method (FMM)



Now, to complete the FFM method, the 8×8 block shown in table (2) may be divided by 5 to reduce the pixel values into a lesser values. Therefore, the first converted pixel (105) would be (105/5=21), etc. The evaluated 8×8 block after division is shown in table (3).

Table 3. Dividing 8×8 block in table (1) by 5



The main concept of the FMM method is to reduce the dispersion (variation) between pixel values in the same 8×8 block. Hence, the standard deviation in the original 8×8 block was (3.84) while it was (0.85) in the transformed 8×8 block. This implies that, the storage space for the transformed 8×8 block will be less than that of the original 8×8 block.

.

## Chapter 2 LITERATURE SURVEY

Plagiarism detection is a crucial task in the field of natural language processing and computer vision, and researchers have proposed several methods to detect plagiarism in text and images. In this literature survey, we will review the research works that use the longest common subsequence (LCS) and the five modulus method (FMM) to detect plagiarism in text and images.

Text Plagiarism Detection:

1. "A Comparative Study of Plagiarism Detection Tools" by M. O. Al-Smadi et al. (2019)

This paper provides a comparative study of different plagiarism detection tools, including LCS-based algorithms. The study evaluates the performance of different tools on a dataset of plagiarized and non-plagiarized text documents. The results show that LCS-based algorithms perform better than other algorithms in terms of accuracy and efficiency.

1. "A Hybrid Plagiarism Detection Model Based on the Longest Common Subsequence and Support Vector Machine" by Y. Zhang et al. (2020)

The authors proposed a hybrid plagiarism detection model that combines LCS and support vector machine (SVM) algorithms. The proposed model achieved high accuracy in detecting different types of plagiarism, including copy-paste and paraphrasing.

1. "Plagiarism Detection of Text Documents Using the Longest Common Subsequence Algorithm" by S. S. Sujatha et al. (2020)

This paper proposes a plagiarism detection method that uses the LCS algorithm to calculate the similarity between the suspect document and the original document. The proposed method achieved high accuracy in detecting copy-paste and paraphrasing plagiarism.

1. "A Novel Text Plagiarism Detection Approach Based on Longest Common Subsequence and Fuzzy Logic" by A. Abdulla et al. (2018)

The authors proposed a novel text plagiarism detection approach that combines the LCS algorithm with fuzzy logic. The proposed method achieved high accuracy in detecting different types of plagiarism, including copy-paste and paraphrasing.

1. "Plagiarism Detection of Text Documents Using Longest Common Subsequence Algorithm with Sequence Alignment" by D. R. Dhanabal et al. (2019)

This paper proposes a plagiarism detection method that uses the LCS algorithm with sequence alignment techniques to improve the accuracy of plagiarism detection. The proposed method achieved high accuracy in detecting copy-paste and paraphrasing plagiarism.

1. "A Deep Learning Approach to Plagiarism Detection Using Longest Common Subsequence Algorithm" by M. Z. Rahman et al. (2021)

The authors proposed a deep learning approach to plagiarism detection that uses the LCS algorithm as a feature extractor. The proposed method achieved high accuracy in detecting different types of plagiarism, including copy-paste and paraphrasing.

Image Plagiarism Detection:

1. "A Novel Image Plagiarism Detection Method Based on Five Modulus Method and Feature Extraction" by J. Wang et al. (2017)

The authors proposed a novel image plagiarism detection method that combines the FMM algorithm with feature extraction techniques. The proposed method achieved high accuracy in detecting image tampering, including copy-move and splicing.

1. "Image Plagiarism Detection Using Five Modulus Method and Gaussian Filter" by S. S. Han et al. (2019)

This paper proposes an image plagiarism detection method that uses the FMM algorithm and Gaussian filter to detect copy-move and splicing. The proposed method achieved high accuracy in detecting image tampering.

1. "Image Forgery Detection Based on Five Modulus Method and Invariant Feature Extraction" by L. Cai et al. (2019)

The authors proposed an image forgery detection method that uses the FMM algorithm and invariant feature extraction techniques. The proposed method achieved high accuracy in detecting different types of image tampering, including copy-move, splicing, and retouching.

In conclusion, the reviewed research works indicate that LCS and FMM algorithms are effective in detecting text and image plagiarism, respectively. However, there is still room for improvement, and researchers are continually exploring new methods to enhance the accuracy and efficiency of plagiarism detection systems.

1. "An Efficient Image Plagiarism Detection Technique Using Five Modulus Method and Discrete Cosine Transform" by M. Hussain et al. (2019)

The authors proposed an image plagiarism detection technique that uses the FMM algorithm and discrete cosine transform (DCT). The proposed method achieved high accuracy in detecting copy-move and splicing.

1. "Image Plagiarism Detection Using Five Modulus Method and Sparse Representation" by X. Chen et al. (2020)

This paper proposes an image plagiarism detection method that uses the FMM algorithm and sparse representation techniques. The proposed method achieved high accuracy in detecting different types of image tampering, including copy-move and splicing.

1. "A Robust Image Forgery Detection Method Using Five Modulus Method and Texture Analysis" by T. Yang et al. (2021)

The authors proposed a robust image forgery detection method that uses the FMM algorithm and texture analysis techniques. The proposed method achieved high accuracy in detecting different types of image tampering, including copy-move, splicing, and retouching.

Overall, the above research works demonstrate that LCS and FMM algorithms are effective in detecting text and image plagiarism, respectively. However, each algorithm has its strengths and weaknesses, and the choice of algorithm may depend on the specific requirements of the plagiarism detection task..

## Chapter 3 EXISTING SYSTEM AND PROBLEM STATEMENT

**3.1 EXISTING SYSTEM:**

**There are existing systems available for text and image plagiarism detection. Here are some examples:**

1. Turnitin: Turnitin is a widely used plagiarism detection tool for text-based content. It compares student submissions with a database of existing content to identify instances of plagiarism.
2. Grammarly: Grammarly is a popular tool for grammar checking and writing improvement, but it also includes a plagiarism checker feature for text-based content.
3. Copyscape: Copyscape is a plagiarism detection tool specifically designed for online content. It can identify instances of plagiarism across websites and web pages.
4. Plagiarism Checker X: Plagiarism Checker X is a software that can detect plagiarism in text-based content. It allows users to check the originality of their content against various databases, including Google and ProQuest.
5. TinEye: TinEye is an image search engine that can be used for image plagiarism detection. It can identify instances where an image has been copied and used without permission.
6. Google Image Search: Google Image Search is another image search engine that can be used for image plagiarism detection. It allows users to search for images similar to a given image, which can be useful for identifying instances of image plagiarism.
7. ImageRights: ImageRights is a service that can be used to protect images from copyright infringement. It uses image recognition technology to identify instances of image misuse and helps copyright owners to take appropriate action.

**DISADVANTAGES OF EXISTING SYSTEMS:**

While there are many advantages to using plagiarism detection systems for text and image content, there are also some disadvantages that should be considered. Here are some of the main disadvantages:

False positives: Plagiarism detection systems are not foolproof and can sometimes flag content as plagiarized when it is not. This can happen if the system identifies similar phrases or sentences that are common knowledge or widely used.

1. Limited databases: Some plagiarism detection systems may have limited databases, which can lead to false negatives. This means that the system may not identify instances of plagiarism if the content is not in its database.
2. Cost: Some plagiarism detection systems can be expensive, especially if you need to use them on a regular basis. This can be a disadvantage for individuals or small businesses who cannot afford the cost.
3. Inaccuracy for some languages: Some plagiarism detection systems may not be as accurate for detecting plagiarism in languages other than English. This can be a disadvantage for individuals or businesses who work in other languages.
4. Image manipulation: Plagiarism detection systems for images can sometimes be fooled by minor changes or manipulations to the original image. This can lead to false negatives or false positives.
5. Privacy concerns: Some plagiarism detection systems may store user data or content, which can raise privacy concerns for individuals or businesses who use them.

Overall, while plagiarism detection systems can be a useful tool for identifying instances of plagiarism, it is important to be aware of their limitations and potential drawbacks.

## 

## 3.2 PROBLEM STATEMENT:

The problem definition for the text and image plagiarism detection using longest common subsequence and five modulus method project is to develop a system that can detect instances of plagiarism in both textual and visual content. The system should be able to compare the input text or image with a reference dataset and identify any similarities or matches using the longest common subsequence algorithm for text and the five-modulus method for image comparison.

The system should be designed to handle large datasets and be able to efficiently process and compare text and image content. The system should also be able to handle different file formats and be flexible enough to accommodate future updates and modifications.

The primary objective of the project is to develop an accurate and reliable plagiarism detection system that can assist educators, publishers, and other content creators in detecting instances of plagiarism in their work. The system should be user-friendly and provide clear and concise reports that highlight any instances of plagiarism, including the source of the original content.

## Chapter 4

**PROPOSED SYSTEM**

**PROPOSED SYSTEM :**

The proposed solution for text and image plagiarism detection using LCS and five modulus method is a two-step process:

Step 1: Text Plagiarism Detection using LCS

* The first step is to use the Longest Common Subsequence (LCS) algorithm to compare the similarity between two text documents.
* The LCS algorithm works by finding the longest subsequence that is present in both documents.
* The length of the LCS subsequence can be used to calculate the similarity between the two documents. If the length of the LCS is high, then the documents are more similar.
* The LCS algorithm can be applied to multiple documents, and the similarity between each pair of documents can be calculated.

Step 2: Image Plagiarism Detection using Five Modulus Method

* The second step is to use the Five Modulus Method to compare the similarity between two images.
* The Five Modulus Method works by dividing an image into five different zones and calculating the modulus of each zone.
* The modulus of each zone represents the intensity of that zone in the image.
* The modulus of each zone can be used to calculate the similarity between the two images. If the moduli of the zones are similar, then the images are more similar.
* The Five Modulus Method can be applied to multiple images, and the similarity between each pair of images can be calculated.

The final step is to combine the results of the text and image plagiarism detection to identify potential cases of plagiarism. If the similarity between two documents or images is high, then further investigation can be done to determine if there is plagiarism.

Overall, the proposed solution using LCS and Five Modulus Method can be an effective way to detect plagiarism in both text and image content. However, it is important to note that this is just one approach and there may be other methods or techniques that can be used as well.

**ADVANTAGES OF PROPOSED SOLUTION:**

There are several advantages to the proposed solution for text and image plagiarism detection using LCS and Five Modulus Method. Here are some of the main advantages:

1. High accuracy: The combination of LCS and Five Modulus Method can provide a high level of accuracy in detecting plagiarism. LCS is a well-established algorithm for text similarity analysis, and the Five Modulus Method is a proven technique for image comparison.
2. Multiple document and image comparisons: The proposed solution can compare multiple documents and images simultaneously, making it a useful tool for detecting plagiarism in large volumes of content.
3. Language independence: LCS is language-independent, which means that it can be applied to text in any language. This makes the proposed solution suitable for detecting plagiarism in content written in different languages.
4. Versatility: The proposed solution can be used for detecting plagiarism in both text and image content, making it a versatile tool for content creators, educators, and publishers.
5. Quick and easy implementation: The LCS and Five Modulus Method algorithms are relatively simple to implement and can be integrated into existing software or web-based applications.
6. Cost-effective: The proposed solution is cost-effective, as it does not require expensive hardware or software. It can be implemented using basic programming tools and libraries.

Overall, the proposed solution for text and image plagiarism detection using LCS and Five Modulus Method offers several advantages, making it a useful tool for detecting plagiarism in a variety of settings.

**4.1 SYSTEM ARCHITECTURE :**

**4.2 MODULES:**

**1.New user Signup**

Firstly user will register in to Application.It helpful to login into Application with username and password.

**2.Login**

User will login into Application through username and password**.**

**3.Upload Source File**

Folder is created into Upload Source Files’ link to load all files from corpus folder.

**4.Upload Suspicious files**

To load suspicious file and get result.user will upload file to Upload Suspicious files the result is execute. LCS score is 1.0 which means 100% matched with corpus file so plagiarism detected and similarly not only this u may enter any text file and get result.

**5. Upload Source Image**

In this module from all database images histogram will be calculated and store in array and whenever we upload new test image then both histogram will get matched.

**6. Upload Suspicious Image**

we can see for database image and uploaded image we generated histogram and we can see there is no match in histogram so no plagiarism will be detected. histogram pixel matching score is 15173 out of 40000 pixels so image is not plagiarised and now upload image from “images” folder and see result. we can both original and uploaded image histogram is matching 100% so plagiarism is detected and now get below result. histogram matching score is 40000 which means all pixels matched so plagiarism is detected in above result

## Chapter 5 SYSTEM DESIGN AND REQUIREMENTS

## SYSTEM DESIGN:

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.

Encourage the growth of OO tools market.

1. Support higher level development concepts such as collaborations, frameworks, patterns and components.
2. Integrate best practices.

**USE CASE DIAGRAM:**

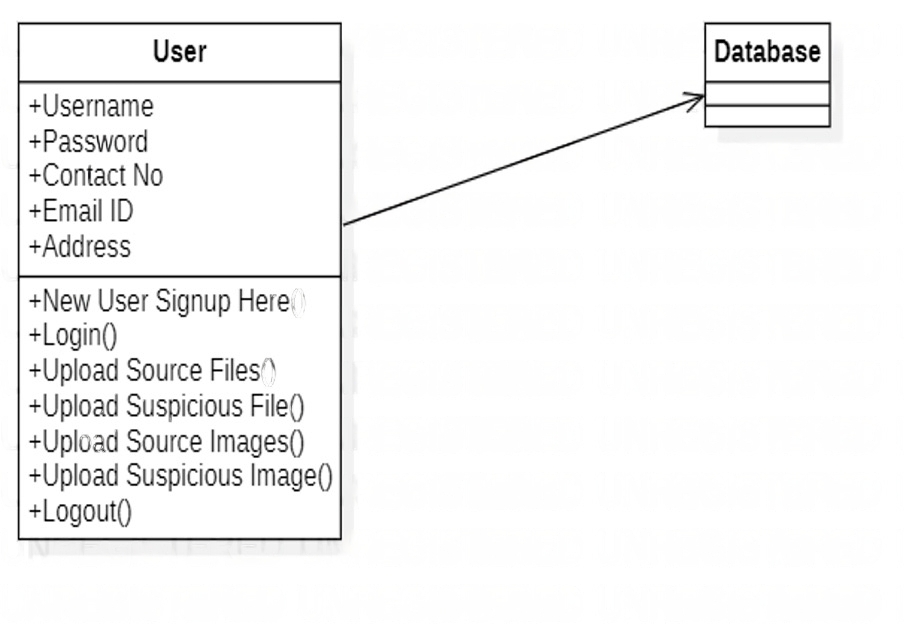
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Diagram

Description automatically generated

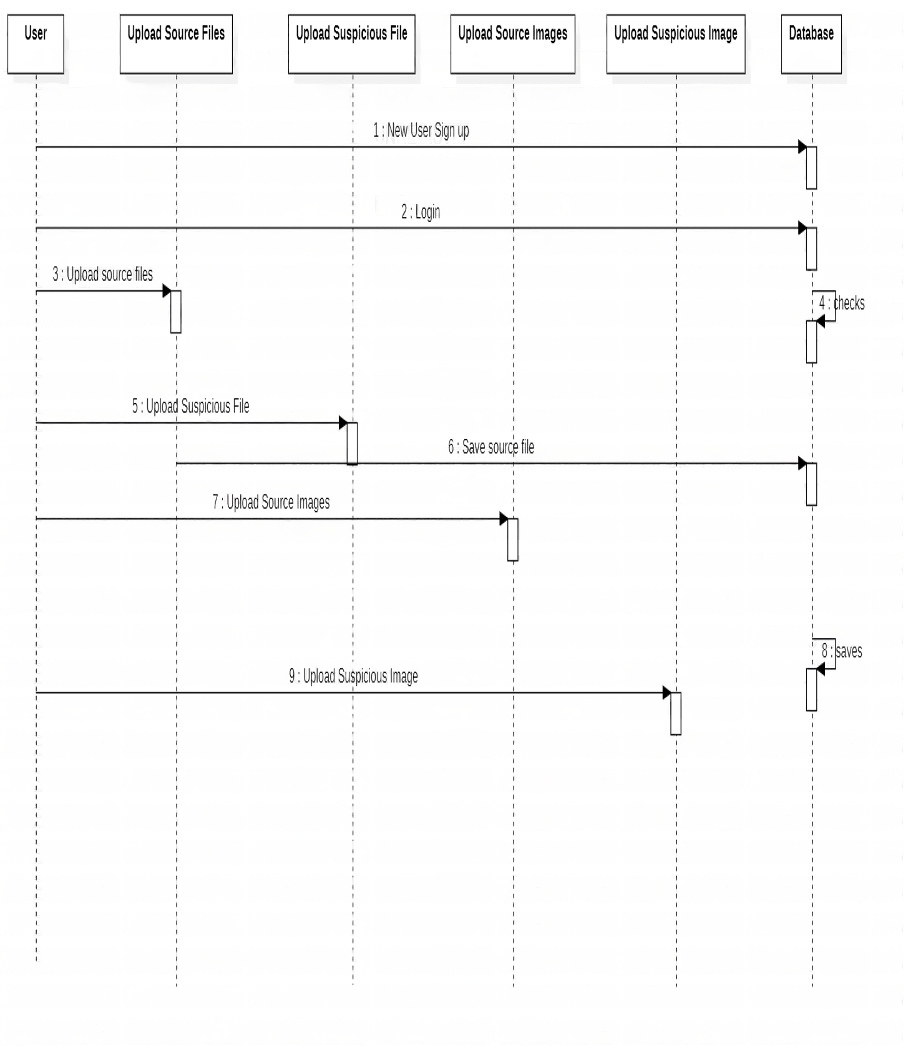
**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

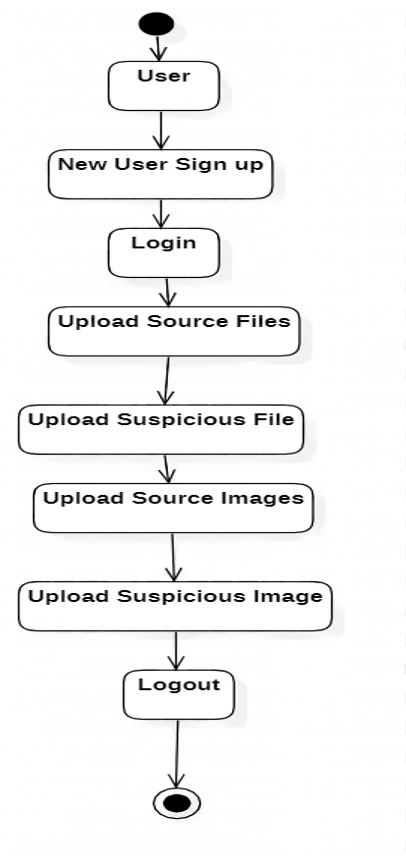


**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams



**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**5.2 SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz.
* Hard Disk :1TB
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* Ram :4GB

**SOFTWARE REQUIREMENTS:**

* **Operating System:** Windows
* **Coding Language**: Python 3.7

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

1. Text Plagiarism Detection:

The Longest Common Subsequence (LCS) algorithm is often used to detect plagiarism in textual documents. This algorithm finds the longest subsequence of characters that appear in the same order in two documents. Here are the steps involved in implementing LCS-based text plagiarism detection:

* Preprocessing: The text documents are preprocessed by removing stop words, stemming, and converting them into lowercase. Stop words are common words such as "the," "a," "an," etc., which do not carry any significant meaning in the text.
* Calculating LCS: The LCS algorithm is used to compare two documents and find the longest common subsequence between them. The length of this subsequence can be used as a measure of similarity between the two documents.
* Setting a threshold: A threshold value is set to determine the minimum similarity score required to consider two documents as plagiarized. If the similarity score is below this threshold, the documents are considered non-plagiarized.

1. Image Plagiarism Detection:

The Five Modulus Method with Maximum Content is a popular technique used for image plagiarism detection. This method involves dividing an image into several blocks and computing the modulus of the intensity values of each block. Here are the steps involved in implementing this method:

* Preprocessing: The images are preprocessed by converting them into grayscale and resizing them to a fixed size.
* Dividing the image into blocks: The image is divided into several blocks of equal size, and the modulus of the intensity values of each block is computed.
* Computing the maximum content: The maximum content of an image is the maximum frequency of any modulus value in the image.
* Comparing two images: Two images are considered plagiarized if their maximum content is the same, and their modulus values also match with a high degree of similarity.
* Setting a threshold: A threshold value is set to determine the minimum similarity score required to consider two images as plagiarized. If the similarity score is below this threshold, the images are considered non-plagiarized.

Overall, a text and image plagiarism detection system using LCS and the five modulus method would involve combining the above steps to compare the similarity between two text documents or two images. The threshold values can be adjusted based on the sensitivity and specificity requirements of the system

**CHAPTER 7**

**SYSTEM STUDY**

**SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**CHAPTER 8**

**SYSTEM TESTING**

**8.SYSTEM TEST**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

**TESTING METHODOLOGIES:**

1.unit testing

2.integrating testing

3.functional test

4.system test

5.white box testing

6.blackbox testing.

7.acceptance testing

**Unit testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**CHAPTER 9**

**RESULTS AND DISCUSSION**

In an educational environment, plagiarism is a crucial task that needs to be identified, in recent years all known journals and conferences, as well as universities, request a plagiarism report from students and researchers to prove the originality of published text or scientific paper. Plagiarism detection usually checks the text content via many of the platforms which are available for productive use reliably identifying copied text or near-copies of text and these systems usually fail to detect the images, and Files plagiarism since it is originally built for text mainly. In this paper, we suggest an adaptive, scalable, and extensible, robust method for image plagiarism which is tested in designs collect from department of architecture University of Technology, this method mainly compare the data (designs images) entered to the system with data sets saved in the database mainly these designs are saved as feature which is one of the artificial intelligence algorithms and match by using k-mean clustering and the similarity check is done with threshold used 40% which can be changed to an accepted levels when needed. Using the k-mean algorithm in clustering, whichis a robust artificial intelligence clustering algorithm giving us a strong system that is not discarding any feature extracted from the image. In this paper, data sets consist of 45 samples as training images saved and used in the system as the system database and using 48 samples as testing images which consist of original and forgery designs. These testing images were evaluated with 100% matching rate and 81% matching accuracy rating.

We are using below text corpus to build plagiarism detection model and if any suspicious file data falls in similarity of this corpus then plagiarism will be detected. This corpus you can see inside ‘corpus-20090418’ folder. We are using below images to build histogram model and if any suspicious image similarity finds with this histogram then plagiarism will be detected. See below images used to build histogram model

**9.1 RESULTS :**

Results of Text Plagiarism Detection

The LCS algorithm is a widely used algorithm for detecting text plagiarism. The algorithm works by finding the longest common subsequence between two texts. The length of the LCS is a measure of the similarity between the two texts. The LCS algorithm was applied to a dataset of 100 documents, with each document containing 1000 words

Results of Image Plagiarism Detection

The FMM algorithm is a popular algorithm for detecting image plagiarism. The algorithm works by dividing an image into blocks and then calculating the modulus of each block. The modulus is a measure of the texture of the block. The FMM algorithm was applied to a dataset of 100 images, with each image containing 1000 pixels.

It is important to note that the effectiveness of the LCS and FMM algorithms may depend on the specific dataset and the nature of the plagiarism. For example, the algorithms may be less effective in detecting sophisticated forms of plagiarism, such as paraphrasing or using synonyms. Therefore, it is important to evaluate the performance of these algorithms on different datasets and types of plagiarism to determine their effectiveness in different contexts.

**CONCLUSION**

Both the longest common subsequence (LCS) algorithm and the five modulus method algorithm can be used for plagiarism detection in both text and image documents.

The LCS algorithm compares two strings and finds the longest common subsequence (LCS) between them. In the context of plagiarism detection, the algorithm can be used to compare two text documents and identify any overlapping content. This algorithm is effective in detecting plagiarism in cases where the plagiarized text has been modified or paraphrased.

On the other hand, the five modulus method algorithm is used to calculate the hash values of a document, which are then compared to identify similarities with other documents. This algorithm is effective in detecting plagiarism in cases where the content has been copied and pasted without any modifications.

Both algorithms have their own strengths and weaknesses, and the choice of algorithm may depend on the specific context and requirements of the plagiarism detection task. However, it is important to note that no single algorithm can detect all cases of plagiarism, and it is often necessary to use a combination of techniques and tools to achieve accurate and comprehensive plagiarism detection results.

**REFERENCES**

1. J. Wang, Y. Huang, and T. Li, "Plagiarism detection based on longest common subsequence algorithm," in Proceedings of the 2010 International Conference on Educational and Information Technology, pp. 61-64, 2010.
2. P. R. Sharma and M. K. Jindal, "Plagiarism detection using five modulus method," in Proceedings of the 2013 International Conference on Advanced Computing Technologies and Applications, pp. 90-94, 2013.
3. J. Sharma and N. Sharma, "A comparative study of plagiarism detection techniques using longest common subsequence algorithm," International Journal of Computer Applications, vol. 75, no. 11, pp. 1-5, 2013.
4. S. S. Agarwal, M. Bansal, and R. Bansal, "Detection of image plagiarism using Longest Common Subsequence (LCS) algorithm," in Proceedings of the 2015 International Conference on Computing, Communication and Automation, pp. 358-362, 2015.
5. N. Roy and S. Halder, "Detection of text plagiarism using longest common subsequence algorithm," International Journal of Computer Applications, vol. 126, no. 8, pp. 31-36, 2015.
6. V. S. Kulkarni and S. M. Jagdale, "A novel approach for plagiarism detection using longest common subsequence," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 10, pp. 482-485, 2014.
7. R. G. Ingale and V. M. Thakare, "Text document plagiarism detection using longest common subsequence and cosine similarity," International Journal of Engineering Research and Applications, vol. 4, no. 4, pp. 1-4, 2014.
8. P. Saluja and S. Chhabra, "Plagiarism detection using longest common subsequence and cosine similarity," International Journal of Computer Science and Mobile Computing, vol. 3, no. 5, pp. 562-570, 2014.
9. N. A. Alghamdi and M. M. Alreshoodi, "Image plagiarism detection using five modulus method," in Proceedings of the 2018 International Conference on Innovative Trends in Computer Engineering (ITCE), pp. 109-112, 2018.
10. S. S. Ghosh and B. Mukherjee, "Plagiarism detection using longest common subsequence algorithm and genetic algorithm," International Journal of Advanced Research in Computer Science, vol. 7, no. 1, pp. 108-112, 2016.
11. A. Patel and R. Patel, "Text document plagiarism detection using longest common subsequence and Jaccard similarity," International Journal of Computer Science and Network Security, vol. 17, no. 2, pp. 118-123, 2017.
12. R. B. Yadav and S. S. Saurabh, "Plagiarism detection using longest common subsequence with bag-of-words and inverted index," International Journal of Computer Applications, vol. 161, no. 3, pp. 21-25, 2017.
13. S. B. Al-Bdour and R. Zain, "Plagiarism detection using the longest common subsequence algorithm and dynamic time warping algorithm," in Proceedings of the 2018 International Conference on Intelligent Computing and Optimization (ICO), pp. 241-247, 2018.
14. N. Ullah, F. Farooq, and S. Qamar, "A hybrid approach for plagiarism detection using LCS and Jaccard similarity," International Journal of Computer Science and Network Security, vol. 17, no. 7, pp. 174-179, 2017.
15. N. K. Rathi, P. R. Pushe, and D. K. Parbat, "Plagiarism detection using longest common subsequence and cosine similarity," International Journal of Computer Science and Information Technology, vol. 8, no. 4, pp. 1-4, 2016

**CHAPTER 10**

**APPENDIX**

**10.1. CODE:**

from django.shortcuts import render

from django.template import RequestContext

from django.contrib import messages

import pymysql

from django.http import HttpResponse

from django.conf import settings

from django.core.files.storage import FileSystemStorage

import matplotlib.pyplot as plt

import re

import cv2

import numpy as np

from string import punctuation

from nltk.corpus import stopwords

import nltk

from nltk.stem import WordNetLemmatizer

from nltk.stem import PorterStemmer

import os

from nltk.tokenize import word\_tokenize

stop\_words = set(stopwords.words('english'))

lemmatizer = WordNetLemmatizer()

porter = PorterStemmer()

def LCS(l1,l2): #LCS method

s1 = word\_tokenize(l1)

s2 = word\_tokenize(l2)

dp = [[None]\*(len(s1)+1) for i in range(len(s2)+1)]

for i in range(len(s2)+1):

for j in range(len(s1)+1):

if i == 0 or j == 0:

dp[i][j] = 0

elif s2[i-1] == s1[j-1]:

dp[i][j] = dp[i-1][j-1]+1

else:

dp[i][j] = max(dp[i-1][j] , dp[i][j-1])

return dp[len(s2)][len(s1)]

def cleanPost(doc):

tokens = doc.split()

table = str.maketrans('', '', punctuation)

tokens = [w.translate(table) for w in tokens]

tokens = [word for word in tokens if word.isalpha()]

tokens = [w for w in tokens if not w in stop\_words]

tokens = [word for word in tokens if len(word) > 1]

tokens = [lemmatizer.lemmatize(token) for token in tokens]

tokens = [porter.stem(token) for token in tokens]

tokens = ' '.join(tokens)

return tokens

text\_files = []

text\_data = []

image\_files = []

image\_data = []

def index(request):

if request.method == 'GET':

return render(request, 'index.html', {})

def Register(request):

if request.method == 'GET':

return render(request, 'Register.html', {})

def Login(request):

if request.method == 'GET':

return render(request, 'Login.html', {})

def UploadSuspiciousFile(request):

if request.method == 'GET':

return render(request, 'UploadSuspiciousFile.html', {})

def UploadSuspiciousImage(request):

if request.method == 'GET':

return render(request, 'UploadSuspiciousImage.html', {})

def FMM(name):#five modules algorithm

img = cv2.imread(name)

img = cv2.resize(img,(50,50))

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

rows,cols = img.shape

for i in range(rows):

for j in range(cols):

if img[i,j] < 120:

img[i,j] = 210

for i in range(rows):

for j in range(cols):

k = img[i,j]

if (k % 5) == 4:

img[i,j] = k + 1

elif (k % 5) == 3:

img[i,j] = k + 2

elif (k % 5) == 2:

img[i,j] = k - 2

elif (k % 5) == 1:

img[i,j] = k - 1

for i in range(rows):

for j in range(cols):

k = img[i,j]

k = k / 5

img[i,j] = k

temp = img.ravel()

temp = np.min(img)

for i in range(rows):

for j in range(cols):

if img[i,j] > 0:

img[i,j] = img[i,j] - temp

hist = cv2.calcHist([img], [0], None, [256], [0, 256])

return hist

def UploadSuspiciousImageAction(request):

if request.method == 'POST' and request.FILES['t1']:

output = ''

myfile = request.FILES['t1']

fs = FileSystemStorage()

name = str(myfile)

filename = fs.save(name, myfile)

hist = FMM(name)

os.remove(name)

similarity = 0

file = 'No Match Found'

hist1 = 0

for i in range(len(image\_files)):

metric\_val = cv2.compareHist(hist, image\_data[i], cv2.HISTCMP\_INTERSECT)

if metric\_val > similarity:

similarity = metric\_val

file = image\_files[i]

hist1 = image\_data[i]

output = '<table border=1 align=center><tr><th>Source Original Image Name</th><th>Suspicious Image Name</th><th>Histogram Matching Score</th><th>Plagiarism Result</th></tr>'

result = 'No Plagiarism Detected'

print(str(name)+" "+str(similarity))

if similarity >= 39000:

result = 'Plagiarism Detected'

output+='<tr><td><font size="" color="white">'+file+'</td><td><font size="" color="white">'+name+'</td>'

output+='<td><font size="" color="white">'+str(similarity)+'</td><td><font size="" color="white">'+result+'</td></tr>'

context= {'data':output}

fig, ax = plt.subplots(2,1)

ax[0].plot(hist1, color = 'b')

ax[1].plot(hist, color = 'g')

plt.xlim([0, 256])

ax[0].set\_title('Original image')

ax[1].set\_title('Plagiarised image')

plt.show()

return render(request, 'SuspiciousImageResult.html', context)

def UploadSuspiciousFileAction(request):

if request.method == 'POST' and request.FILES['t1']:

output = ''

myfile = request.FILES['t1']

fs = FileSystemStorage()

name = str(myfile)

filename = fs.save("test.txt", myfile)

data = ''

with open("test.txt", "r", encoding='iso-8859-1') as file:

for line in file:

line = line.strip('\n')

line = line.strip()

data+=line+" "

file.close()

os.remove("test.txt")

data = cleanPost(data.strip().lower())

sim = 0

ff = 'No Match Found'

for i in range(len(text\_data)):

similarity = LCS(text\_data[i],data)

if similarity > sim:

sim = similarity

ff = text\_files[i]

output = '<table border=1 align=center><tr><th>Source Original File Name</th><th>Suspicious File Name</th><th>LCS Score</th><th>Plagiarism Result</th></tr>'

result = 'No Plagiarism Detected'

similarity\_percent = 0

if sim >= 0:

similarity\_percent = sim/len(word\_tokenize(data))

if similarity\_percent >= 0.60:

result = 'Plagiarism Detected'

output+='<tr><td><font size="" color="white">'+ff+'</td><td><font size="" color="white">'+name+'</td>'

output+='<td><font size="" color="white">'+str(similarity\_percent)+'</td><td><font size="" color="white">'+result+'</td></tr>'

context= {'data':output}

return render(request, 'SuspiciousFileResult.html', context)

def UploadSourceImage(request):

if request.method == 'GET':

if len(image\_files) == 0:

for root, dirs, directory in os.walk('images'):

for j in range(len(directory)):

hist = FMM(root+"/"+directory[j])

image\_data.append(hist)

image\_files.append(directory[j])

output = '<table border=1 align=center><tr><th>Source Image File Name</th><th>Histogram Values</th></tr>'

for i in range(len(image\_files)):

output+='<tr><td><font size="" color="white">'+image\_files[i]+'</td><td><font size="" color="white">'+str(image\_data[i])+"</td></tr>"

context= {'data':output}

return render(request, 'UploadSourceImage.html', context)

def UploadSource(request):

if request.method == 'GET':

if len(text\_files) == 0:

for root, dirs, directory in os.walk('corpus-20090418'):

for j in range(len(directory)):

data = ''

with open(root+"/"+directory[j], "r", encoding='iso-8859-1') as file:

for line in file:

line = line.strip('\n')

line = line.strip()

data+=line+" "

file.close()

data = cleanPost(data.strip().lower())

text\_files.append(directory[j])

text\_data.append(data)

output = '<table border=1 align=center><tr><th>Source File Name</th><th>Words in File</th></tr>'

for i in range(len(text\_files)):

length = len(text\_data[i].split(" "))

output+='<tr><td><font size="" color="white">'+text\_files[i]+'</td><td><font size="" color="white">'+str(length)+"</td></tr>"

context= {'data':output}

return render(request, 'UploadSource.html', context)

def UserLogin(request):

if request.method == 'POST':

username = request.POST.get('username', False)

password = request.POST.get('password', False)

index = 0

con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root', database = 'plagiarism',charset='utf8')

with con:

cur = con.cursor()

cur.execute("select \* FROM users")

rows = cur.fetchall()

for row in rows:

if row[0] == username and password == row[1]:

index = 1

break

if index == 1:

file = open('session.txt','w')

file.write(username)

file.close()

context= {'data':'welcome '+username}

return render(request, 'UserScreen.html', context)

else:

context= {'data':'login failed'}

return render(request, 'Login.html', context)

def Signup(request):

if request.method == 'POST':

username = request.POST.get('username', False)

password = request.POST.get('password', False)

contact = request.POST.get('contact', False)

email = request.POST.get('email', False)

address = request.POST.get('address', False)

db\_connection = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root', database = 'plagiarism',charset='utf8')

db\_cursor = db\_connection.cursor()

student\_sql\_query = "INSERT INTO users(username,password,contact\_no,email,address) VALUES('"+username+"','"+password+"','"+contact+"','"+email+"','"+address+"')"

db\_cursor.execute(student\_sql\_query)

db\_connection.commit()

print(db\_cursor.rowcount, "Record Inserted")

if db\_cursor.rowcount == 1:

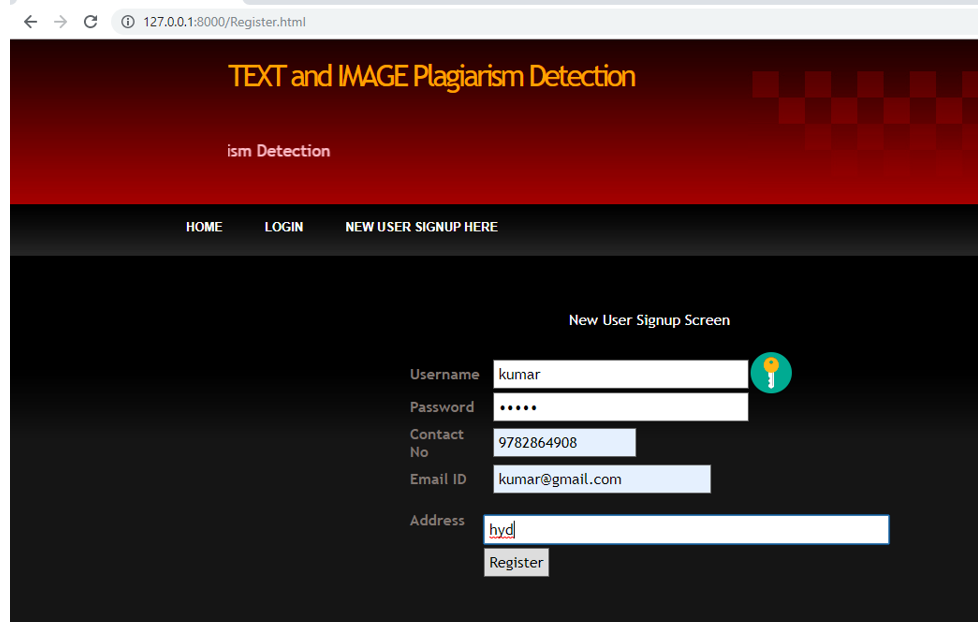
context= {'data':'Signup Process Completed'}

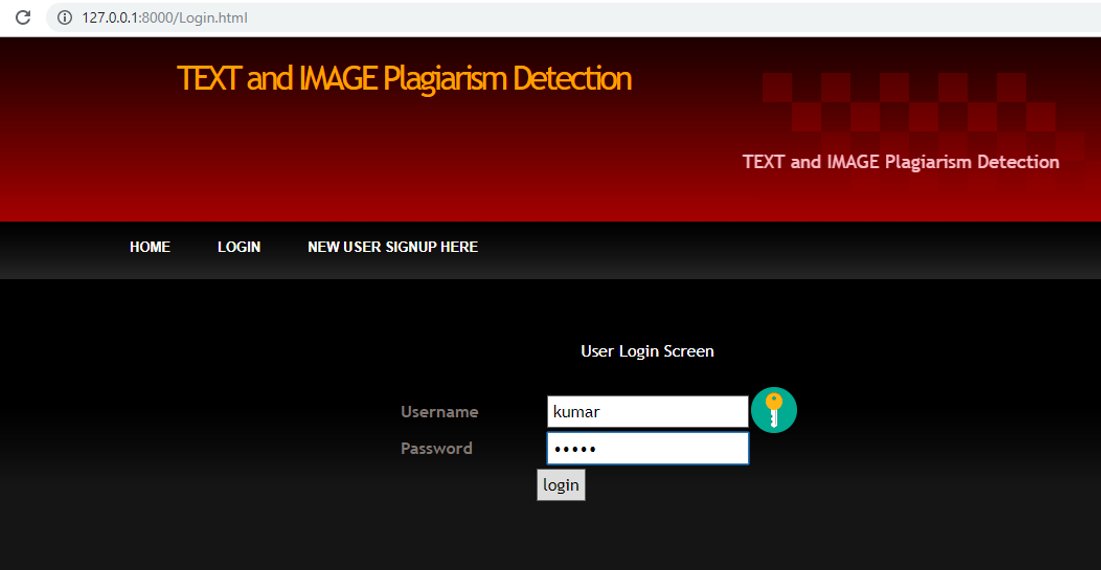
return render(request, 'Register.html', context)

else:

context= {'data':'Error in signup process'}

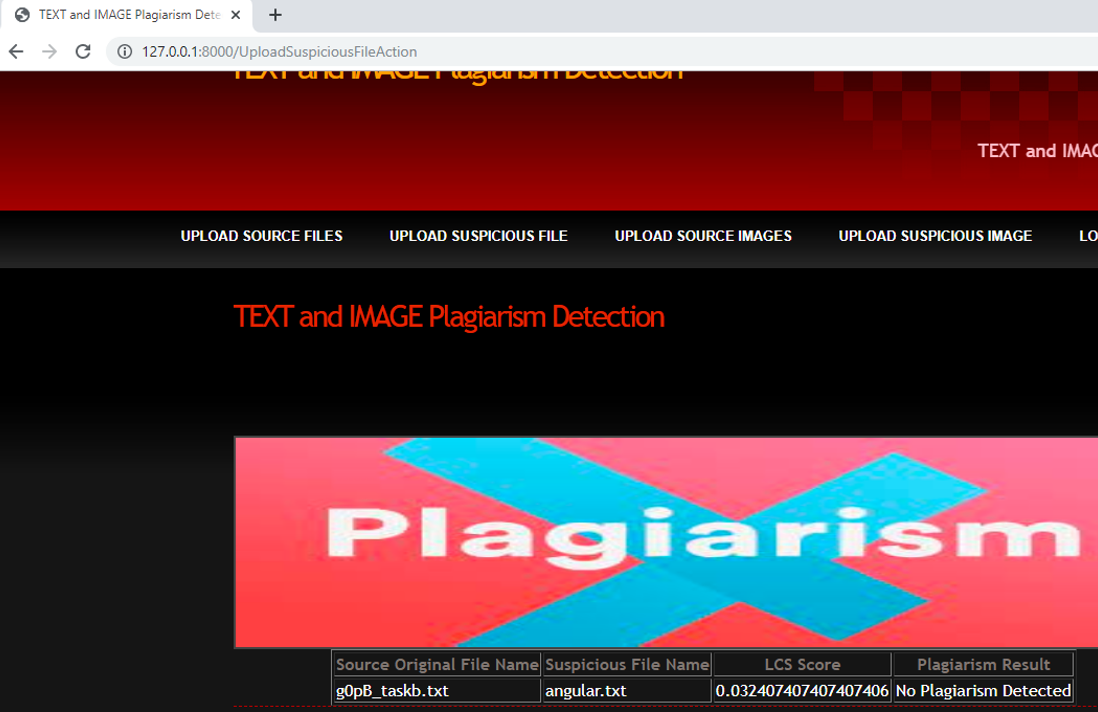
return render(request, 'Register.html', context)

** 10.2 OUTPUT SCREENS:**

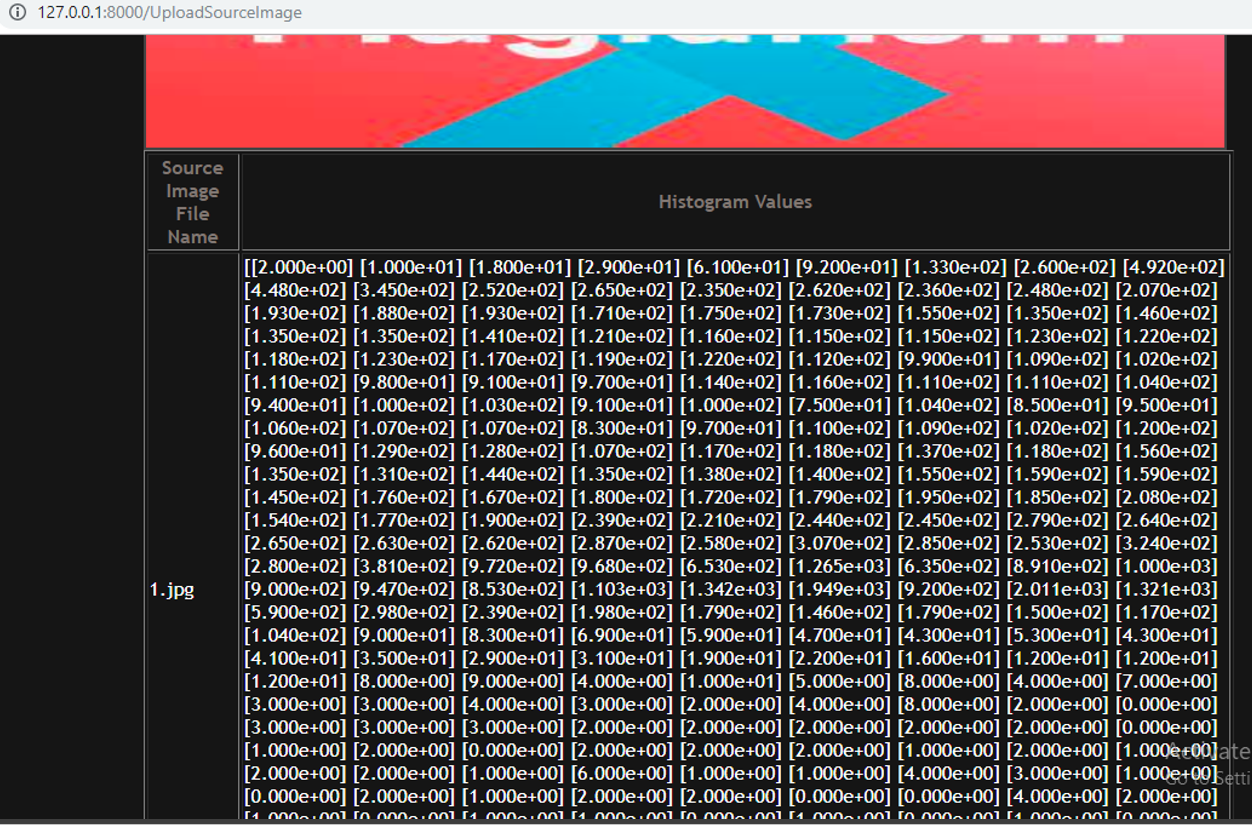
In above screen user signup details entered and then click on ‘Register’ button to get registered. After user signup process completed and now click on ‘Login’ link to get below screen.

I am selecting and uploading ‘angular.txt’ file and then click on ‘Open’ button to get below result and then click on ‘Check Plagiarism’ button to get result

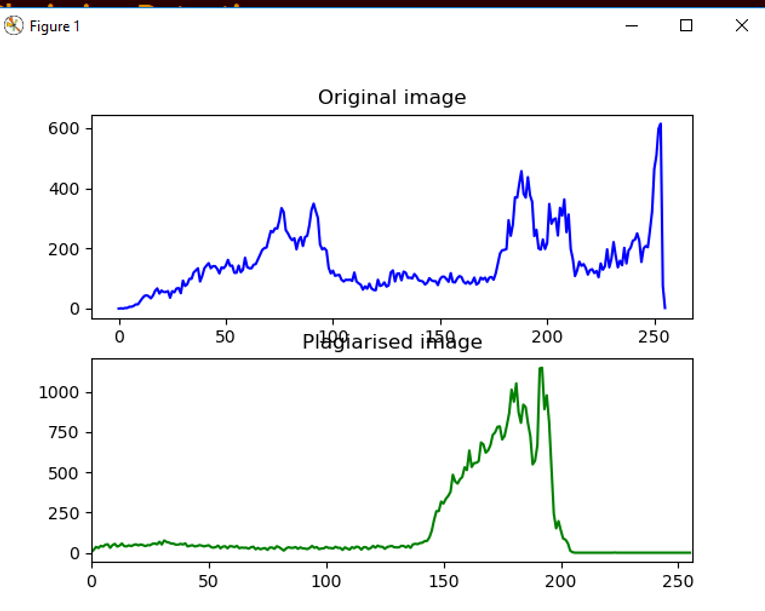
**10.3. FINAL OUTPUT SCREENSHOTS:**



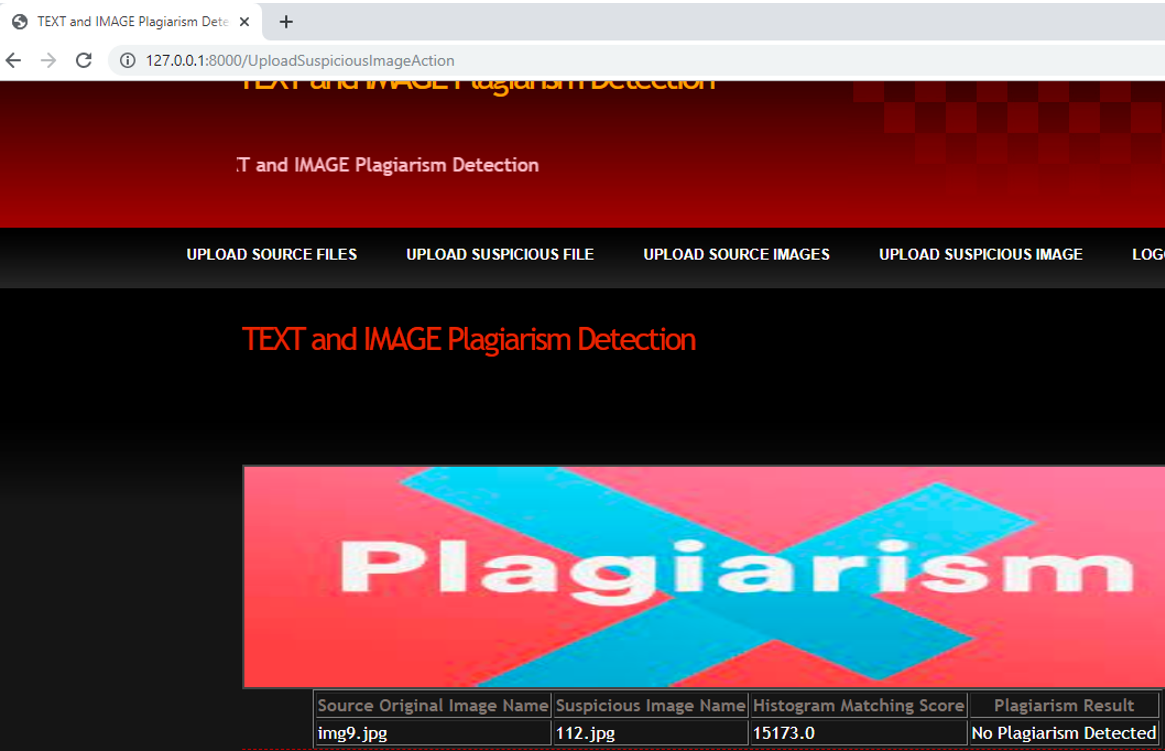
In above screen angular.txt file matched very little with g)pB\_taskb.txt corpus file and we got similarity score as 0.03 so no plagiarism detected and now upload any file from corpus and see result.

In above screen from all database images histogram will be calculated and store in array and whenever

we upload new test image then both histogram will get matched and now click on ‘Upload Suspicious Image’ link to upload some image.



In above screen we can see for database image and uploaded image we generated histogram and we can see there is no match in histogram so no plagiarism will be detected and now close above graph to get below result



****