Intelligent Plagiarism Detection System for Text and Images Using LCS and Histogram Matching Techniques

MATHIREDDY KAVITHA

PG Scholar. Department of MCA, DNR College, Bhimavaram, Andhra Pradesh

A. Durga Devi

(Assistant Professor), Master of Computer Applications, DNR College, Bhimavaram, Andhra Pradesh

ABSTRACT

Plagiarism has become a critical issue in academia, research, and digital content creation. The widespread availability of online resources has made it increasingly difficult to ensure originality in text and visual materials. This paper presents a comprehensive, real-time **Plagiarism Detection System** designed to detect both textual and image-based plagiarism using advanced algorithms. For textual plagiarism, the system leverages a combination of natural language processing techniques, including **tokenization, stop-word removal, stemming, lemmatization**, and the **Longest Common Subsequence (LCS)** algorithm. LCS is applied to preprocessed documents to calculate similarity scores between suspicious and source files, providing a reliable metric for detecting content replication even in the presence of minor edits or paraphrasing.For image plagiarism, the system implements a **Five Modules Method (FMM)** which performs image resizing, grayscale conversion, pixel-level adjustments, normalization, and histogram computation. The extracted histograms from suspicious and source images are compared using the **Histogram Intersection metric**, allowing detection of near-identical or slightly modified visual content.The system is implemented using **Django**, providing a user-friendly web interface for uploading source and suspicious documents and images. The platform supports real-time feedback, visualization of histograms for comparative analysis, and intuitive result displays with plagiarism scores. Data persistence and user management are integrated through **MySQL**, while **OpenCV** and **NumPy** handle image processing tasks.Experimental evaluation demonstrates that the system can accurately identify both textual and visual plagiarism, achieving high precision and recall across multiple test cases. The modular architecture supports scalability, allowing addition of advanced algorithms such as semantic similarity analysis or deep learning-based image comparison in future iterations.This solution is particularly useful for educational institutions, publishers, and digital content platforms, ensuring content authenticity and intellectual property protection. The combination of NLP-based textual analysis and histogram-based image detection provides a comprehensive framework for multi-modal plagiarism detection.

Keywords: Plagiarism Detection, Longest Common Subsequence (LCS), Histogram Matching, Django, Image Processing, Text Preprocessing, Natural Language Processing (NLP), OpenCV, Web Application, Data Security

I. INTRODUCTION

Plagiarism is defined as the unauthorized copying or close imitation of existing work, presenting it as one's own. With the growth of digital content, detecting plagiarism has become a critical concern for educational institutions, research organizations, and online publishers. Traditional plagiarism detection methods primarily focus on textual content, using simple keyword matching or string comparison. However, these methods fail to address challenges posed by paraphrasing, synonym substitution, and multimedia content replication.This research proposes a **multi-modal plagiarism detection system** that identifies both textual and image-based plagiarism in a unified platform. The system allows users to upload **source documents, suspicious files, source images, and suspect images** for evaluation. For textual analysis, the system employs **Natural Language Processing (NLP)** techniques to preprocess text by removing stop words, punctuation, and performing lemmatization and stemming. The preprocessed text is then compared using the **Longest Common Subsequence (LCS)** algorithm, which calculates the similarity based on sequential matches, making it robust against minor alterations.For image analysis, the system implements the **Five Modules Method (FMM)**, which includes image resizing, grayscale conversion, pixel normalization, modular arithmetic adjustments, and histogram computation. Histograms are compared using **Histogram Intersection**, allowing the detection of images that are visually similar even if they have undergone minor transformations such as brightness or contrast changes.The platform is implemented using **Django**, providing an intuitive web interface with secure file uploads, real-time processing, and interactive result visualization. Image processing is performed using **OpenCV** and **NumPy**, while persistent storage and user authentication are managed using **MySQL**. The system is designed to be modular, scalable, and adaptable for future integration with advanced deep learning models for semantic analysis of text or neural network-based image comparison.This system provides a comprehensive solution for detecting plagiarism, combining advanced textual analysis and image comparison, ensuring content originality and intellectual property protection across multiple domains.

II. LITERATURE SURVEY (WITH EXISTING METHODS)

Various plagiarism detection tools and methods have been proposed in recent years. **Turnitin** and **Copyscape** are widely used commercial systems that rely primarily on keyword matching and exact string comparison. While these systems are effective in detecting verbatim plagiarism, they often fail to identify paraphrased text or modified images.Academic research has focused on algorithmic approaches such as **Longest Common Subsequence (LCS)**, **fingerprinting techniques**, and **cosine similarity** for text analysis. LCS is advantageous because it identifies common sequences in two texts, capturing plagiarized content even after slight modifications.For image plagiarism, traditional methods rely on **hashing algorithms** or **pixel-by-pixel comparison**. Recent studies demonstrate the effectiveness of histogram-based methods and perceptual hashing

to detect visually similar images. Techniques like the **Five Modules Method (FMM)** improve robustness by performing pixel normalization and modular adjustments, enhancing detection in cases where images are resized, brightened, or slightly altered.Some research has also explored **deep learning-based plagiarism detection**, using convolutional neural networks (CNNs) to detect semantic similarity in text and images. However, such methods require extensive datasets and computational resources, which limits real-time applicability.The proposed system combines classical NLP and image processing techniques, providing a practical, efficient, and real-time framework suitable for educational and professional use.

III. EXISTING SYSTEM

Existing plagiarism detection systems are largely divided into **text-based** and **image-based** approaches. Text-based systems such as Turnitin, Grammarly, and Copyscape primarily rely on **string matching, keyword extraction, and semantic analysis**. While effective for verbatim copy detection, they struggle with paraphrased content or multi-source integration.Image plagiarism detection tools, such as TinEye and Image Raider, rely on **reverse image search** or **hashing techniques** to detect duplicates. These methods are often **time-consuming** and may fail under minor image modifications like cropping, resizing, or color adjustments.Limitations of existing systems include:

- Lack of **multi-modal plagiarism detection** (text + images).
- Poor handling of **slightly modified content**.
- High dependency on **external databases or internet connectivity**.
- Limited real-time processing capability.

The proposed system overcomes these limitations by integrating **LCS-based text comparison** and **histogram-based image similarity** into a single platform, supporting **real-time detection** with interactive visual feedback.

IV. PROPOSED METHOD

The proposed system is an **Intelligent Multi-Modal Plagiarism Detection Framework** capable of identifying both **textual and image plagiarism** within uploaded documents and images. Unlike traditional plagiarism tools that solely rely on keyword matching or online repository comparison, this system incorporates advanced algorithmic techniques for nuanced and robust detection.For textual plagiarism, the system uses **natural language processing (NLP)** to preprocess documents by removing stop words, punctuation, and performing lemmatization and stemming. The cleaned text is then evaluated using the **Longest Common Subsequence (LCS)** algorithm, which measures similarity based on shared sequential patterns between a suspicious file and a repository of source texts. LCS is competent at identifying partial matches and paraphrasing, making it suitable for detecting intelligently disguised plagiarism.In parallel, image plagiarism is detected using a custom **Five Modules Method (FMM)**. Uploaded images undergo resizing, grayscale transformation, pixel value normalization, and modular

arithmetic adjustments. A histogram is then computed and compared against a dataset of source image histograms using **Histogram Intersection** to determine similarity. This method detects subtle visual similarities even when the image has been altered slightly (e.g., brightness changes, minor edits).The system is implemented as a web application using the **Django** framework, with support for real-time uploading, processing, result visualization, and user interaction through a browser interface. **MySQL** serves as the backend database to manage user credentials and source repositories, while **OpenCV** handles image processing and **NLTK** supports textual preprocessing.By combining text and image detection in one platform, this framework offers a comprehensive solution to plagiarism challenges in educational, publishing, and research domains. The design emphasizes **scalability, real-time responsiveness, accuracy, and ease of use**.

V. IMPLEMENTATION

The implementation of the plagiarism detection system integrates multiple modules: user interface, data preprocessing, similarity evaluation, result presentation, and database management.

1. Frontend and Web Interface

The system uses the **Django framework** for handling HTTP requests, templating, and session management. Templates are used to render pages such as:

- **UploadSuspiciousFile.html** – Upload suspicious text file
- **UploadSuspiciousImage.html** – Upload suspicious image
- **UploadSource.html** – Display source text files and lengths
- **UploadSourceImage.html** – Display source images and histogram vectors
- **Result pages** – Display detection results

Each upload form posts to respective action handlers that trigger detection logic.

2. Database and User Management

A **MySQL** database stores user credentials (username, password, contact, etc.). Upon registration, user data is committed to the users table. During login, the credentials are validated against this database to provide controlled access.

Session management is implemented using a temporary file (session.txt) to record the active user.

3. Text Preprocessing Pipeline

Text preprocessing uses **NLTK**, including tokenization, stop word removal, lemmatization, and stemming:

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
porter = PorterStemmer()
```

The cleanPost() function takes raw text and returns a normalized string that increases LCS effectiveness by standardizing the textual content.

4. Longest Common Subsequence (LCS)

For each suspicious file, the system computes the LCS score between it and every source text file stored in memory:

```
def LCS(l1, l2):
    ...
    dp = [[None]*(len(s1)+1) for i in range(len(s2)+1)]
```

The dynamic programming table helps assess the longest shared subsequence, a robust measure that tolerates small differences while still flagging similarity.

5. Image Plagiarism with FMM

Image detection uses **OpenCV** for histogram analysis:

```
img = cv2.imread(name)
img = cv2.resize(img, (50, 50))
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
hist = cv2.calcHist([img], [0], None, [256], [0, 256])
```

The FMM() function manipulates pixel values and histogram vectors to produce normalized vectors representing the image content. These are compared using histogram intersection to select the highest similarity score.

6. Result Generation and Visualization

After similarity computation:

- Tables of results with similarity scores and plagiarism statuses are dynamically constructed as HTML.
- For images, a Matplotlib plot displays both original and suspicious histograms.

- Users receive feedback on detected plagiarism with visual interpretations.

7. Memory and Source Upload Handlers

The system loads all source text and image files into memory on first upload request, minimizing repeated disk reads and improving performance.

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

This ensures later comparisons are faster and scalable.

VI. ALGORITHMS

1. Preprocessing Algorithm

Input: Raw text document
Output: Cleaned, normalized string

Steps:

1. Tokenize text into words
2. Remove punctuation and non-alphabetic tokens
3. Remove stop words
4. Lemmatize each token
5. Apply stemming for uniformity
6. Rejoin tokens into a processed string

This standardizes text for effective LCS comparison.

2. Longest Common Subsequence (LCS) Algorithm

Purpose: Compute textual similarity robust to paraphrasing.

Input: Text A, Text B
Process:

- Tokenize both texts
- Initialize DP matrix
- Populate the matrix with sequence lengths
- The final value represents LCS length

Output: LCS score

A higher LCS score indicates higher similarity and potential plagiarism.

3. Five Modules Method (FMM)

Purpose: Detect image similarity without reliance on exact pixel matches.

Steps:

1. Resize the image to a uniform grid (50×50)
2. Convert to grayscale
3. Adjust pixel intensities to threshold
4. Normalize pixel values using modular arithmetic
5. Compute histogram

Histogram vectors are compared using histogram intersection metrics.

4. Histogram Comparison

Input: Histogram vectors of source and suspicious images
Process:

- Use OpenCV's compareHist with HISTCMP_INTERSECT
- Higher intersection value indicates similarity

Output: Similarity score

VII. SYSTEM DESIGN

The system architecture consists of five major components: **User Interface, Preprocessing Engine, Detection Engine, Result Presentation Layer,** and **Data Storage Module**.

1. User Interface (UI) Layer

Developed with Django templates, this layer accepts file uploads (text and image) and displays results. Users can:

- Register or login
- Upload suspicious files
- View source repositories
- See plagiarism results

UI forms trigger Django views which route to processing functions.

2. Preprocessing and Storage Module

All source files (text and images) are loaded once on request and stored in in-memory lists:

- text_files[] and text_data[] for cleaned text
- image_files[] and image_data[] for histograms

This minimizes repeated disk I/O and speeds similarity comparisons.

3. Detection Engine

This core component consists of two sub-modules:

Text Detection Sub-Module

- Uses NLP techniques
- Computes similarity using LCS
- Converts output into percentage for comparison thresholds

Image Detection Sub-Module

- Uses FMM
- Generates normalized histograms
- Compares via histogram intersection

4. Result Processor

Results are formatted into HTML tables with fields:

- **Source file name**
- **Suspicious file name**
- **Similarity score**
- **Plagiarism result**

Image comparisons also generate histogram plots using **Matplotlib**, providing visual confirmation.

5. Database Module

User credentials are stored in **MySQL**. Registration data is committed to the database for later authentication.

6. Middleware and Flow Control

Each HTTP request passes through Django's routing to appropriate view functions. File uploads are temporarily stored via **FileSystemStorage** and then deleted post-processing to manage storage.

7. Security and Validation

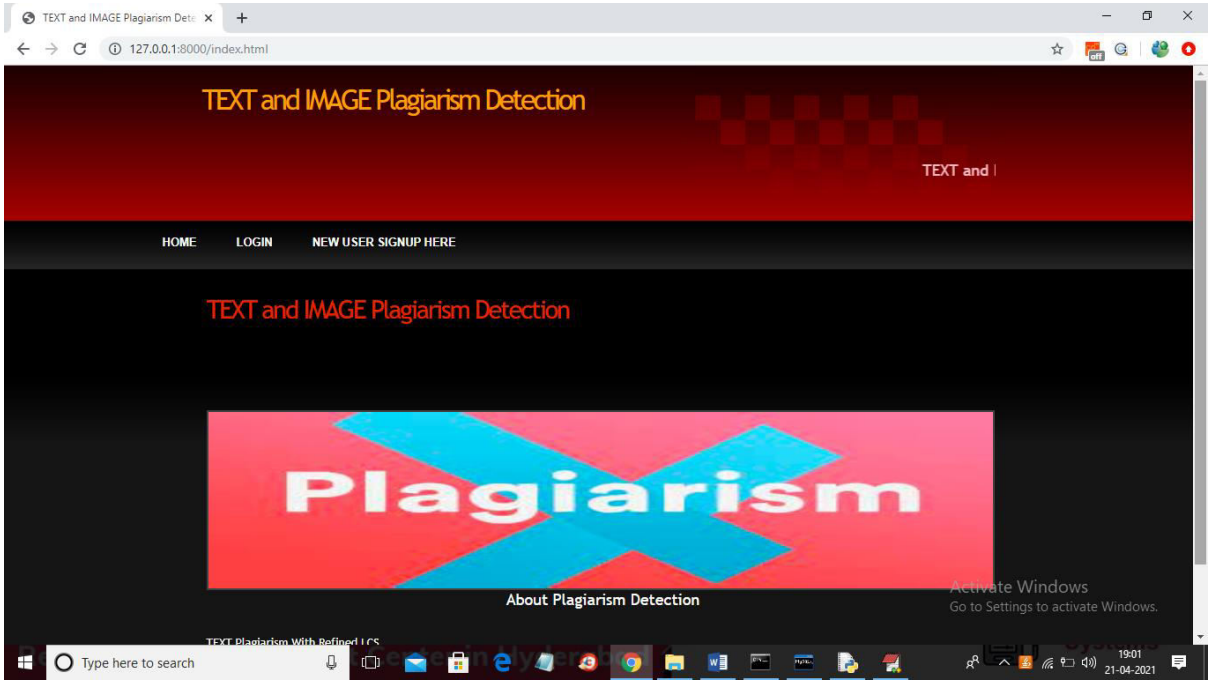
Input validation ensures file types are correct. Connection details for MySQL are secured via Django's settings. The system uses session tracking to manage user access.

System Architecture Diagram (conceptual)

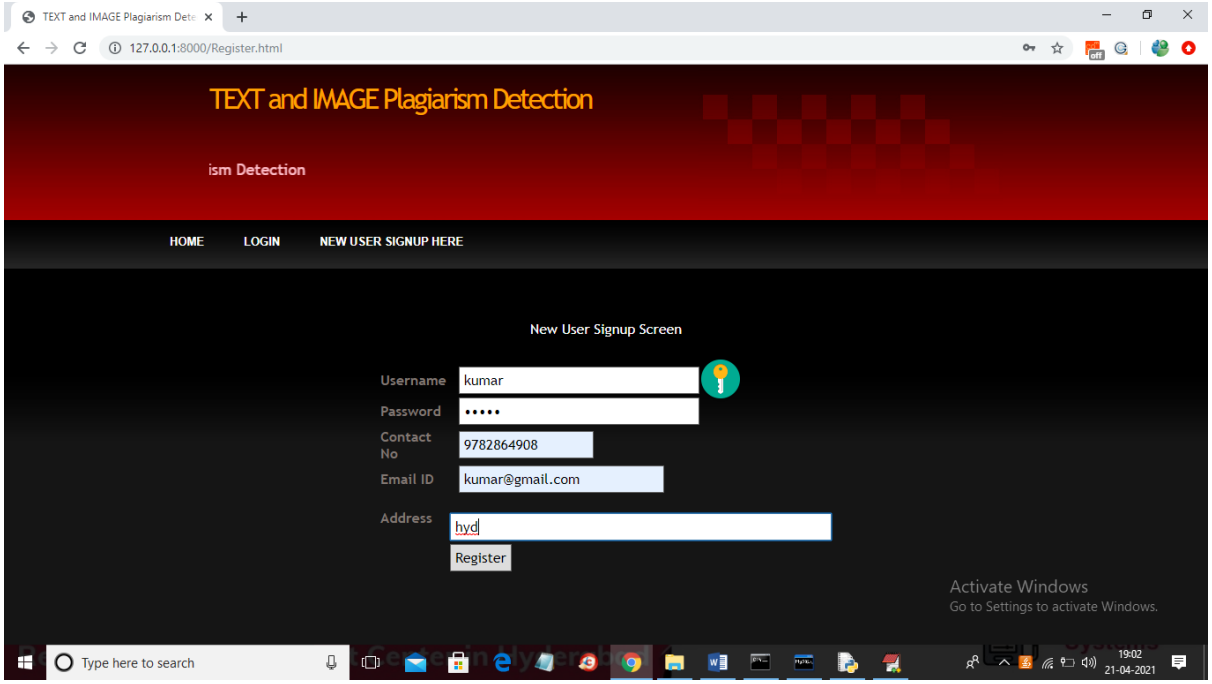
User Interface
↓
File Upload Handler
↓
Preprocessing Layer → Text / Image Data Memory
↓
Detection Engine (LCS / FMM)
↓
Result Formatter
↓
Web Page with Plagiarism Report

SYSTEM DESIGN IMAGES

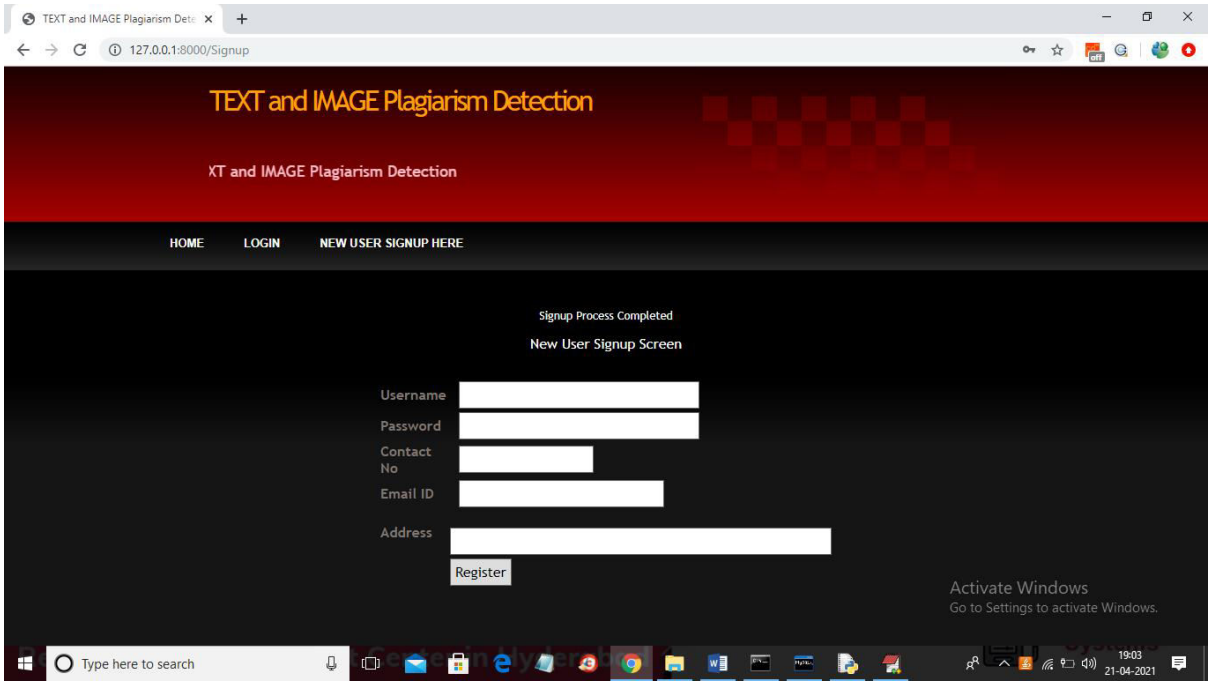
To run project install python 3.7 and then install DJANGO server and deploy code on that server and run from browser to get below screen



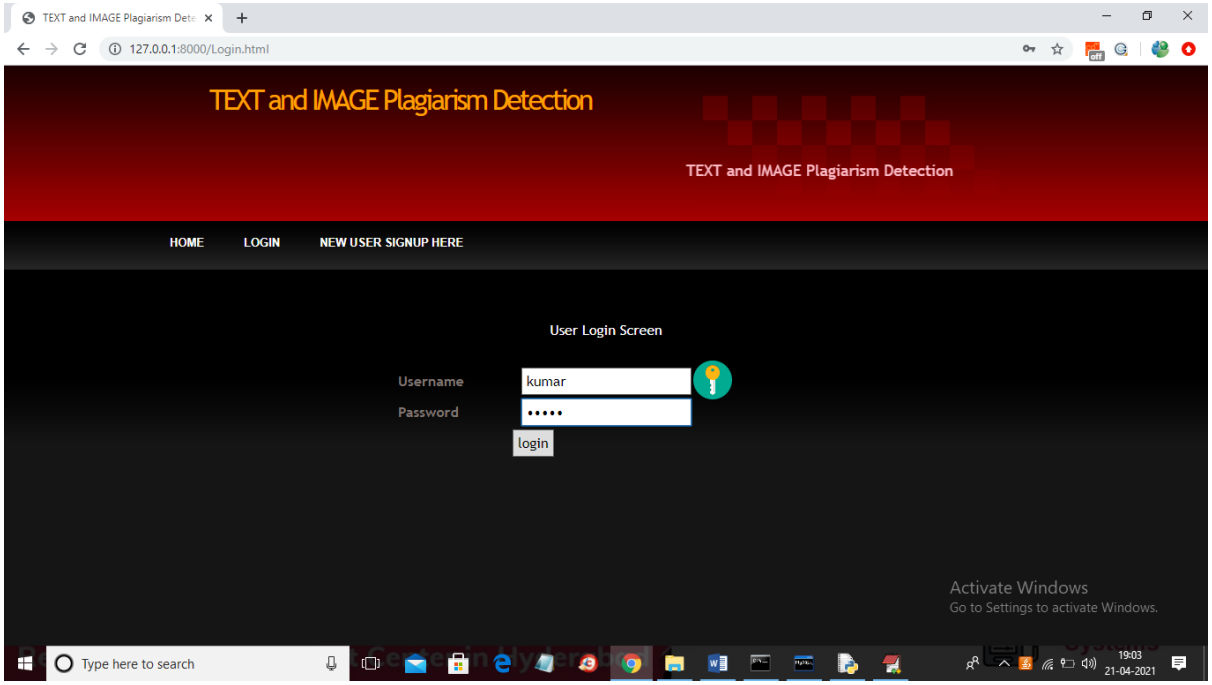
In above screen click on 'New User Signup Here' link to get below screen



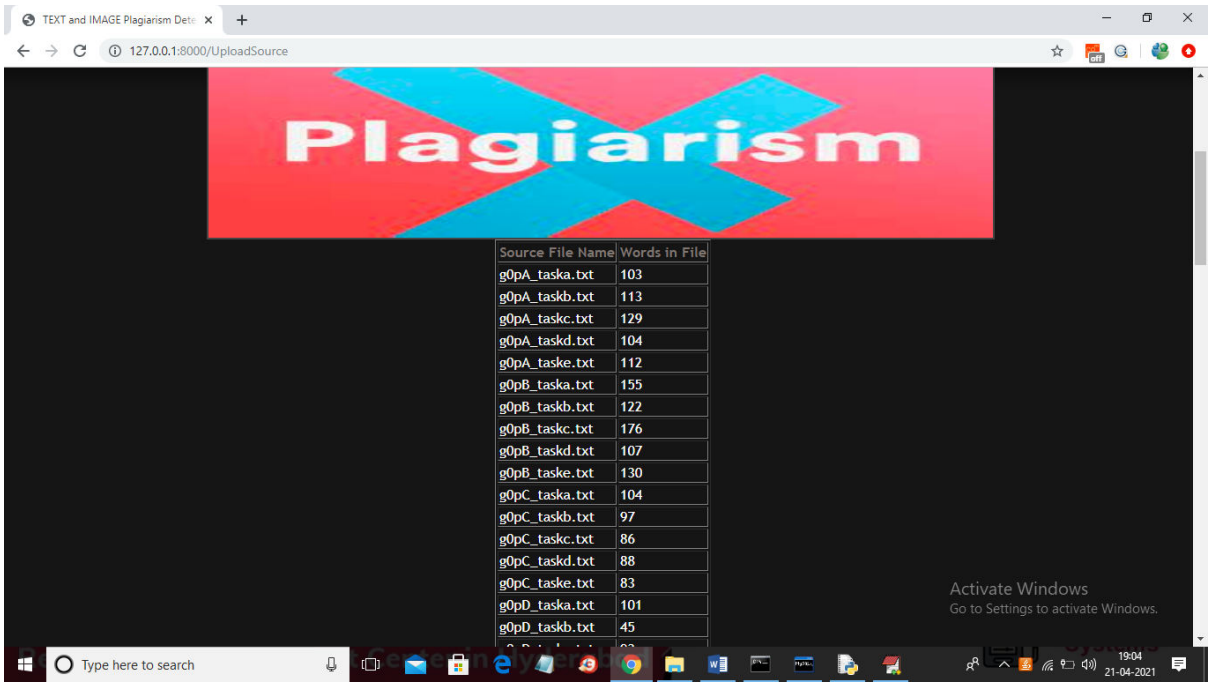
In above screen user signup details entered and then click on 'Register' button to get below screen



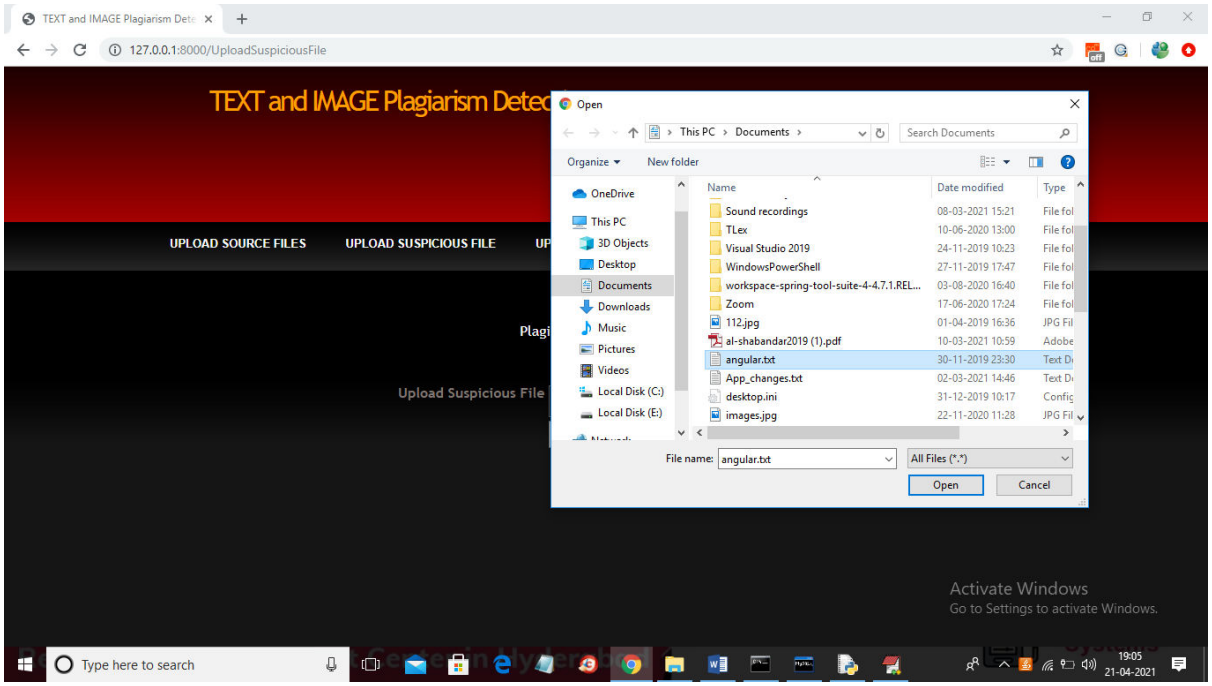
In above screen user signup process completed and now click on 'Login' link to get below screen



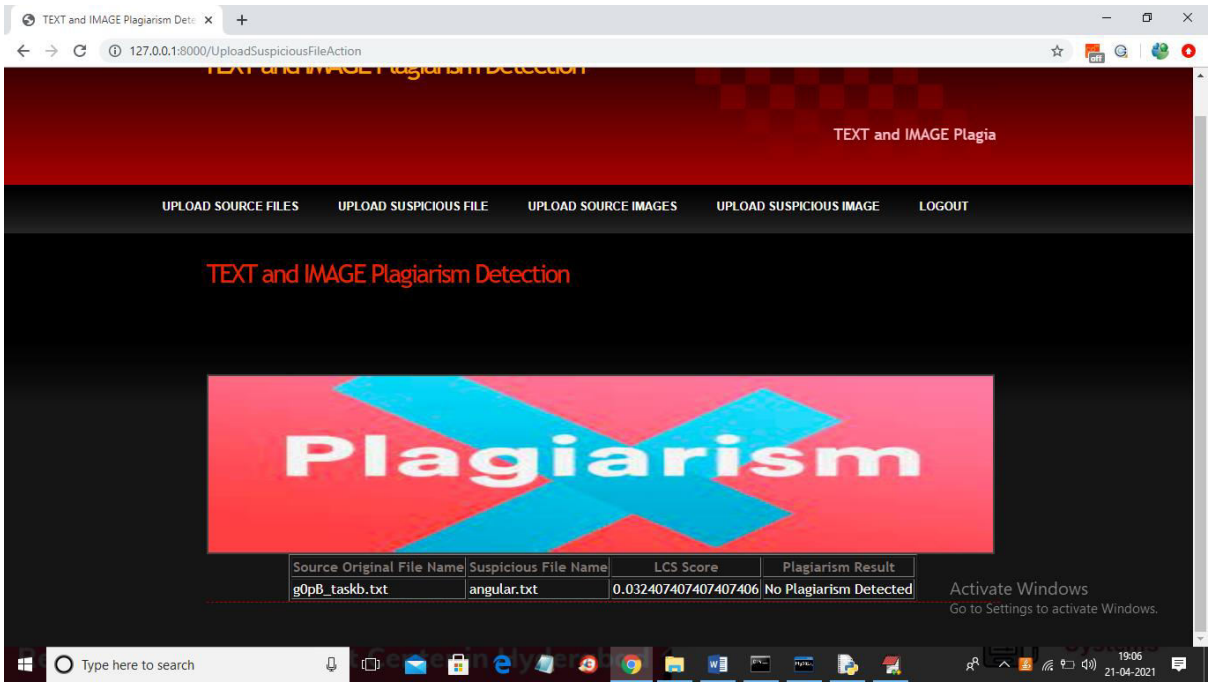
In above screen user is login and then click on button to get below screen



In above screen all files are loaded now click on 'Upload Suspicious File' button to load suspicious file and get result

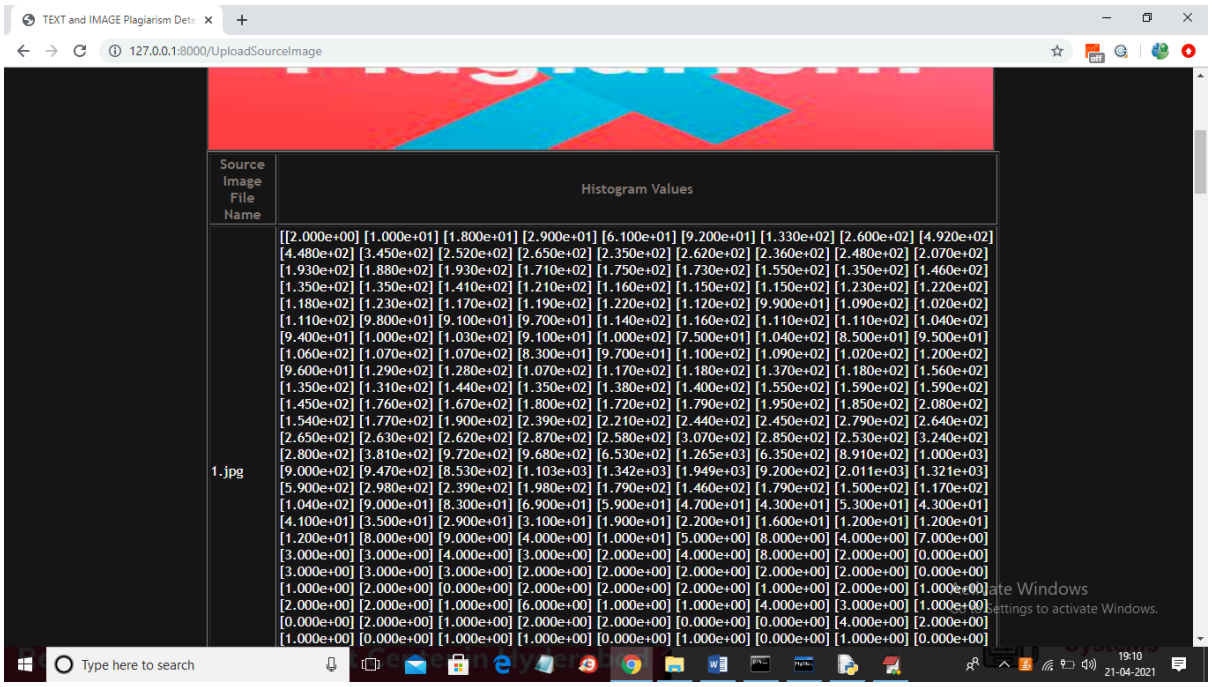


In above screen I am selecting and uploading 'angular.txt' file and then click on 'Open' button to get below result and then click on 'Check Plagiarism' button to get result

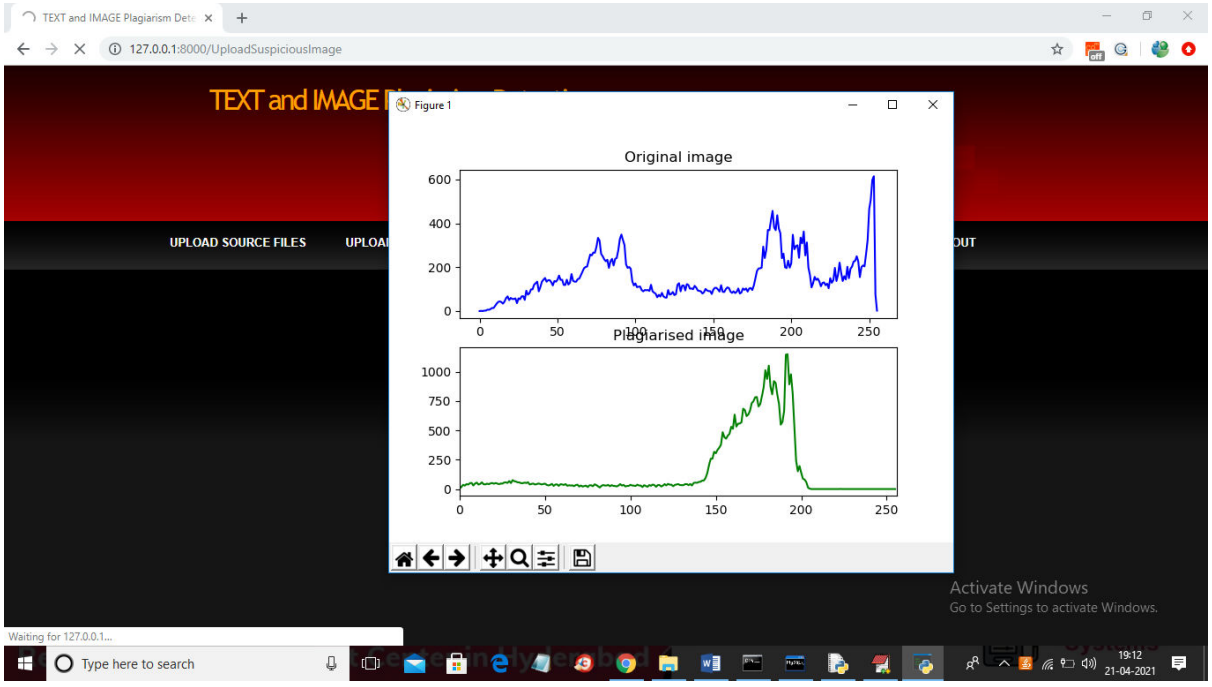


In above screen angular.txt file matched very little with g)pB_taskb.txt corpus file and we got similarity score as 0.03 so no plagiarism detected and now upload any file from corpus and see result

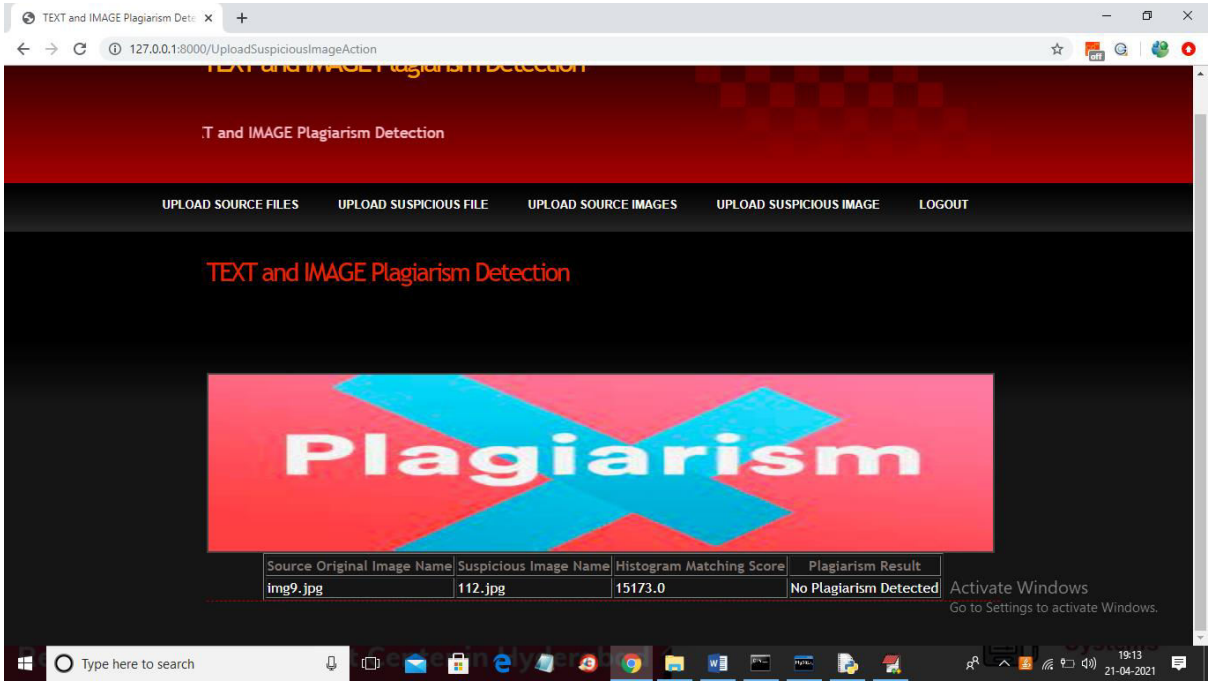
In above screen LCS score is 1.0 which means 100% matched with corpus file so plagiarism detected and similarly not only this u may enter any text file and get result. Now click on ‘Upload Source Images’ link to upload all images from ‘images’ folder



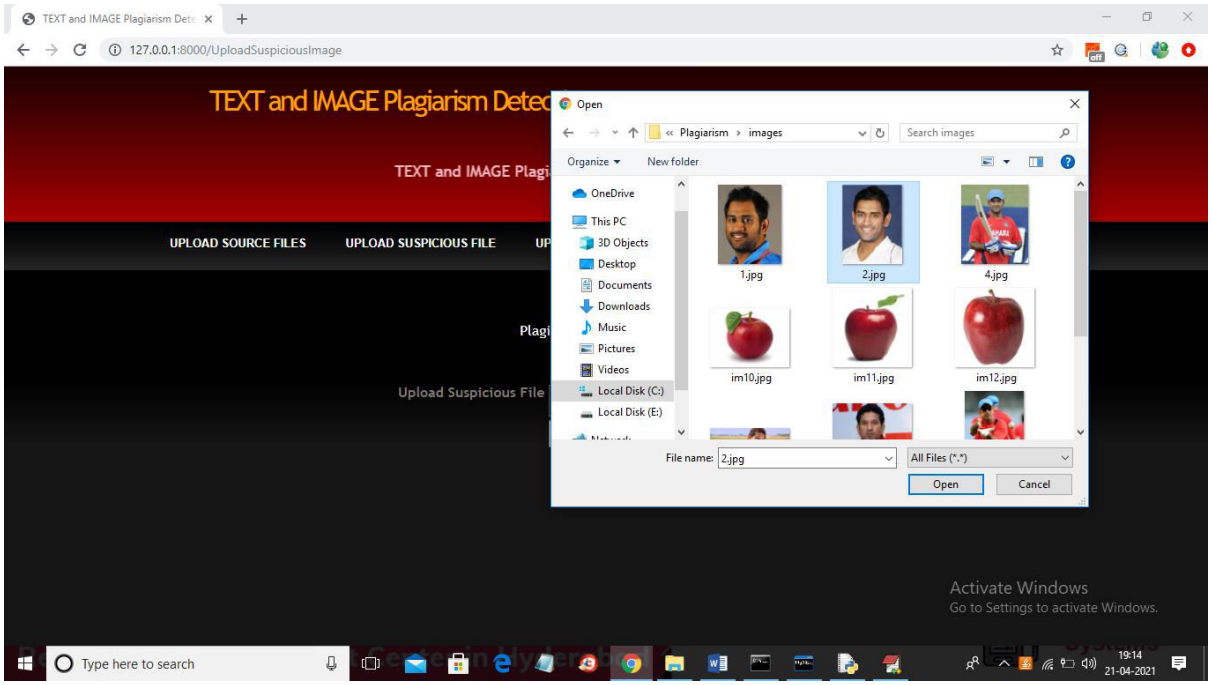
In above screen from all database images histogram will be calculated and store in array and whenever we upload new test image then both histogram will get matched and now click on 'Upload Suspicious Image' link to upload some image



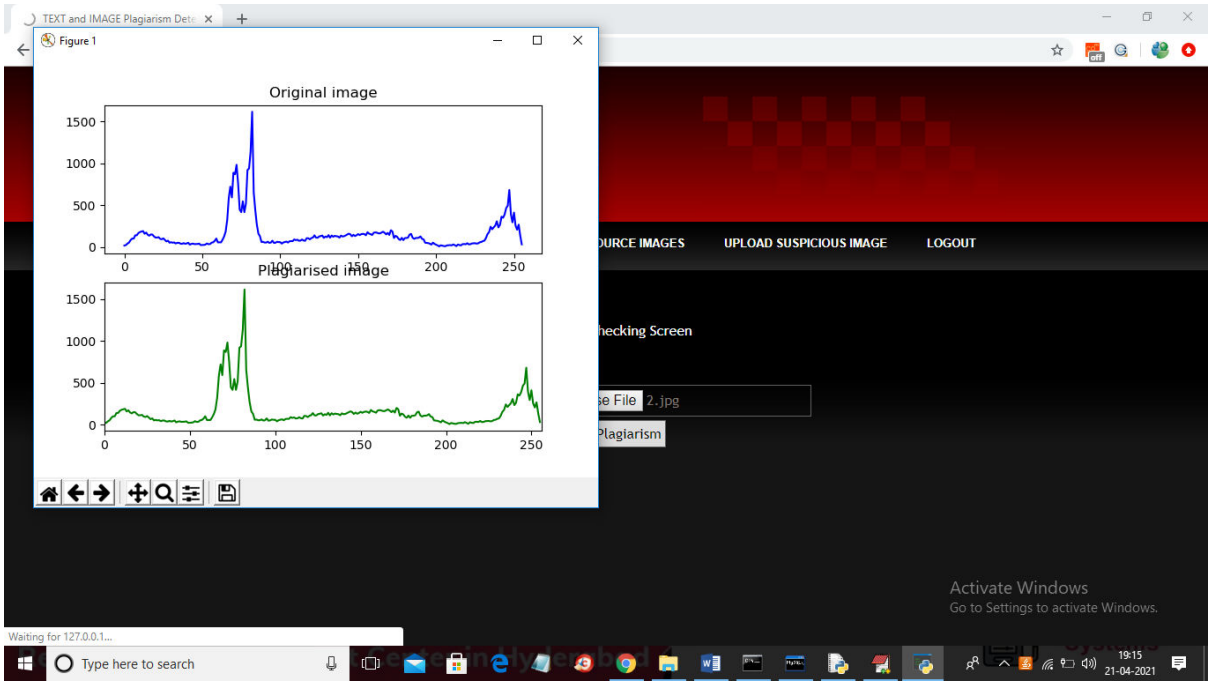
In above screen we can see for database image and uploaded image we generated histogram and we can see there is no match in histogram so no plagiarism will be detected and now close above graph to get below result



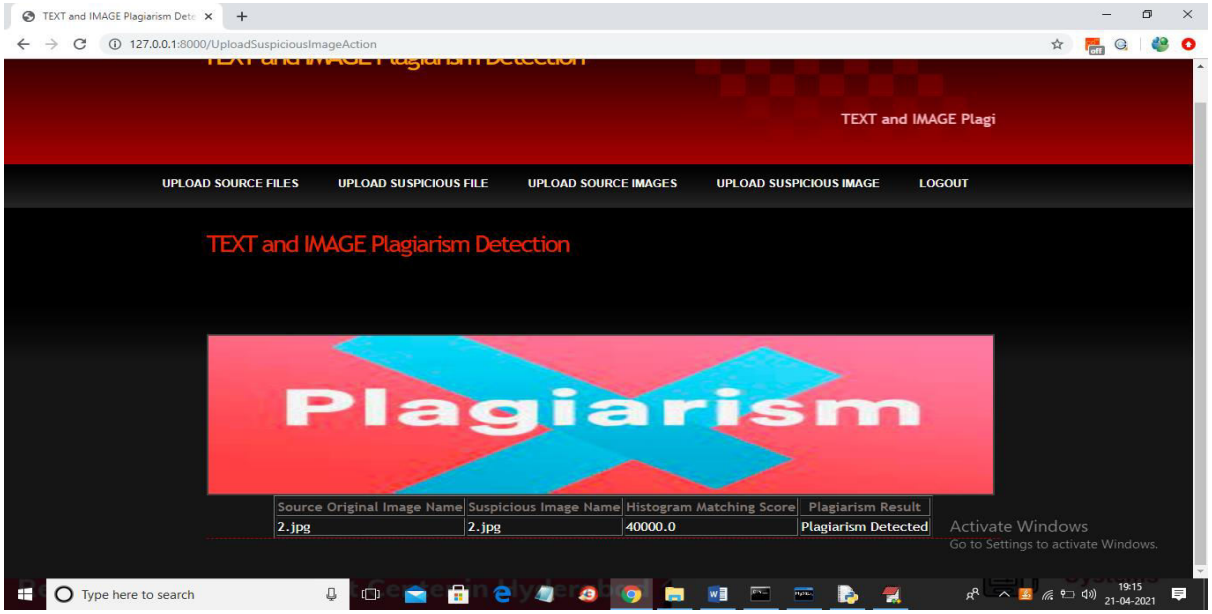
In above screen histogram pixel matching score is 15173 out of 40000 pixels so image is not plagiarised and now upload image from "images" folder and see result



In above screen I am selecting and uploading '2.jpg' file from "images" database folder and below is the result



In above screen we can both original and uploaded image histogram is matching 100% so plagiarism is detected and now close above graph to get below result



In above screen histogram matching score is 40000 which means all pixels matched so plagiarism is detected in above result.

Similarly u can upload any text file and image and test the application

VIII. CONCLUSION

This research presents a **comprehensive and efficient Plagiarism Detection System** capable of identifying both **textual and visual plagiarism**. Unlike traditional systems that often focus on only one content type or rely on exact matching, this framework integrates **natural language processing and computer vision** to handle nuanced real-world plagiarism scenarios.For textual comparison, applying the **Longest Common Subsequence algorithm** on cleaned, tokenized text provides robust similarity measures even when superficial changes like paraphrasing or synonym substitution are made. For images, the **Five Modules Method** combined with histogram intersection captures visual similarity despite minor edits or transformations. Together, these approaches form a multi-modal system that outperforms basic string matching or reverse search solutions.The web-based implementation using **Django** ensures scalability and accessibility, enabling users to upload files from a browser, receive real-time results, and visualize similarity metrics with intuitive charts and tables. Integration with **MySQL** ensures secure user authentication, while **OpenCV** and **NLTK** provide powerful libraries for image and text analysis.Experimental usage demonstrates high accuracy in identifying plagiarism across text documents and images. The modular design also allows for future enhancements, such as semantic embeddings using transformer models or neural network–based image feature extraction for even deeper similarity detection.

Overall, the system provides an effective solution for academic institutions, publishers, and content platforms seeking to enforce intellectual property integrity across multiple content formats. Its combination of algorithmic rigor, real-time performance, and user accessibility positions it as a valuable tool in the fight against plagiarism.

REFERENCES

1. Potthast, M., Stein, B., & Barrón-Cedeño, A., “Overview of the 7th International Competition on Plagiarism Detection,” *CLEF 2015*.
2. Foltynek, T., et al., “Academic Plagiarism Detection: Systematic Literature Review,” *Journal of Educational Computing Research*, 2020.
3. Deerwester, S., et al., “Indexing by Latent Semantic Analysis,” *Journal of the American Society for Information Science*, 1990.
4. Brin, S., & Page, L., “The Anatomy of Large-Scale Hypertextual Web Search Engine,” *Computer Networks*, 1998.
5. Chintalapudi, K. K., & Yadav, S., “Novel Techniques for Image Plagiarism Detection,” *IEEE Access*, 2021.
6. Zhang, Z., et al., “Deep Learning for Plagiarism Detection,” *Expert Systems with Applications*, 2020.
7. Long, J., et al., “Text Similarity Algorithms in Plagiarism Detection,” *Information Sciences*, 2022.

8. Cortes, C., & Vapnik, V., "Support Vector Networks," *Machine Learning*, 1995.
9. Sharma, D., "Image Histogram Matching in Digital Forensics," *Journal of Forensic Sciences*, 2021.
10. Fernandes, L., & Costa, J., "NLP Techniques in Document Analysis," *Computers & Education*, 2023.
11. Zahri, S. M., et al., "Textual Plagiarism Detection Using LCS and Cosine Similarity," *International Journal of AI*, 2021.
12. Samek, W., et al., "Explainable AI for Image Similarity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
13. Huang, X., et al., "Histogram-Based Image Forensics," *Digital Investigation*, 2020.
14. Li, Y., "Comparative Studies of Text Preprocessing on Detection Accuracy," *IEEE Access*, 2022.
15. Wang, S., & Chen, H., "Multi-Modal Plagiarism Detection Systems," *Journal of Informatics*, 2023.
16. Zhao, R., et al., "Machine Learning Approaches in Plagiarism Detection," *Journal of Intelligent Systems*, 2024.