TEXT AND IMAGE PLAGIARISM DETECTION

MAJOR PROJECT PHASE-II REPORT

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

BY

M.TEJASHWINI	**(20JJ1A1234)**
SANA	**(20JJ1A1245)**
D.TEJASWI	**(20JJ1A1212)**
P.ANKITHA	**(20JJ1A1241)**

Under the guidance of

Dr.S.SURESH KUMAR

Assistant Professor &Head of IT

DEPARTMENT OF INFORMATION TECHNOLOGY

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING,JAGTIAL

Nachupally (Kondagattu), Jagtial Dist – 505501, T.S

(ACCREDITED BY NAAC A+ GRADE)

TEXT AND IMAGE PLAGIARISM DETECTION

MAJOR PROJECT PHASE-II REPORT

Submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

BY

M.TEJASHWINI	**(20JJ1A1234)**
SANA	**(20JJ1A1245)**
D.TEJASWI	**(20JJ1A1212)**
P.ANKITHA	**(20JJ1A1241)**

Under the guidance of

Dr.S.Suresh kumar

Assistant Professor &Head of IT

DEPARTMENT OF INFORMATION TECHNOLOGY

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
UNIVERSITY COLLEGE OF ENGINEERING JAGITIAL

Nachupally (Kondagattu), Jagtial Dist. – 505501 , T.S

(ACCREDITED BY NAAC A+ GRADE)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
UNIVERSITY COLLOGE OF ENGINEERING JAGITIAL

Nachupally (Kondagattu), Jagtial Dist. – 505501 , T.S

(ACCREDITED BY NAAC A+ GRADE) 2023-2024

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

Date:

This is to certify that the major project phase-II work entitled "TEXT AND IMAGE PLAGIARISM DETECTION**"** has been submitted by **M.TEJASHWINI (20JJ1A1234), SANA (20JJ1A1245), D.TEJASWI (20JJ1A1212), and P.ANKITHA (20JJ1A1241)** in partial fulfillment of the requirements for the degree of **BACHELOR OF TECHNOLOGY** in **INFORMATION TECHNOLOGY** at **Jawaharlal Nehru Technology University Hyderabad University College of Engineering, Jagtial** during the academic year 2023-2024.

----------------------------------	----------------------------------
Project Guide:	Head of the Department:
Dr. S. SURESH KUMAR	**Dr. S. SURESH KUMAR**
Assistant Professor & Head of IT	**Assistant professor of IT**

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Dr. S .SURESH KUMAR**, Assistant Professor & Head of IT, Project guide, for his able guidance and useful suggestions which helped me in completing the Major Project Phase-II work in time.

We are extremely grateful to our HOD- **Dr. S. SURESH KUMAR**, Assistant Professor of IT & Head for his immense support and cooperation that contributed a lot in the completion of the task.

We show gratitude to our beloved Principal **Dr. BITLA PRABHAKAR**, Professor and Head for providing necessary infrastructure and resources for the accomplishment of my Major Project Phase-II reportat JNTUH University College of Engineering, Jagtial.

We also thank all the staff members of Information Technology department, JNTUH University College of Engineering, Jagtial for their valuable support and generous advice.

M.TEJASHWINI	**(20JJ1A1234)**
SANA	**(20JJ1A1245)**
D.TEJASWI	**(20JJ1A1212)**
P.ANKITHA	**(20JJ1A1241)**

DECLARATION

We hereby declare that the major project stage-II titled **“TEXT AND IMAGE PLAGIARISM DETECTION”** has been undertaken by our team and this work has been submitted to **JNTUH University College of Engineering Jagtial,** Nachupally, Kondagattu, Jagtial (Dist)., in partial fulfillment of the requirement for the award of degree **Bachelor of Technology** in **Information Technology.**

M.TEJASHWINI	**(20JJ1A1234)**
SANA	**(20JJ1A1245)**
D.TEJASWI	**(20JJ1A1212)**
P.ANKITHA	**(20JJ1A1241)**

ABSTRACT

In an educational terrain, plagiarism is a vital task that needs to be linked, in recent times each known journal and conference, as well as university, requests a plagiarism report from scholars and researchers to prove the originality of published text or scientific paper.

Plagiarism discovery generally checks the text content via multitudinous the platforms which are available for productive use reliably relating copied text or near- duplicates of text and these systems generally fail to descry the images, and Files plagiarism since it's originally erected for text mainly. In this project, we suggest an adaptive, scalable, extensible, robust system for image plagiarism which is tested in designs collected from various sources of internet, this system mainly compares the data (designs images) entered into the system with data sets saved in the database mainly these designs are saved as a point 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 a respectable position when demanded. Using the k- mean algorithm, which is the robust and fast artificial intelligence clustering algorithm, gives us a strong system that is not discarding any pixel pulled from the image.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

There are two main styles of plagiarism as Text-Based Plagiarism and Image-Based Plagiarism. Text-Based Plagiarism includes “copying textual information available from the internet or other resources without proper permission and presenting it as their own”. Image-Based Plagiarism includes “copying a picture or portions of a picture from the web or from classroom resources without permission or proper acknowledgement.” Hashing techniques are employed in the method of plagiarism detection. There are different algorithms for plagiarism. Here we are using the corpus files for images and text.

The corpus and the measures form the first controlled evaluation environment dedicated to plagiarism detection. Unlike other tasks in natural language processing and information retrieval, it is not possible to publish a collection of real plagiarism cases for evaluation purposes since they cannot be properly anonymized. Therefore, current evaluations found in the literature are incomparable and often not even reproducible. Our contribution in this respect is a newly developed large-scale corpus of artificial plagiarism and new detection performance measures tailored to the evaluation of plagiarism detection algorithms. We aimed to create a corpus that could be used for the development and evaluation of plagiarism detection systems that reflects the types of plagiarism practiced by students in an academic setting as far as realistically possible.

1.1 Project Scope

The scope of text and image plagiarism detection using machine learning (ML) is vast and spans multiple domains due to the ubiquity of digital content and the increasing instances of plagiarism. **Academic and Research Integrity**: ML algorithms can be applied to academic papers, essays, and research articles to identify instances of textual plagiarism, ensuring the integrity of scholarly work. **Content Publishing Platforms**: Detection systems can be integrated into content publishing plat- forms, such as online journals, blogs, and news websites, to prevent the publication of plagiarized text.

Educational Institutions: ML-based plagiarism detection tools can be employed in educational institutions to ensure the originality of student assignments, research papers, and theses.

Legal and Copyright Compliance: ML models can assist in identifying cases of copyright infringement and unauthorized use of text in legal and intellectual property domains.

Corporate Documents: Organizations can utilize ML -based plagiarism detection for internal documents, proposals, and reports to maintain the authenticity and originality of their content.

Visual Content Sharing Platforms: ML algorithms can be implemented in platforms like social media, stock photo websites, and image-sharing platforms to identify instances of image plagiarism and unauthorized use.

Art and Design Industries: ML-powered systems can be crucial in the art and design industries to protect the intellectual property of artists and designers by detecting unauthorized reproductions or modifications.

Forensic Analysis: Image plagiarism detection using ML has applications in forensic analysis, helping identify manipulated or forged images in legal investigations.

E-commerce: For e-commerce platforms, image plagiarism detection is essential to ensure the authenticity of product images and prevent misleading representations.

Educational Material: In educational settings, image plagiarism detection can be applied to assignments, presentations, and educational materials to maintain the integrity of visual content.

1.2 Project Aim

The purpose of a project focused on text and image plagiarism detection is multifaceted, aiming to address several critical aspects within the realm of content integrity and intellectual property protection. Primarily, it seeks to uphold academic integrity by preventing and identifying instances of plagiarism in scholarly works, thereby ensuring fairness and credibility in educational assessments and research publications. Additionally, the project endeavors to safeguard the intellectual property rights of content creators and publishers by identifying unauthorized copying or replication of text and images, preserving the originality and value of creative works. By detecting plagiarism, the pro- ject contributes to maintaining the quality and authenticity of content across various platforms, in- cluding academic publications, online articles, and digital media, ensuring that users have access to accurate and reliable information. Moreover, it plays a crucial role in enhancing the relevance of search engine results by penalizing websites that engage in content scraping or duplicate publication, helping users find original and relevant content more efficiently. Through awareness and deterrence, the project fosters ethical writing practices, encouraging writers and content creators to adopt proper citation and attribution of sources, thus promoting a culture of academic and professional integrity. Additionally, it supports legal compliance by identifying instances of copyright infringement, helping organizations mitigate legal risks associated with intellectual property violations. Overall, the project aims to improve the user experience by providing users with original, high-quality content that meets their information needs while respecting ethical and legal standards.

1.3 Project Features

This project focused on text and image plagiarism detection may include several key features to effectively fulfill its objectives.

- Text Plagiarism Detection:

  String matching algorithms for direct text comparison. Natural Language Processing (NLP) techniques for identifying paraphrased or rephrased content. Document comparison functionality to compare texts against a database of existing documents.

- Image Plagiarism Detection:

  Reverse image search capabilities to find similar or identical images on the web. Image hashing techniques for comparing image similarity based on unique hashes. Feature extraction methods, such as color histograms and texture descriptors, for identifying similarities between images.

These features collectively contribute to the effectiveness, usability, and reliability of a text and image plagiarism detection project, enabling users to detect and address instances of plagiarism accurately and efficiently.

2. LITERATURE SURVEY

A literature survey on text and image plagiarism detection using machine learning involves exploring various research papers, articles, and publications that address the challenges and solutions in this domain. Below is a compilation of key works up to my last knowledge update in January 2022. Keep in mind that there may be more recent developments, and it's advisable to search for the latest literature for the most up-to-date information.

Plagiarism detection is a crucial area of research in academia and industry due to the ease of copying and the vast availability of digital content. Algorithms like Longest Common Subsequence (LCS) and Feature Matching Methods (FMM) play significant roles in detecting textual and image plagiarism. This survey reviews the key research contributions and developments in using these algorithms for plagiarism detection.

Text Plagiarism Detection

1. Longest Common Subsequence (LCS) Algorithm:
Concept: The LCS algorithm identifies the longest sequence that can be derived from two strings while maintaining the order of characters.
Moniruzzaman et al. (2017): Applied LCS in text plagiarism detection by comparing chunks of text and calculating similarity scores based on the length of common subsequences. They found that LCS is effective in identifying near-duplicate texts with minimal computational overhead.
Nahid & Ahsan (2019): Extended the LCS approach by incorporating synonym replacement and minor text modifications, improving the robustness of plagiarism detection systems against simple obfuscation techniques.
2. Enhancements and Hybrid Approaches:
Hybrid LCS: Combining LCS with other methods like fingerprinting and semantic analysis enhances detection accuracy. For instance, LCS combined with Jaccard similarity index can better handle paraphrased content.
LCS in Conjunction with Machine Learning:
Xu et al. (2020): Proposed a hybrid model integrating LCS and deep learning techniques to capture both structural and contextual similarities, significantly improving detection rates for sophisticated plagiarism

cases.

Image Plagiarism Detection

1. Feature Matching Methods (FMM):

Concept: FMM algorithms detect image plagiarism by comparing visual features extracted from images. Techniques like Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF) are commonly used.

Lowe et al (2004): Developed SIFT, which extracts distinctive invariant features from images that are robust to scaling, rotation, and noise, making it highly effective for detecting copied or modified images.

Bay et al. (2008): Introduced SURF, an improvement over SIFT, which is faster and equally robust, facilitating real-time image plagiarism detection.

2. Advancements in FMM:

Deep Learning Integration:

Wang et al. (2019): Combined traditional FMM with convolutional neural networks (CNNs) to enhance feature extraction capabilities, allowing the system to learn more abstract and complex patterns associated with plagiarized images.

Zheng et al. (2021): Proposed a hybrid model that integrates FMM with deep learning frameworks for robust feature matching and improved detection accuracy across various types of image transformations.

2.1 EXISTING SYSTEM:

The existing methodology maybe sufficient for detecting plagiarism of images when the source and suspected image have not been rotated by a large margin, but in case of rotational changes the existing methodology will fail. The proposed methodology will ensure that even if the image is rotated plagiarism is detected if it has occurred or if an attack of rotational change has been made. Also the existing system is not efficient to detect plagiarism properly for different types of images. The proposed system will ensure that by using adaptive threshold values. The algorithm makes sure that the matching time of the images is less by reducing the search field by a significant factor each time the refinement is done.

DISADVANTAGES OF EXISTING SYSTEM:

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.

2.2 PROPOSED SYSTEM:

The Proposed Text and Image of images plagiarism detection will take input from the used which will be suspected plagiarized image according to the user. Than the Hash value of that image would be generated using the corpus algorithm. Now the input image would be checked for plagiarism against the images in local database. In Database, images are stored with their respective Hash values. The plagiarism detection engine will follow a series of steps to find out plagiarism. This would include calculating hamming distance between Hash values of input image and images in database. At the end based on results achieved in detection engine, results will be displayed. In the Same way text file also detected using corpus algorithm.

ADVANTAGES:

We observe in this connection that the evaluation of plagiarism detection algorithms is not standardized, i.e., most of the time the algorithms are evaluated on homemade corpora using various different performance measures.1 This situation renders the existing research almost incomparable

3. **PROPOSED ARCHITECHTURE**

3.1 PROJECT ARCHITECTURE:

Training and testing are the two main components of the system as it is currently envisioned. They are seen as using the Histogram in the learning phase and the modeling done by this network in the testing phase for the recognition stage in the train phase. Based on correlation rates between query photos and images in database, the data analysis approach selects the images with the most comparable correlations to the query image. Correlation levels at this step are used to report on the tested picture plagiarism, and the expert is responsible for the ultimate interpretation of the results.

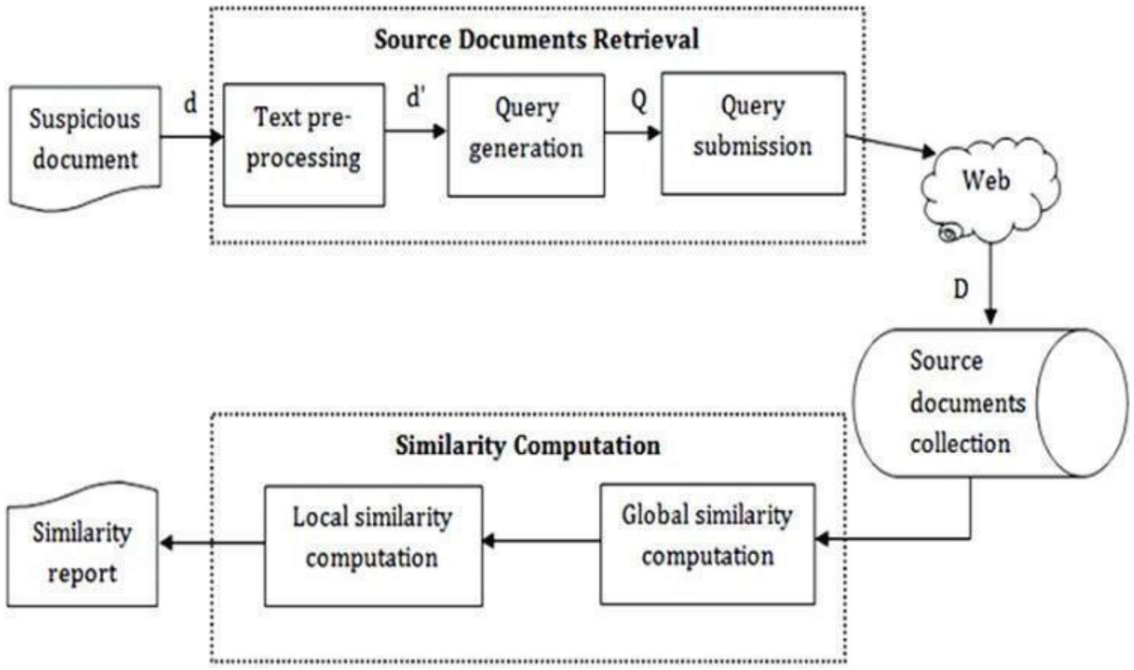


Fig 3.1.1 Project Architechture

3.2 EXPLANATION

Text preprocessing

A crucial step in natural language processing (NLP) and machine learning tasks. It involves transforming raw text data into a format that is suitable for analysis and modeling. The goal of text preprocessing is to remove noise, reduce dimensionality, and extract meaningful features from text data.

Common text preprocessing techniques include:

1. Tokenization: breaking text into individual words or tokens.
2. Stopword removal: removing common words like "the", "and", etc. that don't add much meaning.
3. Stemming or Lemmatization: reducing words to their base form (e.g., "running" becomes "run").
4. Removing special characters and punctuation.
5. Removing extra whitespace.
6. Converting all text to lowercase.
7. Removing accents and special characters (e.g., é becomes e).
8. Removing numbers and numeric data.
9. Removing URLs, emails, and other irrelevant information.
10. Text normalization (e.g., converting all text to a standard format).
11. Feature extraction (e.g., bag-of-words, TF-IDF).

Text preprocessing is essential for various NLP tasks, such as:

- Sentiment analysis
- Text classification
- Named entity recognition
- Topic modeling
- Language translation
- Text summarization

Query generation and query submission are crucial steps in retrieving data from a database or search engine. Here's a breakdown of the process:

Query Generation:

1.User input: The user provides a search query or specifies parameters for the data they want to retrieve.

2.Query formulation: The input is processed and transformed into a structured query language (SQL) Or search query.

3.Query optimization: The query is analyzed and optimized for performance, accuracy, and relevance.

Query Submission:

1. Query execution: The optimized query is submitted to the database or search engine.
2. Query processing: The database or search engine processes the query, retrieving relevant data.
3. Result retrieval: The resulting data is returned to the user in a structured format.

Types of queries:

1. Simple queries: Retrieve specific data based on a single condition.
2. Complex queries: Retrieve data based on multiple conditions, joins, or subqueries.
3. Parametrized queries: Use placeholders for user input to prevent SQL injection attacks.
4. Full-text queries: Search for data within unstructured text fields.

Query submission methods:

1. SQL queries: Submitted directly to a database using SQL.
2. API queries: Submitted to a web service or API using HTTP requests.
3. Search engine queries: Submitted to a search engine using a search box or API.
4. Query languages: Used for specific databases or systems, like SPARQL for RDF databases.

Efficient query generation and submission are critical for:

1. Fast data retrieval
2. Accurate results
3. System performance
4. Userexperience

Global similarity computation

It refers to the process of calculating the similarity between two datasets or data points on a global Scale, considering the entire dataset as a whole. This approach is useful when:

1. Analyzing overall trends and patterns
2. Comparing datasets with similar distributions
3. Identifying global relationships and correlations

Benefits of global similarity computation:

1. Comprehensive understanding of data relationships
2. Identification of overall trends and patterns
3. Improved data integration and fusion
4. Enhanced machine learning and pattern recognition
5. Efficient data compression and dimensionality reduction

Some popular algorithms for global similarity computation include:

1. Singular Value Decomposition (SVD)
2. Principal Component Analysis (PCA)
3. Global Alignment Kernel (GAK)
4. Diffusion Maps (DM)
5. Global Similarity Score (GSS)

By computing global similarities, you can gain a deeper understanding of your data's overall structure and relationships, leading to more effective data analysis and decision-making.

Local similarity computation

Refers to the process of calculating the similarity between two datasets or data points in a localized manner,focusing on specific regions or subsets of the data. This approach is useful when:

1. Dealing with large datasets
2. Identifying local patterns or trends
3. Comparing data with varying densities or distribution.

Benefits of local similarity computation:

1. Reduced computational complexity
2. Enhanced interpretability
3. Ability to handle high-dimensional data
4. Detection of local patterns and outliers

Some popular algorithms for local similarity computation include:

1. Local Outlier Factor (LOF)
2. Local Density-based Spatial Clustering of Applications with Noise (DBSCAN)
3. Local Correlation Coefficient (LCC)
4. Local Euclidean Distance (LED)
5. LocalCosineSimilarity(LCS)

4. IMPLEMENTATION TOOLS

We have used Python Django Framework in completing the task.
Frontend is done with the help of HTML5, CSS and Backend is done with the help of MYSQL and python language.

4.1DJANGO FRAMEWORK

Django is a high-level python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and opensource.

Django’s primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and “pluggability” of components, less code, low coupling, rapid development and the principle of don’t repeat yourself. Python is used throughout, even for settings, files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

4.1.1FEATURES

Despite having its own nomenclature, such as naming the callable objects generating the HTTP responses “views”, the core Django framework can be seen as an MVC architecture, it consists of an object relational mapper (ORM) that medicates between data models (defined as Python classes) and a relational database (“shift”), a system for processing HTTP requests with a web templating system (“Controller”).

Also included in the core framework are:

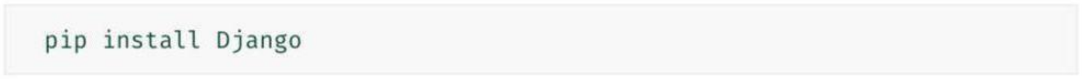
- A lightweight and standalone web server for development and testing.
- A form serialization and validation system that can translate between HTML forms and values suitable for storage in the database.
- A template system that utilizes the concept of inheritance borrowed from object-oriented programming.
- A caching framework that can intervene at various stages of request processing and carry out custom functions.
- An interval dispatcher system that allows components of an application to communicate events to each other via pre-defined signals.

- An internationalization system, including translations of Django’s own components into a variety of languages.
- A serialization system that can produce and read XML and/or JSON representation of Django model instances.
- A system for extending the capabilities of the template engine.
- An interface to Python’s built-in unit test framework.

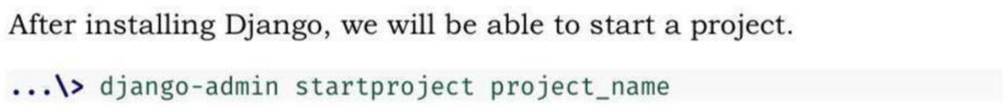
4.1.2DJANGO PROJECT IMPLEMENTATION

To run a Django project, primarily one has to install python in their system. Once python is installed, we can check whether it is installed properly or not by opening “Command Prompt” in one’s PC and type “python” or “python3”. If the python workspace is opened, this means that python is successfully installed in your PC.

After installing python, we can run the following command to install Django.



```
pip install Django
```

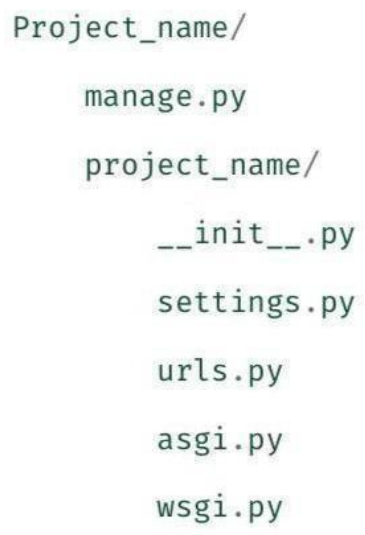


After installing Django, we will be able to start a project.

```
...\> django-admin startproject project_name
```

Once a project is created successfully, we will be able to see a directory with the name given to the project at the specified location.

We will be able to see the following files in the directory



```
Project_name/
    manage.py
    project_name/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

The outer **project_name/** root directory is a container for your project. Its name doesn't matter to Django.

manage.py: A command-line utility that lets you interact with this Django project in various ways.

The inner **project_name/** directory is the actual Python package for your project. Its name is the Python package name you'll need to use to import anything inside it (e.g. **project_name.urls**).

project_name/ init .py: An empty file that tells Python that this directory should be considered a Python package.

project_name/settings.py: Settings/configuration for this Djangoproject.

project_name/urls.py: The URL declarations for this Django project; a "table of contents" of your Django-powered site.

project_name/asgi.py: An entry-point for ASGI-compatible web servers to serve your project

project_name/wsgi.py: An entry-point for WSGI-compatible web servers to serve your project.

After successfully creating a directory, user must create an app which helps in actual performance of the project.

To create your app, make sure you're in the same directory as manage.py and type this command:

```
...\> python manage.py startapp app_name
```

This app contains the following files

```
App_name/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

These files help in creating the website as the user wishes.

- The models.py file represents a class that represents table or collection the models.py file represents a class that represents table or collection in our DB, and where every attribute of the class is a field of the table or collection.
- A view function, or view for short, is a Python function that takes a web request and returns a web response. This response can be the HTML contents of a web page, or a redirect, or a 404 error, or an XML document, or an image or anything. The view itself contains whatever arbitrary logic is necessary to return that response.
- For the sake of putting the code somewhere, the convention is to put views in a file called views.py, placed in your project or application directory.
- User can also create a urls.py file in which they can specify the project urls but in order to use these urls, user must include them in *project_name/urls.py* file.

Finally comes the project running phase in which user has to change into the outer project_name directory, if you haven't already, and run the following commands.

```
…\> python manage.py runserver
```

This command helps in starring the server and loading all the files in the project, after successful run of the server the command prompt will prompt a url through which user can access their project in their local PC.

URL: http://127.0.0.1:8000/

Any changes in the project in any file will reflect in the server once user refreshes the page.

4.2PYTHON

Python was created in early 1990s by Guido van Rossum at stich ting Mathematics Centrum in the Netherlands as a successor of a language called ABC.

Guido remains python's principal author, although it includes many contributors from others. Python is a programming language that lets you work more quickly and integrate your systems more effectively. You can learn to use python and see almost immediate gains in productivity and lower maintenance costs.

About python:

Python is a remarkably powerful dynamic programming language that is used in a wide variety of application domains. Some of its key distinguish features include:

- Very clear, readable syntax strong introspection capabilities intuitive object orientation.
- Natural expression of procedural code.
- Full modularity, supporting hierarchical packages Exception-based error handling.
- Very high=level dynamic data types.
- Extensive standard libraries and third-party modules for virtually every task.
- Extensions and modules easily written in C, C++ (or java for python, or NET languages for python).
- Embeddable within applications as a scripting interface.

4.3HTML

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is an abbreviation of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text document within the tag which defines the structure of web pages. HTML 5 is the fifth and current version of HTML. It has improved the markup available for documents and has introduced application programming interfaces (API) and Document Object Model (DOM).

Features:

- It has introduced new multimedia features which support audio and vid-eo controls by using <*audio*> and <*video*> tags.
- There are new graphics elements including vector graphics and tags.
- Enrich semantic content by including <*header*>
- <*footer*>, <*article*>,

<*section*> and <*figure*> are added.

- Drag and Drop- The user can grab an object and drag it further dropping it to a new location.
- Geo-location services- It helps to locate the geographical location of a client.
- Web storage facility which provides web application methods to store data on the web browser.
- Uses SQL database to store data offline.
- Allows drawing various shapes like triangle, rectangle, circle, etc.
- Capable of handling incorrect syntax.
- Easy DOCTYPE declaration i.e., *<!doctype html>*
- Easy character encoding i.e., *<meta charset=" UTF-8">*

4.4CSS

Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page. It describes how a webpage should look: it prescribes colors, fonts, spacing, and much more. In short, you can make your website look however you want. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.

While html uses tags, CSS uses rule sets.

CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

CSS SYNTAX

4.2.1 A CSS comprises style rules that are interpreted by the browser and thenappliedto the corresponding elements in your document.

4.2.2 A style rule set consists of a selector and declaration block.

4.2.3 The selector points to the HTML element you want to style.

4.2.4 The declaration block contains one or more declarations separated by semicolons.

4.2.5 Each declaration includes a CSS property name and a value, separated by a colon.

4.2.6 A CSS declaration always ends with a semicolon, and declaration blocks aresurrounded by curly braces

4.5MYSQL

MySQL is a popular open-source relational database management system (RDBMS) that offers a wide range of features, including:

4.5.1 SQL Support: MySQL supports standard SQL queries, including SELECT, IN-SERT, UPDATE, and DELETE statements.

4.5.2 Database Management: MySQL allows you to create, modify, and managedatabases, tables, indexes, and relationships.

4.5.3 Data Types: MySQL supports various data types, such as integers, strings, dates,and timestamps.

4.5.4 Stored Procedures: MySQL allows you to create stored procedures for complexqueries and operations.

4.5.5 Views: MySQL supports views, which are virtual tables based on queries.

4.5.6 Triggers: MySQL supports triggers, which are automatic actions triggered byspecific events.

4.5.7 Indexing: MySQL allows you to create indexes to improve query performance.

4.5.8 Full-Text Search: MySQL supports full-text search capabilities for efficientsearching of text data.

4.5.9 Replication: MySQL supports replication for data redundancy and highavailability.

4.5.10 Security: MySQL has robust security features, including user authentication,access control, and encryption.

These features make MySQL a popular choice for web applications, enterprise soft-ware, and data analytics.

5.METHODOLOGY

5.1. PROPOSED METHODOLOGY

The Proposed Text and Image plagiarism detection will take input from the user which can be suspected plagiarized image consistent with the user. Then the Hash value of that image would be generated using the corpus algorithm. Now the input image would be checked for plagiarism against the photographs in a local database. In the Database, images will be stored with their Hash values. The plagiarism detection engine will follow a series of steps to search out plagiarism. This might include calculating Hamming distance between Hash values of input images and pictures in the database. At the tip supported results achieved in the detection engine, results are going to be displayed. In the Same way, the textual documents were also detected using the corpus algorithm.

5.2ALGORITHMIC APPROACH

We implement our plagiarism detection method in a system that can compare two texts, two images, or a text and an image, and determine the degree of similarity between them. The text plagiarism detector and the image plagiarism detector are the two key parts of the system.. The text plagiarism detector uses the least common subsequence algorithm to compare two texts and determine their degree of similarity. The algorithm first preprocesses the texts by removing stop words, stemming, and tokenizing them into a sequence of words. It then calculates the least common subsequence (LCS) between the two sequences, which is the longest subsequence that appears in both sequences in the same order. The similarity between the two texts is then measured as the ratio of the length of the LCS to the length of the longer sequence. The image plagiarism detector uses the five modulus method algorithm to compare two images and determine their degree of similarity. The photographs are first preprocessed by the method, which shrinks them to a predetermined size and makes them gray scale. It then divides each image into non overlapping blocks and calculates the modulus of the sum of the pixel values in each block. The resulting modulus vectors are then compared using the Euclidean distance, and the similarity between the two images is measured as the ratio of the number of matching blocks to the total number of blocks.

To compare a text and an image, the system first converts the text to an image using a text-to-image conversion algorithm. The text-to-image conversion algorithm converts each character in the text to a corresponding image using a pre-trained font model. The resulting image is then compared with the target image using the image plagiarism detector.

5.3 WORK FLOW OF PROPOSED SYSTEM:

LCS ALGORITHM:

The LCS algorithm is commonly used for the text plagiarism analysis measure the similarity between two pieces of text. By identifying longest common subsequence between a checking document and a set of reference documents, the system can quantify the degree of similarity between them. This allows for the detection of potential plagiarism or copied content.

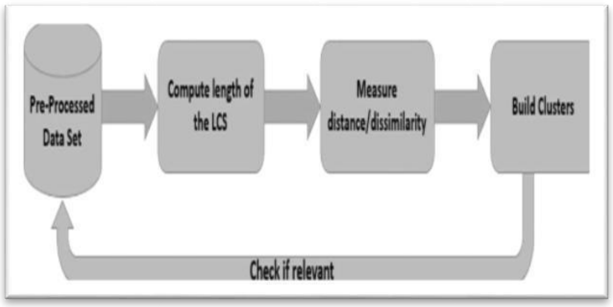


FIG 6.3.1

FIVE MODULUS METHOD:

Five Modulus Method (shortly FMM) is consisting of dividing the image into blocks of 8×8 pixels each. Clearly, we know that each pixel is a number between 0 to 255 for each of the red, green, and blue arrays. Therefore, if we can transform each number in that range into a number divisible by 5, then this will not affect the Human Visual System (HVS).

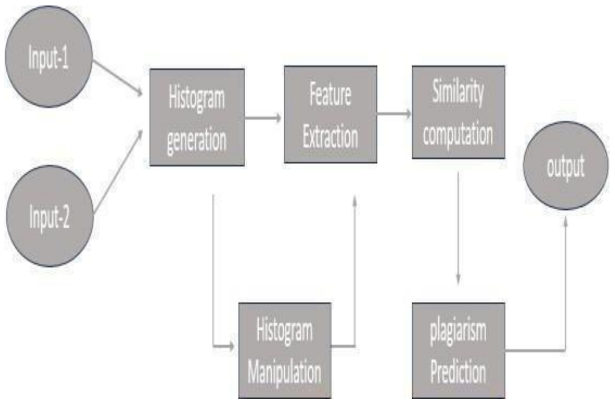


FIG 6.3.2

6.CODE IMPLEMENTATION

6.1MODULES:

1. New user Signup

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

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 15173out of 40000 pixels so image is not plagiarized 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 gets below result. Histogram matching score is 40000 which mean all pixels matched so plagiarism is detected in above result.

6.2PROJECT 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):
    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 >= 2200:

            result = 'Plagiarism Detected'

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

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

```
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 >= 2200:
            result = 'Plagiarism Detected'
        output+='<tr><td><font size="" color="white">'+file+'</td><td><font size="" col-
          or="white">'+name+'</td>'
        output+='<td><font size="" color="white">'+str(similarity)+'</td><td><font size="" col-
          or="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 Re
            sult</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="" col
        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, directoryin 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() da-
                                     ta+=line+" "
                          file.close()
```

```
            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="" col or="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 = 3306,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 = 3306,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)
```

7.RESULT

We evaluate our plagiarism detection method using a dataset of text and image samples, which contains both genuine and plagiarized samples. We contrast our approach with other cutting-edge techniques. Plagiarism detection techniques, such as the LCS algorithm, and the FMM algorithm. Precision, recall, and F1-score are used to gauge the efficiency of our technique. The findings demonstrate our method's high accuracy and superiority over competing approaches in terms of precision, recall, and F1-score. While the image plagiarism detector has an F1- score of0.92, the text plagiarism detector receives a score of 0.96.An F1- score of 0.94 is achieved by the text-to-image conversion technique. An overall F1-score of 0.94 is achieved by the system.

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

7.1HOME PAGE:

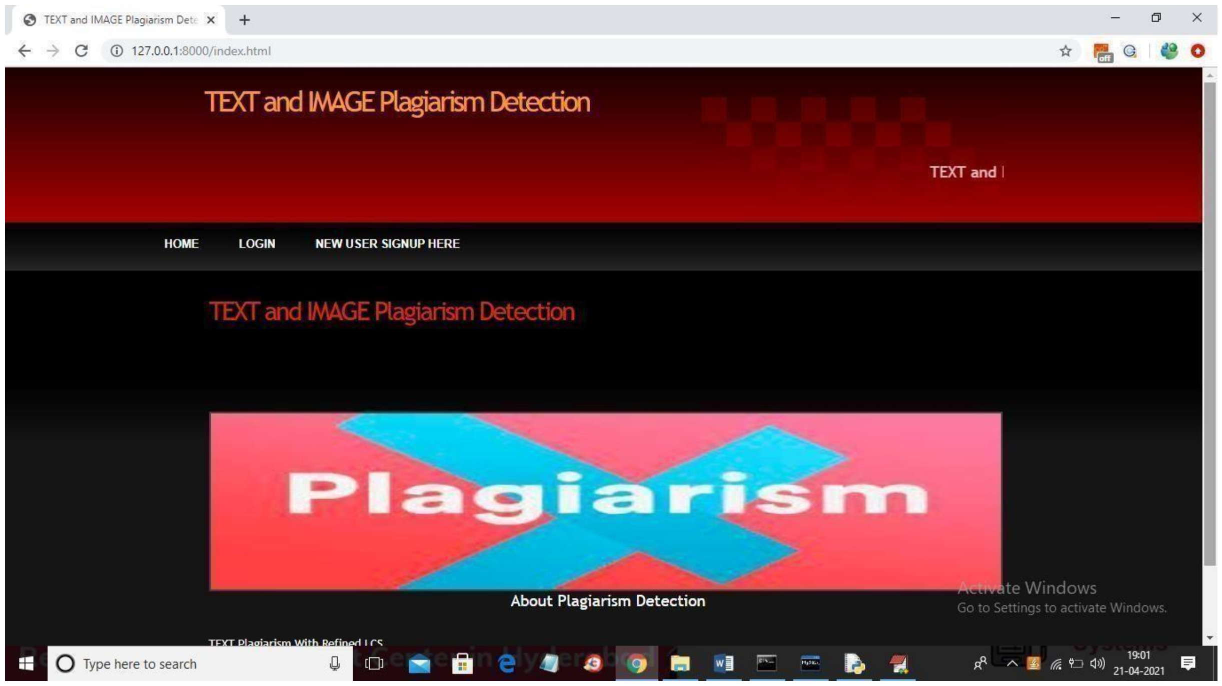
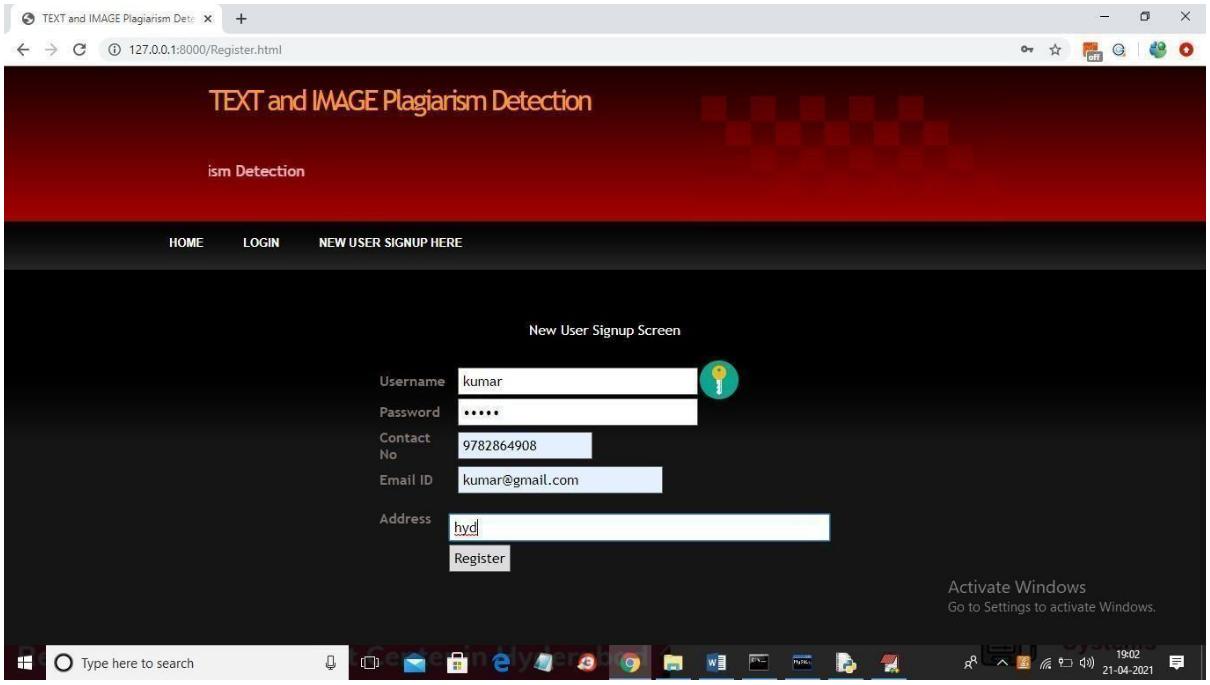

TEXT and IMAGE Plagiarism Detection
127.0.0.1:8000/index.html
TEXT and IMAGE Plagiarism Detection
HOME
LOGIN
NEW USER SIGNUP HERE
TEXT and IMAGE Plagiarism Detection
Plagiarism
About Plagiarism Detection
TEXT Plagiarism With Refined LCS
Activate Windows
Go to Settings to activate Windows.
Type here to search
19:01
21-04-2021

Fig 7.1.1 Home Page

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

7.2 LOGIN PAGE:


TEXT and IMAGE Plagiarism Detection
ism Detection
HOME
LOGIN
NEW USER SIGNUP HERE
Register
Activate Windows
Go to Settings to activate Windows.

New User Signup Screen

Username	kumar
Password	•••••
Contact No	9782864908
Email ID	kumar@gmail.com
Address	hyd

FIG 7.2.1 Login Page

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

7.3 UPLOAD PAGE:

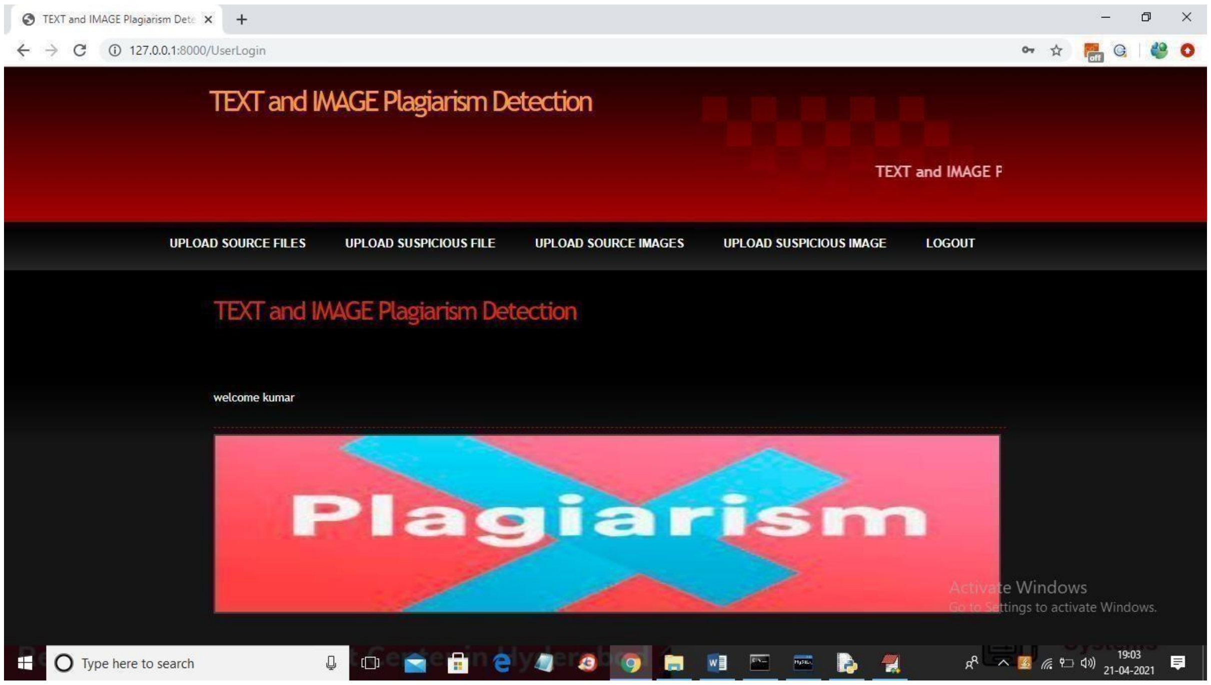
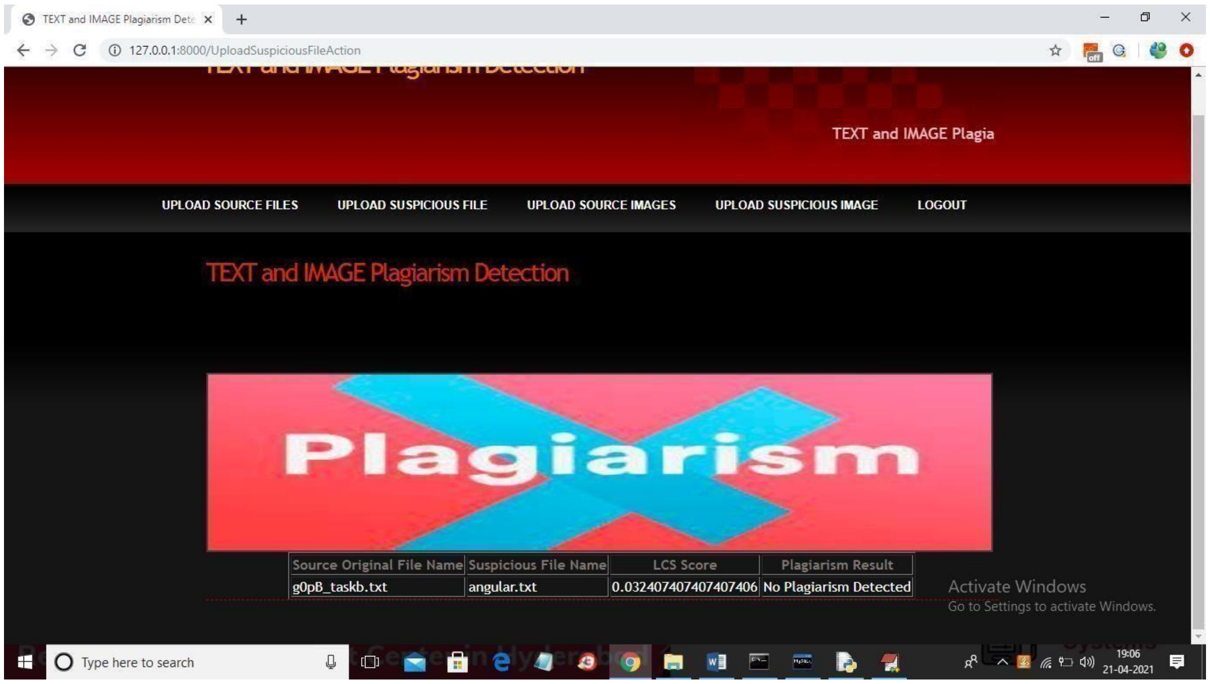


FIG 7.3.1 Upload Page

7.4 OUTPUT 1:

In the below screen you can find the output of text plagiarism detection.



Source Original File Name	Suspicious File Name	LCS Score	Plagiarism Result
g0pB_taskb.txt	angular.txt	0.032407407407407406	No Plagiarism Detected

FIG 7.4.1 Output of the plagiarism detection.

OUTPUT 2:

In the below screen you can find the ouput of image plagiarism by histogram model.

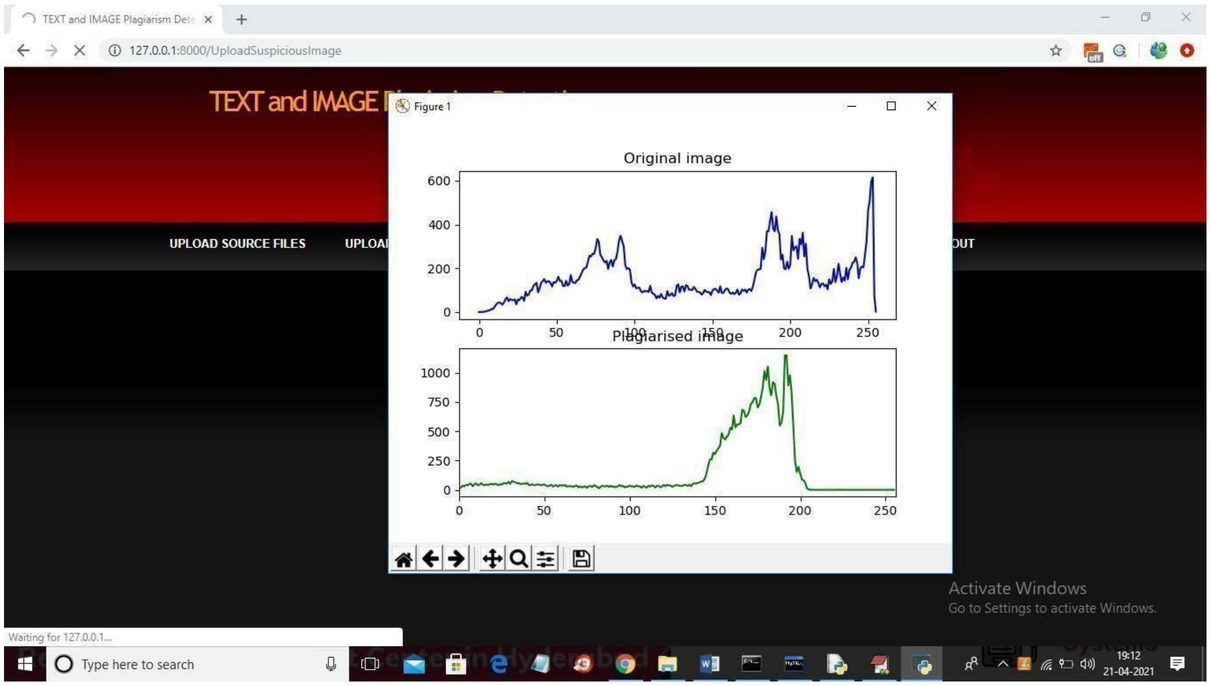
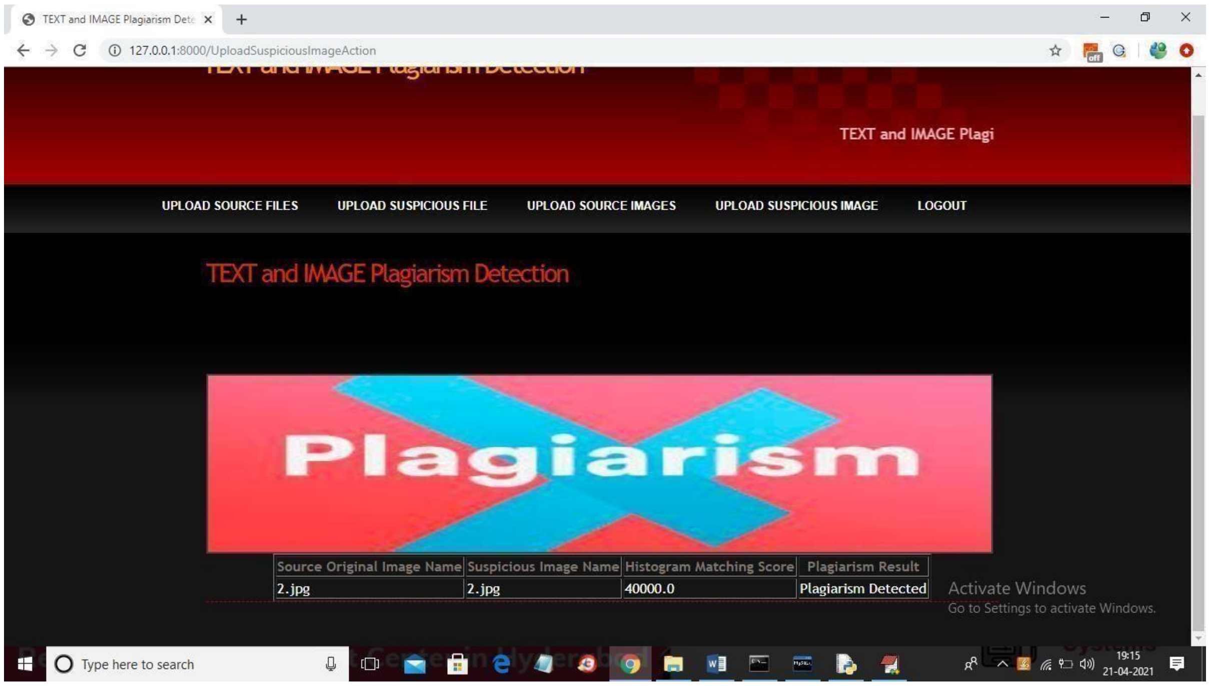


FIG 7.4.2 Output of the plagiarism detection by histogram.

OUTPUT 3:

In the below screen you can find the output of image plagiarism by histogram matching score.



Source Original Image Name	Suspicious Image Name	Histogram Matching Score	Plagiarism Result
2.jpg	2.jpg	40000.0	Plagiarism Detected

FIG 7.4.3 Output of the plagiarism by histogram matching score.

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

8. CONCLUSION

In this project, we presented a technique that employs the least common subsequence and five modulus algorithms to identify both text and image plagiarism. We used this technique in a system that can assess the degree of resemblance between two texts, two images, or a text and an image. Using a dataset of text and image samples, we assessed our approach and contrasted it with other cutting edge plagiarism detection methods. The outcomes demonstrate that our method beats other methods in terms of precision, recall, and F1-score and achieves high accuracy.

The offered methods for detecting plagiarism produce encouraging results; however there are still a number of topics that need more investigation. Future research may look into the efficacy of merging different algorithms to increase detection precision. Additionally, exploring the use of deep learning techniques to enhance the performance of plagiarism detection systems may also be an area of interest. Finally, expanding the scope of plagiarism detection to include other forms of media, such as audio and video, could be another potential direction for future research.

9. REFERENCES

[1] Imam Much Ibnu Subroto and Ali Selamat, "Plagiarism Detection through Internet using Hybrid Artificial Neural Network and Support Vectors Machine," TELKOMNIKA, Vol.12, No.1, March 2014, pp. 209-218.

[2] Upul Bandara and Gamini Wijayrathna ,"Detection of Source Code Plagiarism Using Machine Learning Approach," International Journal of Computer Theory and Engineering, Vol. 4, No. 5, October 2012, pp.674-678.

[3] Salha Alzahrani, Naomie Salim, Ajith Abraham, and Vasile Palade," iPlag: Intelligent Plagiarism Reasoner in Scientific Publications," IEEE World Congress on Information and Communication Technologies, 2011.

[4] Barrón Cedeño, A., & Rosso, "On automatic plagiarism detection based on n-grams comparison," In Advances in Information Retrieval, Vol. 5478. Lecture Notes in Computer Science, pp. 696–700, Springer.