Enhanced Text & Image Plagiarism Detection System

BY

Anupkumar Koturwar
Omkar Angulwar

Under the Guidance

of

Dr. M.Y. Joshi

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Mahatma Gandhi Mission's College of Engineering, Nanded (M.S.)

Academic Year 2024-25

A Project Report on

Enhanced Text & Image Plagiarism Detection System

Submitted to

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE

in partial fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

By

Anupkumar Koturwar
Omkar Angulwar

Under the Guidance
of

Dr. M.Y. Joshi

(Department of Computer Science and Engineering)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MAHATMA GANDHI MISSION'S COLLEGE OF ENGINEERING
NANDED (M.S.)

Academic Year 24-25

Certificate

This is to certify that the project entitled

"Enhanced Text & Image Plagiarism Detection System"

being submitted by ***Mr. Anupkumar Koturwar and Mr. Omkar Angulwar*** *to the Dr. Babasaheb Ambedkar Technological University, Lonere, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by her under my supervision and guidance. The matter contained in this report has not been submitted to any other university or institute for the award of any degree.*

Dr. M.Y. Joshi
Project Guide

Dr. A. M. Rajurkar
H.O.D
Computer Science & Engineering

Dr. G. S. Lathkar
Director
MGM's College of Engg., Nanded

ACKNOWLEDGEMENT

We are greatly indebted to our project guide **Dr. M.Y. Joshi** for her able guidance throughout this work. It has been an altogether different experience to work with her and we would like to thank her for her help, suggestions and numerous discussions.

We gladly take this opportunity to thank **Dr. Rajurkar A. M.** (Head of Computer Science & Engineering, MGM’s College of Engineering, Nanded).

We are heartily thankful to **Dr. Lathkar G. S.** (Director, MGM’s College of Engineering, Nanded) for providing facility during progress of project; also, for her kindly help, guidance, and inspiration.

Last but not least we are also thankful to all those who help directly or indirectly to develop this project and complete it successfully.

With Deep Reverence,

Anupkumar Koturwar (BTech-B)
Omkar Angulwar (BTech-A)

ABSTRACT

This project is a continuation and enhancement of a previously developed text-based plagiarism detection system. In this extended work, significant improvements have been made to develop a more comprehensive and efficient Online Plagiarism Detection System. The major enhancement in this iteration is the integration of image plagiarism detection, enabling the system to analyze both textual and visual content for originality. This addition addresses a growing gap in traditional systems that often overlook visual plagiarism in academic and professional submissions. The system performs real-time plagiarism checks by leveraging web crawling algorithms that scan publicly available online sources for potentially plagiarized material. For textual content, it employs a hybrid approach using cosine similarity, Longest Common Subsequence (LCS), and the Ratcliff/Obershelp algorithm, enabling it to detect exact copying, paraphrasing, and structural rewording. Visual content is extracted from documents and compared using ORB (Oriented FAST and Rotated BRIEF) for keypoint-based matching and SSIM (Structural Similarity Index) for perceptual similarity analysis, ensuring robust image plagiarism detection even in cases of resized, cropped, or edited visuals. The system supports document formats such as PDF, Text and standard image files. Built using Python and Flask, it offers a user-friendly web interface for document submission and generates downloadable reports that highlight plagiarized segments, show source comparisons, and provide detailed similarity scores. By combining web crawling techniques with dual-mode content analysis and automated reporting, this upgraded system significantly enhances the accuracy, scope, and usability of plagiarism detection. It serves as a valuable tool for ensuring originality in educational, creative, and research-based content.

TABLE OF CONTENTS

List of Figures

Chapter 1

Introduction

1.1 Background and Significance of Plagiarism Detection

In the digital age, the ease of accessing and sharing information has revolutionized education, publishing, research, and content creation. The internet has made it possible for individuals to access a vast range of academic and creative materials from across the globe within seconds. While this has significantly improved the efficiency and availability of knowledge dissemination, it has also brought about complex ethical challenges, one of the most prominent being plagiarism. Plagiarism is broadly defined as the act of presenting someone else's work whether in part or in whole as one’s own, without appropriate acknowledgment or citation. It can take various forms, including direct copying, paraphrasing without credit, reuse of images or graphics, and even self-plagiarism, where an author republishes their own previously submitted work without disclosure [1]. As academic institutions and publishing bodies strive to maintain standards of originality and intellectual honesty, the detection and prevention of plagiarism has become a priority.

The significance of plagiarism detection is particularly pronounced in academia. Students, researchers, and scholars are expected to produce original work that contributes to their field of study. However, due to a combination of pressures such as strict deadlines, lack of understanding about citation norms, or deliberate dishonesty, plagiarism has become increasingly common. Left unchecked, it undermines academic integrity, devalues legitimate work, and compromises the credibility of educational institutions [2]. In professional and creative fields, plagiarism is not merely unethical—it can also have legal and financial implications. Copyright infringement can lead to lawsuits, financial penalties, and damaged reputations. In journalism, copying content without permission can result in a loss of readership trust and credibility. Similarly, in fields such as marketing, design, or software development, plagiarism can result in serious intellectual property disputes.

As plagiarism methods have become more sophisticated, the traditional methods of detection such as manual review by educators or editors have proven insufficient. Direct copying is often easy to detect, but cases involving paraphrasing, synonym substitution, sentence restructuring, and visual content reuse are much more challenging. This has necessitated the development of automated plagiarism detection systems that utilize computational algorithms and online search capabilities to evaluate originality with speed and precision [3].

Modern plagiarism detection tools go beyond keyword or phrase matching. They use a variety of techniques such as string-matching algorithms (e.g., Longest Common Subsequence), semantic analysis (e.g., cosine similarity, TF-IDF), and machine learning models to compare documents against large repositories of published material [4]. Some systems are integrated with global academic databases and use automated web crawling tools to retrieve and analyze online content in real-time. Others incorporate image similarity analysis, capable of detecting reused charts, infographics, or diagrams, which are often ignored by conventional systems. An important advantage of online plagiarism detection is its access to an ever-growing corpus of digital content, including websites, blogs, e-journals, whitepapers, and online archives. This allows for real-time validation of originality across billions of web pages. Furthermore, web-based systems are scalable, fast, and do not rely on static local databases that may become outdated. Another significant factor is the integration of user-friendly interfaces into detection platforms, allowing educators, students, content creators, and publishers to upload documents and receive detailed reports that highlight plagiarized sections, provide source links, and generate similarity scores. These reports are instrumental in academic evaluation, editorial reviews, and intellectual property verification [6].

As digital content continues to proliferate, plagiarism detection has emerged as an indispensable component of modern information ethics. The implementation of robust online detection tools not only safeguards the rights of original authors but also fosters a culture of honesty, transparency, and accountability in academia and beyond. With increasing awareness, technological advancement, and institutional support, the role of plagiarism detection will only become more critical in ensuring the authenticity and credibility of knowledge in the digital age. The combination of these methods ensures that the system is capable of detecting plagiarism across a wide range of scenarios, whether it involves exact matches, paraphrased content, or partially copied text. This project represents a step forward in addressing plagiarism detection challenges. Beyond its practical applications, it underscores the importance of using technology to promote ethical practices in academic, professional, and creative domains. By enabling users to verify the originality of their work, this system fosters a culture of integrity and innovation, helping individuals and organizations uphold the values of fairness and intellectual rigor.

1.2 Objectives of the System

The primary objective of the proposed online plagiarism detection system is to create an effective, accurate, and scalable solution that can detect both textual and visual forms of plagiarism from online sources in real time. As the digital ecosystem continues to grow exponentially, the need for tools that can uphold content integrity becomes more urgent. The system aims to bridge the gap between traditional text-based checkers and the increasing complexity of plagiarism practices, especially those involving paraphrased content and reused images [1]. Comprehensive detection of plagiarism in text and images is one of the central goals of the system is to go beyond simple keyword or string matching and incorporate multi-algorithmic approaches to detect varied forms of plagiarism, for text, the system integrates methods such as Longest Common Subsequence (LCS), Cosine Similarity, and the Ratcliff/Obershelp algorithm to ensure both syntactic and semantic matches are captured [2]. For images, the system uses ORB (Oriented FAST and Rotated BRIEF) to detect feature-point similarities, and SSIM (Structural Similarity Index) to analyze perceptual similarity [9][10]. This comprehensive strategy enables the system to catch exact copies, paraphrased segments, structural similarities, and reused visuals that may otherwise go unnoticed.

The system is designed to function entirely online by utilizing real-time web crawling techniques instead of fixed APIs. It automatically navigates and retrieves content from websites, blogs, research articles, and digital libraries. This allows the system to detect copied content effectively using live and continuously updated sources. Unlike static or offline databases that become outdated over time, this crawling-based approach ensures the system remains current and adaptable to the evolving web [4]. Elimination of the limitations associated with offline or static databases that can quickly become outdated. By relying on dynamic online resources, the system remains current and adaptable to the evolving nature of content on the internet. Deliver High Accuracy and Minimize False Positives are plagiarism detection systems must strike a balance between sensitivity (detecting all true cases of plagiarism) and specificity (avoiding false accusations). This system is designed with tunable thresholds and multi-layered scoring to, Ensure high confidence in detected plagiarism. Allow distinction between accidental overlap and deliberate copying. Provide a graded score instead of binary decisions, which offers a more nuanced view of originality [6]. By combining algorithmic outputs and weighting them appropriately, the system enhances its ability to reflect true similarity rather than just surface-level matches.

User-Friendly Interface for Diverse User Groups, to ensure accessibility, the system provides a clean and intuitive web interface built using Flask, HTML, CSS, and JavaScript. Users are able to, Upload documents (PDF, images, or plain text). View real-time analysis with highlights of potentially plagiarized content. Download a detailed PDF report containing scores, sources, and matched sections. This simplicity ensures that even non-technical users such as students, educators, editors, or administrators can benefit from the tool without requiring special training [3]. Generate Detailed, Actionable Reports, another key objective is to produce comprehensive reports that do more than just assign a plagiarism percentage. These reports are, Highlight matched text with reference numbers. Include source URLs for all online matches. Present image comparison scores and URLs of similar visuals found online. Are available in downloadable PDF format, generated using Python's Report Lab library. These structured reports are not only valuable for academic review processes but also serve as documented in disputes or content validation scenarios [5].

Ensure Performance, Scalability, and Responsiveness of the system is built to perform efficiently even under load, using, Thread pooling for parallel crawling and image comparisons. Lightweight file handling to support large PDFs and high-resolution images. Optimized algorithms that return results within seconds. This allows institutions to scale the solution across classrooms or departments without significant infrastructure or performance concerns [8]. Promote Ethical Content Creation Finally, at a broader level, the system aims to foster a culture of originality, academic honesty, and ethical content usage.

By making plagiarism detection accessible, fast, and transparent, it encourages the students to validate their work before submission. Educators to verify assessments with confidence. Authors, designers, and researchers to respect intellectual property. The system becomes not just a gatekeeper, but an educational tool that helps users understand the importance of proper citation, originality, and ethical writing. In the online plagiarism detection system has been thoughtfully designed to deliver a balance of technical sophistication, usability, and ethical impact. By fulfilling these objectives, it addresses a pressing real-world need in academic, creative, and professional domains creation.

1.3 Problem Statement and Research Gap

In the current era of ubiquitous digital access, the ability to copy, modify, and redistribute content has become effortless. While this digital transformation has democratized knowledge and promoted global learning, it has also led to a significant rise in plagiarism across academic, professional, and creative domains [1]. Educational institutions face challenges in evaluating the originality of student work, while publishers, content creators, and researchers struggle to protect intellectual property in an environment where duplication can go unnoticed.

The problem of plagiarism has become increasingly complex. Gone are the days when plagiarism was limited to simple copy-paste operations. Today's challenges include, Paraphrased plagiarism, where ideas are reworded to escape detection. Structural plagiarism, where content is rearranged or partially edited. Image plagiarism, involving the reuse of visual elements such as figures, diagrams, or infographics. Cross-format plagiarism, where content is lifted from websites or digital sources and inserted into academic submissions or reports [4]. Despite these evolving threats, many existing plagiarism detection systems fall short in a number of key areas, revealing a critical gap in current research and practice.

Given the above limitations and the rising complexity of plagiarism in the digital age, there is a pressing need for a comprehensive, real-time, online plagiarism detection system that, Detects plagiarism in both textual and visual content. Utilizes crawling and retrieval techniques to compare against live, global web content. Applies a multi-algorithm approach for more accurate similarity scoring. Offers interactive reporting with detailed highlights and source links. Operates via a user-friendly web interface, accessible to both academic and non-academic users. By addressing this problem, the proposed system aims to make a meaningful contribution to the fields of education technology, digital forensics, and content validation. It fills a crucial gap in existing plagiarism detection tools and paves the way for future research and enhancement, particularly in the integration of machine learning, multilingual detection, and deep semantic analysis.

A review of current plagiarism detection systems and academic literature reveals several key gaps that the proposed project aims to address, Integration of Textual and Visual Plagiarism Detection: Most tools handle either text or image analysis, not both. There is a need for a unified system that can extract and evaluate both textual and image content for originality. Real-Time Online Detection: Existing tools often depend on internal document repositories. There is a gap in the development of systems that actively leverage the vast and dynamic content available on the internet through real-time crawling and retrieval techniques [4]. Combination of Multiple Algorithms: While some systems use cosine similarity or LCS individually, few tools combine multiple algorithms (e.g., LCS, cosine similarity, Ratcliff/Obershelp) in a hybrid model to improve accuracy and robustness. Explainable Detection Results: There is a growing need for systems that not only flag plagiarized content but also highlight it clearly, cite the source, and explain the nature of the match. This is particularly useful in academic and legal contexts. Visual Reporting and User Guidance: Tools should provide clear, structured, and visually intuitive plagiarism reports that help users understand their mistakes and learn from them, rather than just penalizing them [6].

1.4 Scope of the Project

The scope of this project is centered on the development of a comprehensive, web-based plagiarism detection system that functions exclusively in an online mode. This system is designed to analyze both textual and visual content in real time by leveraging the vast resources available on the internet. Its objective is to create a reliable, scalable, and accurate tool that can detect instances of plagiarism by comparing user-submitted documents against publicly accessible online sources. The project explicitly excludes the use of offline databases or local content repositories, focusing instead on dynamic, real-time analysis using automated web crawling techniques. At its core, the system provides plagiarism detection for two major content types: text and images. Text content from uploaded documents is extracted and segmented into manageable chunks, each of which is then searched using real-time crawling and retrieval of content from the web. The retrieved snippets are compared to the original document using multiple algorithms including Longest Common Subsequence (LCS), cosine similarity, and the Ratcliff/Obershelp pattern-matching method. This multi-algorithmic approach ensures that the system is not limited to detecting only exact matches but is also capable of identifying reworded, paraphrased, or structurally altered content [2]. Unlike traditional systems that merely scan for direct copying, this implementation considers the context and structure of the content, resulting in a more nuanced and accurate evaluation. In addition to textual analysis, the system also extends its scope to detect image-based plagiarism, a frequently overlooked but increasingly relevant concern in academic and creative works. Many documents include images such as diagrams, charts, and illustrations that are susceptible to being reused without attribution. The system addresses this by extracting images embedded within submitted PDFs and performing reverse image searches across multiple online platforms, including Google Images, Yandex, Bing, and TinEye. The downloaded image matches are then compared with the original images using two key techniques: the Structural Similarity Index (SSIM) for pixel-level comparisons and ORB (Oriented FAST and Rotated BRIEF) for keypoint-based feature matching [9][10]. These methods enable the system to detect both exact and partial image reuses, even when alterations like resizing, cropping, or minor modifications have been applied. From a technical standpoint, the project supports a wide range of file formats commonly used in academia and publishing. Users can submit files in PDF, DOCX, TXT, JPG, PNG, and JPEG formats. Text content is extracted using libraries such as PyMuPDF (fitz), and image content is identified and saved for visual analysis. The entire process, from file upload to result generation, is conducted via a user-friendly web interface built using Flask and standard web technologies like HTML, CSS, and JavaScript. This ensures that users with limited technical expertise can operate the system effectively without requiring software installation or manual configuration.

The output of the system is a structured and informative plagiarism report. Once the analysis is complete, users receive a detailed report that includes highlighted text segments showing detected similarities, a list of matching online sources with clickable URLs, plagiarism percentage scores, and a breakdown of image similarity results where applicable. The report is generated in PDF format using Python's ReportLab library and is downloadable for academic or professional documentation purposes. This functionality enhances transparency and usability by giving users not just a numeric score, but also meaningful, actionable insight into the nature and extent of the plagiarism detected. Although the current system is focused entirely on English-language content, future enhancements may include support for multilingual detection and semantic analysis using deep learning models. Similarly, while this version is designed for use on desktop browsers, future iterations may incorporate mobile responsiveness or dedicated mobile applications. For now, however, the scope remains clearly defined to ensure a stable and effective online-only solution that performs robust plagiarism detection on both text and images using real-time internet queries and algorithmic comparison. By setting these boundaries, the project avoids the limitations of offline or static detection systems while addressing the real-world challenges associated with identifying both textual and visual content plagiarism. The clearly defined scope ensures that the system is not only practical and implementable but also highly relevant in today’s digital landscape, where originality and authenticity are critical to educational and professional success.

1.5 Methodology Overview

The development of this online plagiarism detection system was guided by a structured, modular methodology aimed at achieving accuracy, efficiency, and user accessibility. The process began with requirement analysis to define the system’s goals: to detect textual and visual plagiarism in real-time using internet-based sources. A web application architecture was selected, with Python and Flask managing the backend logic and HTML/CSS forming the user interface. Users can upload files such as PDFs, DOCX documents, and images, which are then processed to extract both text and embedded visuals. For textual plagiarism detection, the system applies a combination of Longest Common Subsequence (LCS), cosine similarity using TF-IDF, and the Ratcliff/Obershelp algorithm to identify direct copying as well as paraphrased and structurally altered content [2]. These text segments are compared with real-time content retrieved through automated web crawling from online sources.

For visual plagiarism, the system extracts images and submits them to reverse search engines such as Google Images, Bing, Yandex, and TinEye. It then evaluates the similarity of retrieved images using two algorithms: Structural Similarity Index (SSIM) for pixel-based comparison, and ORB (Oriented FAST and Rotated BRIEF) for keypoint-based feature matching [9][10]. All analysis results—textual matches, image similarities, and matched source URLs—are compiled into a downloadable report generated using ReportLab. This report includes highlighted plagiarized content, visual match indicators, and an overall similarity score. The system was thoroughly tested through unit, integration, and system-level testing, with a focus on crawler reliability and consistent performance under varied inputs. This methodological approach ensures the platform is accurate, scalable, and practical for users seeking real-time, web-based plagiarism detection.

Chapter 2

Literature Review

2.1 Existing Text and Image Plagiarism Detection Techniques

According to A. Barrón-Cedeño et al. [1], Plagiarism detection has evolved over the years from manual review and basic software tools to sophisticated systems that leverage algorithms, artificial intelligence, and web-based search technologies. As both academic and professional content increasingly becomes digitized, the challenges associated with ensuring originality have grown in complexity. Today, plagiarism can no longer be confined to exact text duplication. It encompasses paraphrasing, translation, content spinning, and even the unauthorized reuse of visual materials. Consequently, the techniques required to detect such practices have also diversified and advanced. In the context of text-based plagiarism detection, the earliest techniques focused on exact string matching, which compared blocks of text for direct duplication. This method is fast and effective for detecting copy-paste plagiarism, but it fails when the plagiarist alters the sentence structure or uses synonyms. To address these limitations, fingerprinting techniques were developed, which involve creating small, unique hashes (typically of n-grams) that represent a document's content. These hashes are compared across other documents or against an index. While this method improves detection of partial plagiarism, it is still vulnerable to paraphrasing and structural rewriting.

To improve upon these traditional models, semantic similarity techniques such as cosine similarity and TF-IDF (Term Frequency-Inverse Document Frequency) became widely used. These techniques convert documents into vector representations, enabling the system to compute how closely two pieces of text are related in meaning, not just in form. Cosine similarity measures the angle between two document vectors, giving a score that reflects their semantic overlap. This makes it possible to detect content that has been rewritten but still expresses the same ideas. Longest Common Subsequence (LCS) algorithms have also been incorporated into text analysis, especially for identifying sequences that remain unchanged across documents, even when their surrounding context differs. These structural techniques are especially useful in academic writing, where entire paragraphs or data-driven descriptions are often reused with minor edits. According to R. Kumar et al. [2], advanced semantic and structural approaches such as cosine similarity and LCS offer increased detection coverage for paraphrased or transformed plagiarism cases.

Further improvements in accuracy have been achieved using pattern recognition algorithms, notably the Ratcliff/Obershelp algorithm, which evaluates the similarity between strings by identifying matching subsequences and recursively comparing unmatched parts. Although less complex than deep learning models, these algorithms are particularly effective in systems where performance and response time are essential. According to A. Barrón-Cedeño et al. [1], hybrid approach that combines string-based, structural, and semantic analysis to maximize their detection capability. While textual plagiarism detection is a mature area of research, image plagiarism is a relatively recent concern. In academic and professional documents, the inclusion of visual content such as charts, figures, illustrations, and infographics is now common practice. Unfortunately, many detection systems focus exclusively on text, allowing visual plagiarism to go unnoticed. Simple pixel-based image comparison techniques may identify identical copies, but they are highly sensitive to modifications such as resizing, cropping, color correction, and watermarking.

To overcome these limitations, systems now employ feature-based and perceptual similarity measures for image analysis. A widely adopted method is ORB (Oriented FAST and Rotated BRIEF), which detects and describes keypoints within an image and compares them to those in another image. This method is rotation- and scale-invariant, making it useful for identifying modified or partially altered images. SSIM (Structural Similarity Index) is another important metric, which evaluates the perceptual similarity between images based on luminance, contrast, and structural details. According to Z. Wang et al. [9], SSIM aligns more closely with human perception of similarity and is more robust to minor edits. In addition to algorithmic approaches, reverse image search engines like Google Images, TinEye, Bing Visual Search, and Yandex offer practical tools for web-based image plagiarism detection. These platforms analyze an uploaded image and return visually similar images found across the internet, including their source URLs. While primarily designed for manual use, these services can be integrated into automated systems using web crawling and parsed search results, enabling them to function as part of a broader plagiarism detection workflow.

Despite the advancements in both textual and visual plagiarism detection, most existing systems treat these two modalities in isolation. Moreover, many tools rely on static local databases, which cannot reflect the constantly evolving nature of online content. This makes them less effective in detecting plagiarism from blogs, forums, online publications, or academic portals that are updated regularly as a result, there is a growing need for comprehensive, real-time, internet-integrated systems that can handle both text and image plagiarism using modern algorithms and automated web crawling methods. The evolving nature of digital content and the creative ways in which plagiarism is conducted highlight the need for more sophisticated detection systems. The techniques discussed here form the foundation for developing advanced tools, such as the system proposed in this project, which integrates multiple methods to ensure comprehensive detection.

2.2 Advances in Text Similarity Algorithms

As plagiarism has grown more complex, so too have the techniques used to detect it. While early plagiarism detection systems relied heavily on exact string matching and n-gram comparison, modern developments in natural language processing (NLP) and computational linguistics have led to the creation of advanced text similarity algorithms that consider not just surface-level text but also the underlying semantics and structure of language. These advances have enabled systems to identify paraphrased, restructured, and contextually modified content with far greater accuracy than traditional approaches. According to Salha and Abdellaoui [1], one of the most widely used approaches in recent years is the cosine similarity algorithm, particularly in combination with TF-IDF (Term Frequency-Inverse Document Frequency). This method transforms documents into high-dimensional vectors based on the frequency of significant terms, and then computes the cosine of the angle between two vectors to assess similarity. The closer the angle is to zero, the more similar the texts are. Cosine similarity is especially effective in detecting partial matches and paraphrased content where exact wording differs but core terms remain consistent. Unlike string matching, this technique accounts for the importance of individual words in context, thereby increasing the system's sensitivity to meaning rather than form.

Alongside cosine similarity, the Longest Common Subsequence (LCS) algorithm continues to be an important tool in modern plagiarism detection. LCS measures the length of the longest sequence of words that appear in the same order in both texts, regardless of whether they are adjacent. This makes LCS particularly useful in identifying content that has been structurally copied but slightly modified with insertions, deletions, or rephrased clauses. LCS is computationally efficient and provides valuable insights into the structural overlap between two documents, making it a useful supplement to vector-based similarity models.

Another notable development is the application of edit distance algorithms, such as the Levenshtein distance, which calculates the minimum number of single-character edits required to change one string into another. While originally designed for spelling correction, edit distance has found use in detecting near-duplicate sentences and minor content alterations. Similarly, the Ratcliff/Obershelp algorithm, also known as Gestalt pattern matching, detects the longest common substrings and recursively compares the surrounding text. Its recursive nature makes it useful for finding partial or scattered similarities that do not appear sequentially but still represent copied structure or idea.

However, one of the most transformative shifts in the field of text similarity has been the introduction of semantic embeddings and contextual models. According to Mikolov et al. [2], Word embedding techniques such as Word2Vec, GloVe, and FastText represent words as vectors based on the context in which they appear. This allows for the identification of semantically similar words even if they are not exact matches. For instance, doctor and physician or university and college may be treated as similar by embedding models, enabling the detection of paraphrased content that uses synonyms or conceptually related terms.

The rise of transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa has significantly advanced the field of semantic similarity. According to Devlin et al. [3], These models consider the full context of each word in a sentence by examining both its preceding and succeeding words. This deep contextual understanding allows for sentence- and paragraph-level similarity detection that goes beyond mere word-level comparison. Transformer models can detect subtle paraphrasing and shifts in tone or emphasis that earlier models could not. For instance, BERT can understand that the sentences "The study shows a significant increase in cases" and "There has been a notable rise in cases according to the study" carry the same meaning, despite their different syntax and vocabulary.

In addition to identifying semantic similarity, modern algorithms also aim to quantify syntactic similarity—the structural arrangement of words and phrases in a sentence. Techniques such as dependency parsing and POS tagging (part-of-speech tagging) allow systems to analyze the grammatical framework of sentences and identify restructured content that retains the same functional relationships. For example, the active voice sentence “Researchers conducted the experiment” and its passive counterpart “The experiment was conducted by researchers” differ in structure but not in meaning. Systems that account for syntactic variation are far better equipped to handle such cases.

Despite their advantages, the implementation of these advanced algorithms in plagiarism detection systems is not without challenges. High-performance models such as BERT are computationally intensive and may not be suitable for lightweight or real-time applications without significant hardware support. Moreover, the threshold for semantic similarity is often subjective; two texts may be semantically close without being plagiarized, especially if they discuss common knowledge or generic concepts. This creates a need for fine-tuning and interpretability in similarity scoring systems to avoid false positives or unfair accusations.

Nevertheless, the progress made in the field of text similarity algorithms has significantly improved the precision, scope, and fairness of modern plagiarism detection systems. By incorporating both semantic and syntactic analysis, these algorithms provide a multidimensional view of similarity that is capable of uncovering even the most subtly disguised instances of plagiarism. As these technologies continue to evolve, they will play an increasingly critical role in supporting academic integrity, protecting intellectual property, and encouraging original expression across digital platforms.

2.3 Role of the Ratcliff/Obershelp Algorithm in Text Matching

According to Ratcliff and Metzener [1], The Ratcliff/Obershelp algorithm, also referred to as Gestalt Pattern Matching, plays an important role in the field of text similarity and plagiarism detection. Unlike traditional string comparison methods that rely solely on matching fixed-length substrings or exact token sequences, the Ratcliff/Obershelp algorithm is designed to recognize patterns of similarity that reflect how humans perceive textual resemblance. This makes it especially valuable for detecting complex cases of plagiarism, such as partial matches, scattered similarities, and structurally altered but semantically consistent passages. Developed by John Ratcliff and Dave Metzener, the Ratcliff/Obershelp algorithm works by identifying the longest common subsequence (LCS) between two strings and then recursively comparing the unmatched segments that surround the identified subsequence. This process continues until all remaining segments have been analyzed. The final similarity score is computed based on the total length of all matched substrings divided by the sum of the lengths of the two original strings. This approach allows the algorithm to be sensitive to changes in word order and insertion or deletion of phrases, making it well-suited for identifying restructured but related text.

One of the algorithm's greatest strengths lies in its recursive nature, which enables it to detect nested similarities. For instance, in a sentence where certain phrases have been inserted, removed, or swapped, traditional string comparison might fail to identify the underlying commonality. However, the Ratcliff/Obershelp algorithm continues breaking the text into smaller parts and searching for common sequences within them, thereby uncovering deeper structural similarities. This makes it particularly useful in academic writing, where a sentence or paragraph may be reworded extensively while still retaining the original source's conceptual framework. In the context of plagiarism detection, this algorithm is often used in combination with other techniques such as cosine similarity or LCS to provide a hybrid similarity measure. While cosine similarity captures semantic overlap and LCS detects linear structural repetition, Ratcliff/Obershelp focuses on partial matches and displaced fragments. For example, if a student reorders sections of a copied sentence or merges parts of it with their own content, this algorithm can still detect the resemblance that other algorithms may overlook.

Its pattern-based approach simulates the human ability to recognize that two differently worded sentences may share a significant portion of conceptual or structural similarity. Another advantage of the Ratcliff/Obershelp algorithm is its computational simplicity and ease of implementation. Unlike deep learning models that require training on large datasets and consume significant computing resources, this algorithm can be implemented using standard programming libraries. For instance, Python's built-in difflib library uses a variation of this algorithm in its SequenceMatcher class, making it accessible for integration into lightweight and real-time plagiarism detection systems. Its low overhead makes it suitable for web-based applications where fast and scalable performance is critical. However, despite its strengths, the Ratcliff/Obershelp algorithm also has certain limitations. Since it operates at the character or word level, it does not account for semantic meaning. It can identify that the structure of two texts is similar but cannot interpret whether they convey the same idea. As a result, it may miss cases where the language is completely paraphrased but semantically identical. Conversely, it may overestimate similarity between two texts that coincidentally share similar word arrangements without actually plagiarizing content. This limits its standalone reliability and underscores the need for its use as part of a multi-algorithmic detection system.

Furthermore, the algorithm is not robust against synonym replacement or grammatical transformations, both of which are common in sophisticated plagiarism. The student performed the experiment and the pupil conducted the test would be marked as dissimilar by Ratcliff/Obershelp despite having the same meaning. Therefore, its primary utility lies in identifying pattern-level overlap, such as reordered phrases, fragmented quotes, or lightly edited segments, rather than capturing semantic equivalence.

The Ratcliff/Obershelp algorithm serves a crucial supporting role in modern plagiarism detection systems. Its ability to detect partial, recursive, and structurally displaced matches complements other techniques that focus on semantic and linear similarity. When combined with vector-based and syntactic algorithms, it contributes to a robust framework capable of identifying a wide range of plagiarism strategies. Its simplicity, speed, and intuitive pattern recognition make it particularly valuable in online, real-time detection platforms, where a balanced trade-off between precision and performance is essential. As plagiarism techniques continue to evolve, algorithms like Ratcliff/Obershelp will remain an integral part of hybrid detection strategies that aim to replicate human-like judgment in assessing textual similarity.

2.4 Image-Based Plagiarism and Visual Similarity Detection

While text plagiarism has been extensively researched and addressed through a wide array of detection techniques, the challenge of identifying image-based plagiarism has only recently begun to receive the attention it warrants. In modern academic, professional, and digital content, images play a crucial role in communicating ideas, illustrating data, and enhancing understanding. These visual elements may include graphs, charts, infographics, photographs, illustrations, and design layouts. As a result, the unauthorized reuse or slight alteration of such images constitutes a form of intellectual dishonesty that, despite its seriousness, is often overlooked or inadequately addressed by most traditional plagiarism detection systems.

Image plagiarism can occur in several forms. In its most basic manifestation, it involves the direct copying of an image from one source to another without any modification or citation. More subtly, it may involve resizing, cropping, color adjustment, flipping, or the addition or removal of elements such as labels, annotations, or watermarks. In some cases, plagiarists may recreate charts or diagrams from existing sources with minor aesthetic changes to conceal the original reference. These techniques make detection by the human eye difficult, and automated systems must rely on sophisticated algorithms to uncover the original source and assess the degree of similarity between images.

Traditional plagiarism detection tools were not designed to analyze visual data, and as such, they lack the functionality to detect copied or altered images. Early methods that attempted image comparison used pixel-by-pixel matching, which only identifies exact duplicates and fails entirely when even minor changes are applied. For example, altering the contrast of an image or compressing it can render pixel-level comparison ineffective. As a result, more advanced and resilient approaches have been developed to measure visual similarity in ways that are tolerant of transformations and manipulations. Among these methods, ORB (Oriented FAST and Rotated BRIEF) has emerged as a popular and efficient algorithm for detecting image similarities [10]. ORB is a feature-based detection technique that identifies and extracts key points from an image. These key points, or descriptors, are then compared with those from another image to assess the degree of similarity. Unlike pixel-based methods, ORB is robust against changes in orientation, scale, and lighting, making it suitable for identifying copied or modified images even when they are not identical in appearance. ORB is particularly useful for detecting plagiarism in diagrams, logos, and engineering figures where structural elements remain consistent despite stylistic changes.

Another highly regarded method in visual similarity detection is the Structural Similarity Index (SSIM). SSIM measures similarity by analysing luminance, contrast, and structure between two images [9], aligning more closely with human visual perception. It provides a similarity score that reflects how closely two images resemble each other in form and composition, rather than at the individual pixel level. This makes SSIM ideal for evaluating image integrity and detecting subtle reuse of images in academic papers, design projects, and presentations. When used together, ORB and SSIM offer a balanced evaluation—ORB capturing structural patterns and key points, and SSIM capturing perceptual similarities.

Beyond algorithmic analysis, reverse image search engines such as Google Images, Yandex, Bing Visual Search, and TinEye can aid in identifying image plagiarism [4] become essential tools in identifying instances of image plagiarism across the internet. These platforms allow users or automated systems to upload an image and search for visually similar images already published online. The results typically include thumbnails, source URLs, and snippets of the page content where the image was found. Integrating these platforms into plagiarism detection workflows enhances the system's ability to trace plagiarized images to their original sources, especially when those sources are publicly accessible web content.

However, despite these technological advancements, detecting image-based plagiarism still faces several challenges. Unlike text, images lack a standard unit of measurement like words or sentences. An image may convey meaning through layout, design, or graphical representation, which cannot be broken down into discrete elements for direct comparison. Furthermore, many plagiarized images may exist in formats that include embedded text, complex color gradients, or layered components, all of which complicate the analysis. Another limitation lies in the availability of comprehensive image datasets or crawler that allow for automated large-scale visual matching without violating copyright or platform policies.

Moreover, the intent behind visual reuse is often harder to interpret than in the case of textual plagiarism. Two authors may use similar stock images or charts based on publicly available data, which does not necessarily constitute plagiarism unless originality or ownership has been misrepresented. As such, image similarity detection systems must be cautious not to flag false positives and must provide contextual evidence such as the original image source and the extent of visual overlap—for users to make informed decisions. The detection of image-based plagiarism represents an essential yet complex dimension of modern plagiarism detection systems. Algorithms such as ORB and SSIM provide a solid foundation for measuring visual similarity with resilience against modifications, while reverse image search engines offer a powerful way to track the online presence of plagiarized visuals. Although there are challenges in interpreting and automating image-based comparisons, the integration of these techniques into plagiarism detection systems marks a significant advancement in safeguarding creative integrity and academic honesty in a multimedia-rich digital environment.

2.5 Web Crawling Techniques for Plagiarism Source Retrieval

In the context of online plagiarism detection, the ability to accurately retrieve and analyze potential source material from the vast expanse of the internet is crucial. One of the key techniques enabling this capability is web crawling the process of systematically browsing and indexing web pages to extract data or locate content that matches a given query [1]. Web crawling plays a vital role in modern plagiarism detection systems by enabling them to search the internet for previously published content and identify potential matches with submitted documents. This section explores the techniques, tools, and challenges associated with using web crawling for plagiarism source retrieval. Unlike traditional search engines that index a static local corpus or curated academic repository, plagiarism detection systems often require real-time or near-real-time web access to retrieve content that may have been plagiarized from public websites, blogs, forums, or news outlets. Since the web is a dynamic and ever-growing environment, relying on offline databases significantly limits the scope and relevance of source detection. Web crawling bridges this gap by programmatically navigating and extracting relevant data from web pages based on user-submitted text. This allows systems to check for plagiarism against live content rather than outdated or incomplete indexes.

The process typically begins with text segmentation, where the input document is broken down into smaller, manageable units such as sentences or paragraphs. These segments are then used to formulate search queries, which are submitted to online search engines. The results known as snippets are returned along with associated metadata such as the page title, URL, and sometimes a brief description of the matched content. These snippets form the basis of the comparison between the user-submitted content and publicly available web content Depending on the search engine and crawling depth, additional data such as page rank, date of publication, and domain authority may also be extracted to improve result quality. Most modern plagiarism detection systems do not implement full-scale web crawlers due to the complexity, time consumption, and ethical concerns involved in large-scale data collection. Instead, they rely on shallow or focused crawling strategies that selectively fetch pages relevant to user input by leveraging search engine result links [2]. These crawling strategies allow systems to retrieve live web content using tailored search queries and process the resulting pages to extract and compare relevant information. Selective crawling techniques can be configured to target specific domains or avoid irrelevant content, offering fine-tuned control over the scope of material being analyzed. In systems that do require deeper crawling such as when a matched snippet leads to a partial article or paywalled page web scraping techniques may be used. Web scraping involves extracting data directly from the HTML structure of a web page using libraries such as

BeautifulSoup, Selenium, or Scrapy. These tools allow the system to bypass superficial summaries and fetch full-text content for a more accurate comparison. However, this approach must be implemented carefully to respect the terms of service of target websites and avoid legal or ethical violations, particularly when scraping copyrighted or sensitive content. The integration of web crawling or search-based querying into plagiarism detection systems also introduces several technical and performance-related challenges. One such challenge is requesting rate limiting. Many websites and search services impose limits on how frequently automated crawlers can access their pages, which may restrict the system's throughput during high-demand periods. Another challenge is latency since web search and snippet retrieval rely on external servers; network delay can impact the overall speed of plagiarism analysis. Additionally, the system must be equipped to handle incomplete or malformed results, such as broken links, redirects, and truncated snippets that provide insufficient context for accurate similarity computation.

Another concern is the reliability of extracted content. Since web content varies widely in quality and format, systems must include preprocessing techniques to clean and normalize retrieved text before comparing it to the input document. This includes removing HTML tags, JavaScript code, advertisements, and navigation elements that are irrelevant to the plagiarism check. Moreover, the content must be tokenized and vectorized converted into a format suitable for similarity algorithms such as cosine similarity or LCS before it can be meaningfully compared with the original material. To further enhance accuracy, some systems implement chunk-based analysis, in which each sentence or paragraph is queried independently. This increases the likelihood of finding partial matches and allows the system to assign a plagiarism score to individual sections of the document, rather than applying a blanket percentage to the entire text. This granular approach not only improves detection precision but also enables the generation of detailed reports that highlight which parts of the content are matched and which are original, along with corresponding source URLs. It enables systems to remain current, adaptive, and effective against a wide range of plagiarism tactics, including paraphrasing, structural modification, and partial reuse of content from obscure or rapidly updated online sources. As the internet continues to grow and evolve, web-based source retrieval will become even more essential in the development of future-proof plagiarism detection technologies.

Chapter 3

System Design and Architecture

3.1 Architectural Overview of the System

The plagiarism detection system is built with a modular and intelligent architecture that supports comprehensive plagiarism analysis for both textual and visual content. It accepts either raw text input or document uploads (e.g., PDF, image formats), and performs an in-depth comparison using advanced algorithms for both domains. The architecture is designed for scalability, real-time analysis, and seamless user interaction, while preserving the confidentiality of the submitted content.

i. User Interface (UI)

The system provides a web-based user interface that facilitates interaction through a clean and accessible layout.

- **Multi-format Input:** Users can input raw text or upload documents including PDF, TXT, and image formats.
- **Dual Analysis Support:** Automatically triggers both text and image plagiarism detection based on the type of content provided.
- **Progress Feedback:** Displays real-time status of the plagiarism analysis.
- **Results Visualization:** Highlights plagiarized sections of text, shows side-by-side image matches, and provides similarity percentages with source references.
- **Report Export:** Allows users to download a detailed report summarizing both text and image analysis in PDF format.

ii. File Handling and Storage

Upon upload, files are securely handled and temporarily stored for processing. The system supports structured extraction of both text and embedded images. For image-only files, OCR is applied to extract hidden text where applicable. All data is automatically cleared after processing to ensure privacy.

iii. Text and Image Extraction Module

This module is responsible for parsing the uploaded input and distinguishing between textual and image content. Text is extracted from PDFs or DOCX using parsing libraries, while images are isolated from the document structure or directly uploaded. OCR tools are used to handle scanned or image-based text.

iv. Text Plagiarism Detection Engine

The core text analysis module processes the input content using a combination of complementary algorithms, each addressing a different type of textual plagiarism:

- **Ratcliff/Obershelp Algorithm (via Python's difflib library)**: This algorithm detects the longest common subsequences and matches them based on character patterns. It is highly effective in identifying reordered or scattered duplicate fragments, making it ideal for spotting cases where content has been copied but slightly rearranged or partially omitted.
- **TF-IDF Vectorization with Cosine Similarity**: This method transforms textual content into weighted vectors based on term frequency-inverse document frequency (TF-IDF). By comparing these vectors using cosine similarity, the system can detect semantic plagiarism where the structure is changed, but the meaning is preserved through paraphrasing or synonym substitution.
- **Exact String Matching**: For verbatim duplication, the system performs direct comparisons at the sentence or phrase level to catch literal copy-paste instances.

To enable dynamic detection beyond internal comparisons, the system integrates a web crawler that scans publicly accessible content across the internet. It generates segmented queries from the input text and searches online sources for matches.

v. Image Plagiarism Detection Engine

For visual content, this module performs analysis using:

- **ORB (Oriented FAST and Rotated BRIEF):** For key point-based feature matching.
- **BFMatcher:** For matching extracted image descriptors.
- **SSIM (Structural Similarity Index):** For perceptual similarity evaluation.

The system compares extracted images against a crawled dataset of publicly available images and visually similar content. Matched image segments are displayed with percentage scores and reference URLs.

vi. Web Crawling and Matching

A dedicated crawler component dynamically searches the web for both text and image comparisons. It collects relevant content, processes it for indexing, and returns results with strong contextual or visual similarity. This eliminates the need for static datasets and enhances real-time detection accuracy.

vii. **Results Generation and Reporting Module**

After analysis, a comprehensive report is generated containing:

- **Overall Similarity Percentages** for text and image.
- **Color-coded Highlights** of plagiarized text with source attribution.
- **Side-by-side Image Previews** showing matches and similarity scores.
- **Reference Links** to online sources for each matched instance.
- **Final Conclusion** section summarizing the originality status.

Reports are rendered in a modern, print-ready PDF format with a clear structure suitable for academic or professional review.

viii. **Backend Processing and Coordination**

The backend layer coordinates the flow between modules: input handling, text/image preprocessing, crawler operations, analysis execution, and report generation. It ensures fault tolerance and efficient task execution across various user requests.

ix. **Security and Privacy Layer**

Security is embedded throughout the architecture. All uploaded files and results are handled securely. User data is encrypted during transmission and erased after report generation to ensure confidentiality and compliance with data protection norms.

x. **Technology Stack Overview**

- **Frontend:** HTML, CSS, JavaScript (with optional framework support such as React or Vue.js).
- **Backend:** Python with Flask for routing and processing.
- **Text Processing:** PyPDF2, sklearn (TF-IDF), difflib, NLTK.
- **Image Analysis:** OpenCV (ORB, BFMatcher), SSIM, Tesseract OCR.
- **Crawling & Matching:** Custom web crawler for online content retrieval.
- **Report Generation:** HTML templating with PDF rendering tools
- **Security:** SSL for secure transfer, AES for encrypted temporary storage.

3.2 Detailed Workflow and Processing Steps

The plagiarism detection system follows a systematic and multi-stage workflow to analyze both textual and visual (image-based) content for originality. The design ensures that users whether students, educators, content creators, or professionals—can submit their work in various formats and receive a comprehensive plagiarism report that evaluates and highlights duplicated material from online sources. The system supports two primary input types: direct text input and file uploads (including PDFs and image files). Based on the input, the workflow branches into text analysis, image analysis, or both. This section outlines the complete end-to-end processing pipeline from user interaction to report generation.

Input Acquisition Stage; The system begins by collecting user-submitted content. This can occur through two main interfaces:

- **Text Input Interface**: The user directly pastes raw text into a web-based editor.
- **File Upload Interface**: The user uploads a file (e.g., PDF, TXT, JPEG, PNG). The interface accepts both text-based and image-based documents.

All uploads are temporarily stored in an encrypted local repository, ensuring confidentiality during processing

Content Type Detection and Extraction; Once input is received, the system determines the nature of the content:

- **If the input is raw text**, it is passed directly to the text processing pipeline.
- **If the input is a file**, the system performs format detection:
  - o **PDF/ TXT**: Uses libraries like PyPDF2 or docx to extract structured textual content.
  - o **Image files**: Apply Tesseract OCR to extract any embedded or scanned text.
  - o **All embedded images** (from PDFs) are also extracted separately for plagiarism analysis.

To prepare the data for accurate analysis, both the textual and visual content undergo extensive preprocessing.

i. **Text Preprocessing**

- **Tokenization**: Breaks text into words, sentences, and tokens.
- **Stopword Removal**: Eliminates common words (like "is", "the", "of") that do not contribute meaningfully to plagiarism detection.
- **Normalization**: Converts all text to lowercase, strips punctuation, and handles formatting inconsistencies.
- **Stemming/Lemmatization**: Reduces words to their base form for better matching.
- **Cleaning**: Removes headers, footers, extra line breaks, and non-content elements from extracted files.

ii. **Image Preprocessing**

- Converts images to grayscale to standardize pixel data.

- Resizes and crops where necessary to maintain aspect ratio.
- Applies feature extraction tools like ORB (Oriented FAST and Rotated BRIEF) to capture key visual elements.

This preprocessing ensures that both paraphrased text and visually altered images can still be flagged during analysis

iii. Web Crawling for Online Matching

The system then initiates custom crawler-based web exploration to compare extracted content with publicly available sources.

- **For text**: The system generates query from preprocessed text and searches across online academic, news, blog, and reference websites.
- **For images**: ORB descriptors are compared against web-hosted visuals by crawling indexed media content repositories, infographics, educational diagrams, and more.

The crawler indexes and caches key results from various URLs for real-time referencing and comparison.

iv. Plagiarism Analysis Technique

The plagiarism analysis engine operates in two parallel modules—one for text and one for image content. In the text analysis module, the system first applies exact string-matching using Python's difflib library to detect direct duplications. It then uses the Ratcliff/Obershelp algorithm, also implemented through difflib, to identify partially copied or reordered segments of text. To detect paraphrased content, the system uses TF-IDF vectorization combined with cosine similarity, allowing it to measure semantic similarity even when the original wording is changed. Each algorithm produces a similarity score, and a weighted average is calculated to determine the overall text plagiarism percentage. Matched content is highlighted and linked to its source in the final report.

For image analysis, the system uses ORB to extract keypoints and features from input images, followed by BFMatcher to compare these with visuals found online. Structural similarity is further evaluated using SSIM, which helps detect modified or partially altered images. Each image match is scored, and the top results are displayed side-by-side in the report along with similarity percentages and source links. This dual-module approach ensures accurate detection of both text and image plagiarism.

v. Match Highlighting and Source Mapping

Once analysis is complete, the system:

- Maps plagiarized text back to the original input with highlights.
- Annotates each image match with:
    - Source link
    - Matched region
    - Similarity score

This phase makes the results easy to interpret by showing what was plagiarized, how it was plagiarized, and where it came from.

vi. Report Generation and Rendering

A detailed report is generated in PDF format, which includes:

- Overall similarity percentages for both text and image content.
- Tables summarizing top matching sources and their respective scores.
- Text highlights with side-by-side original source excerpts.
- Image match previews, side-by-side, with reference URLs and similarity metrics.
- Pie charts or graphs for visual breakdown of originality vs plagiarism.
- Conclusion section stating whether the document is original or contains plagiarized sections.

The PDF is styled with clear headings, table of contents, proper formatting, and professional layout suitable for academic and institutional use.

vii. Post-Processing and Cean-Up

To maintain system security and performance:

- Uploaded documents and extracted data are automatically deleted after the report is generated.
- Temporary metadata is cleared from memory.
- The user is notified that their data was processed securely and no content has been retained.

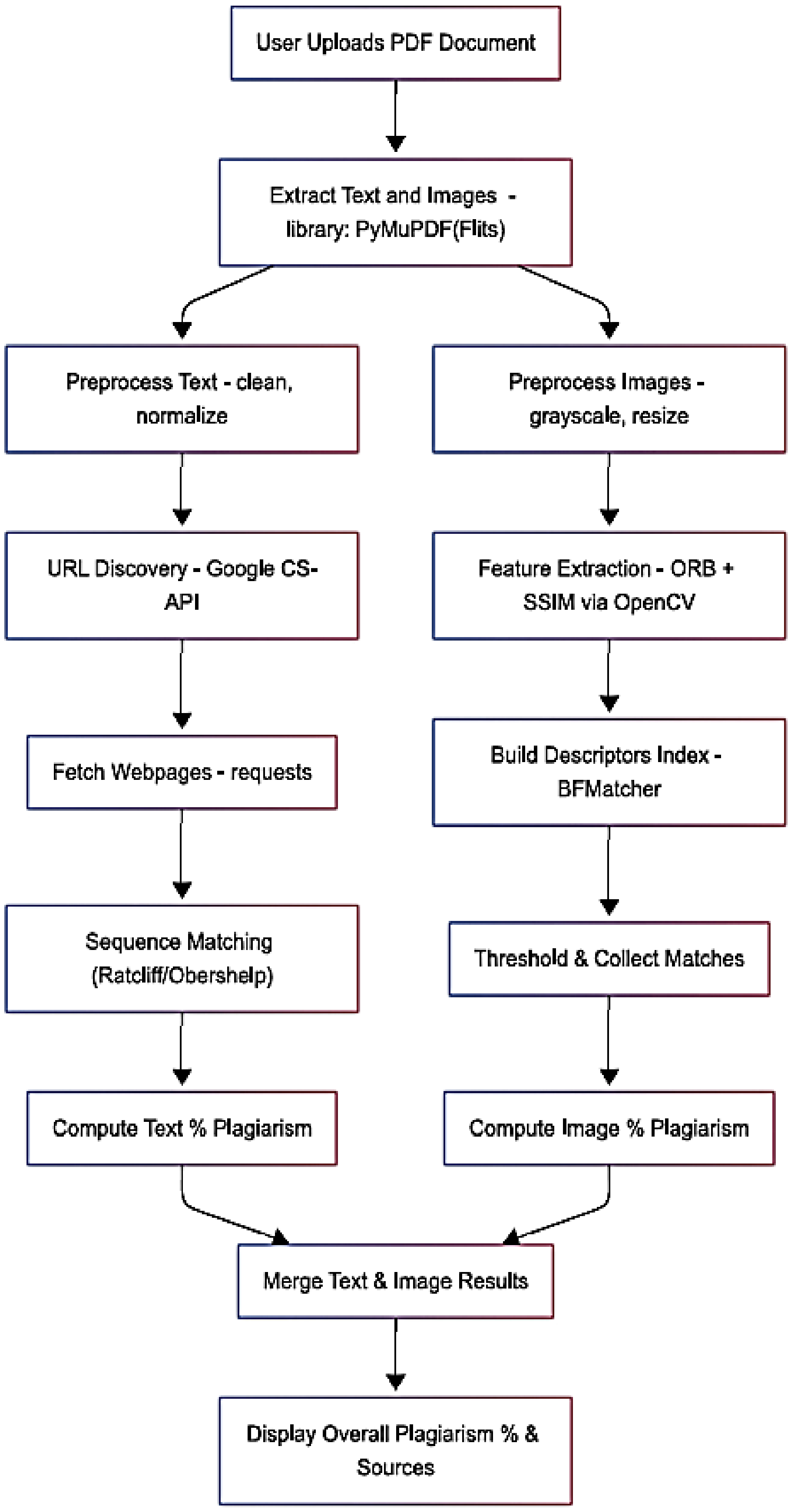

User Uploads PDF Document
Extract Text and Images - library: PyMuPDF(Flits)
Preprocess Text - clean, normalize
Preprocess Images - grayscale, resize
URL Discovery - Google CS-API
Feature Extraction - ORB + SSIM via OpenCV
Fetch Webpages - requests
Build Descriptors Index - BFMatcher
Sequence Matching (Ratcliff/Obershelp)
Threshold & Collect Matches
Compute Text % Plagiarism
Compute Image % Plagiarism
Merge Text & Image Results
Display Overall Plagiarism % & Sources

Fig. 3.1 Workflow Diagram

3.3 Practical Approach to System Development

The implementation of the plagiarism detection system followed the System Development Life Cycle (SDLC) to ensure a structured, phased, and goal-oriented approach. Each stage played a critical role in building a reliable, efficient, and user-focused platform capable of detecting both text and image-based plagiarism.

i. Requirement Analysis

The development process began by identifying core functional requirements. These included support for direct text input and file uploads, ability to analyze both textual and visual content, real-time online comparison using web crawling, and generation of detailed, user-friendly plagiarism reports. Feedback was gathered from academic users, including students, teachers, and researchers, to understand practical needs and common challenges. This helped define the system's scope and establish its key objectives: accuracy, speed, usability, and comprehensive detection.

ii. System Design

A modular and scalable architecture was designed to separate concerns across different layers of the system. The design included distinct components for the frontend UI, backend processing, text and image extraction, plagiarism detection logic, and report generation. The system design also included workflow diagrams to map the step-by-step flow from user input to report output. Care was taken to ensure that both text and image processing pipelines could operate in parallel without interfering with each other.

iii. Implementation

The system was implemented using Python for the backend, specifically using the Flask framework, and HTML, CSS, and JavaScript for the frontend. Text processing modules handled input sanitization and extraction, while image analysis modules utilized ORB, BFMatcher, and SSIM. The core text plagiarism detection used Ratcliff/Obershelp (via difflib), TF-IDF cosine similarity, and exact string matching. A custom-built web crawler retrieved and compared content from online sources. File handling supported PDF, TXT, and image formats, while OCR was used to extract text from scanned files.

iv. Testing

Multiple levels of testing were conducted to ensure reliability. Unit testing verified each module independently—such as text extraction, similarity scoring, and image matching—while integration testing ensured smooth interaction between components. Test scenarios included uploading different file types, analyzing large documents, handling image-only files, and verifying accurate plagiarism detection in both obvious and paraphrased cases. The web crawler's ability to return valid online matches was also validated.

v. Deployment

The completed system was deployed to a secure web server, making it accessible through modern web browsers. Scalability was configured to support multiple concurrent users, and SSL encryption was implemented to protect user data during upload and retrieval. Backend processes were optimized for responsiveness even when processing larger documents. The deployment setup ensures compatibility with institutional environments such as schools or universities.

vi. Maintenance and Updates

Following deployment, the system continues to be monitored for performance, accuracy, and user feedback. Updates are planned to enhance algorithmic performance, improve matching precision for paraphrased and translated text, and optimize image detection accuracy. Future development includes introducing machine learning models to better detect complex textual modifications and adding support for more file types and languages.

3.4 Functional Requirements

The This section outlines the core functional requirements of the plagiarism detection system. Each function is designed to handle specific stages of the detection process, from content input to comparison, analysis, and final report generation.

i. Document Input and Storage

The system must allow users to submit documents either through direct text input or by uploading files in formats such as PDF, TXT or image formats (JPG, PNG). Upon submission, documents are temporarily stored for processing.

Functionality: Accept and store documents for further analysis. Each file is assigned a unique ID and metadata such as filename and type.

Inputs: Document text (from input field) or file (uploaded)

Outputs: Confirmation message after upload and temporary storage

ii. Text and Image Preprocessing

The system must preprocess the input content to prepare it for accurate comparison. For text, it performs normalization tasks such as lowercasing, tokenization, and stopword removal. For image input, it extracts text using OCR (if applicable) and isolates embedded images for visual analysis.

Functionality:

- **Text**: Clean and tokenize input for better semantic and lexical comparison
- **Image**: Apply OCR to extract text; isolate images for similarity analysis

Inputs: Uploaded text or document

Outputs: Cleaned, normalized text; extracted image segments

iii. **Plagiarism Detection (Text and Image)**

The system compares the processed input against a dynamically retrieved set of online sources using crawler-based content collection. Text comparison is performed using Ratcliff/Obershelp, TF-IDF + cosine similarity, and string matching. Image comparison is done using ORB, BFMatcher, and SSIM.

Functionality:

- Perform multi-method similarity comparison for text
- Match extracted images with online visuals
- Generate similarity scores for both content types

Inputs: Preprocessed text and extracted image data

Outputs: Similarity scores and detected match information for each content segment

iv. **Similarity Reporting and Visualization**

After detection, the system generates a comprehensive similarity report. The report includes overall plagiarism percentages, source URLs, highlighted plagiarized text, matched images with similarity scores, and a final originality conclusion.

Functionality:

- Present detailed comparison results for both text and images
- Display source-wise match percentages and references
- Allow report download in PDF format

Inputs: Similarity results from text and image analysis

Outputs: User-friendly report containing plagiarism highlights, matched sources, percentage scores, and downloadable summary

3.5 Non-functional Requirements

i. **Performance**

The system must deliver fast and reliable processing regardless of the size of the input document or the number of concurrent users. It should efficiently handle text extraction, image parsing, and plagiarism analysis with minimal latency. Real-time feedback should be provided to users, especially for short- to medium-length documents, ensuring that response times remain within acceptable limits under typical usage loads.

ii. Security

To protect sensitive data, the system must implement strong security measures. Uploaded documents and input text should be handled confidentially, with temporary storage and automatic deletion post-analysis. SSL encryption must be used for all user interactions, and the system must include access control to prevent unauthorized use. Sensitive operations, such as crawling and result generation, should be isolated from direct client access.

iii. Usability

The interface must be simple and intuitive, allowing users to easily upload documents or enter text for analysis. Navigation, file selection, and result viewing should require minimal technical knowledge. Results must be presented clearly, with properly formatted similarity percentages, matched content highlights, and visual cues for interpreting plagiarism detection results in both text and images.

iv. Scalability

The system must be designed to scale effectively as the number of users increases. It should support multiple concurrent plagiarism checks without performance degradation. Backend modules should be optimized for lightweight, stateless processing to facilitate deployment on scalable infrastructure (e.g., containers or cloud services), enabling horizontal scaling when necessary.

v. System Architecture

The system uses a client-server architecture, where users interact through a web-based client interface to submit input. All heavy computations, including OCR, feature matching, and vector similarity calculations, are performed on the server side to reduce client-side load and ensure consistency.

3.6 Use Case Diagram

Actor User: A User interacts with the system to check for plagiarism in their documents. Their primary goal is to upload a document, check for plagiarism, and view the results in the form of a report.

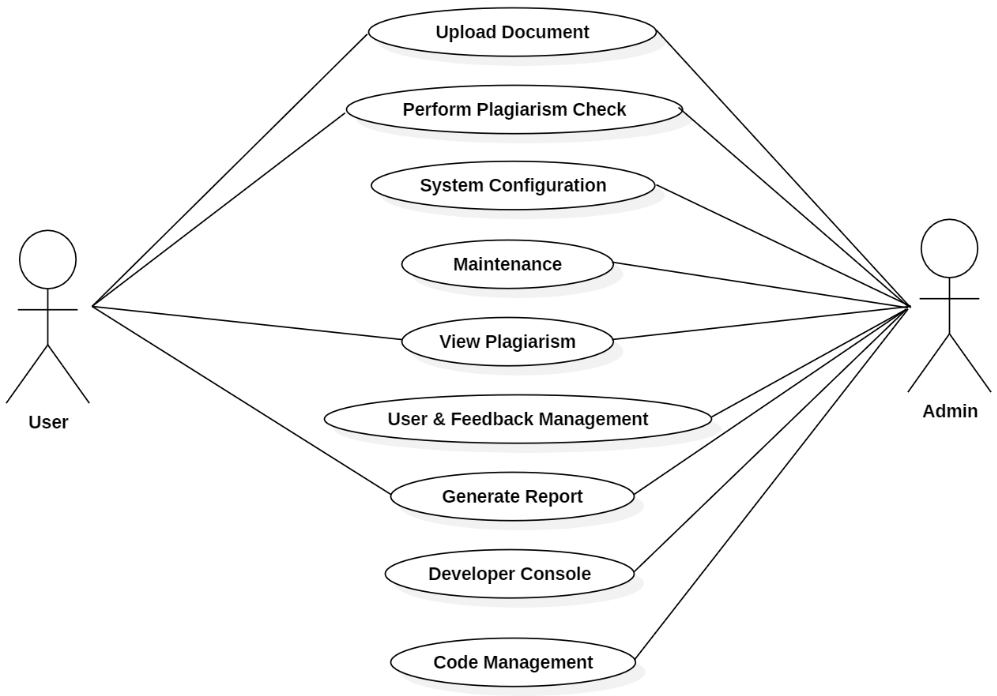


Fig.3.2 Use case Diagram

i. **Upload Document:** the user can upload a document that they wish to check for plagiarism. The system will accept the uploaded file, which will then be used in the plagiarism checking process.

ii. **Perform Plagiarism Check:** Once the document is uploaded, the system will automatically perform a plagiarism check. This involves comparing the uploaded document to a database of other documents (or web content) to detect similarities or exact matches.

iii. **Generate Report:** After the plagiarism check is complete, the system generates a plagiarism report. This report includes details such as the percentage of plagiarized content and possibly a list of sources from which the plagiarized content was detected.

iv. **View Plagiarism Report:** The user can then view the plagiarism report. This is an essential feature that provides the user with the results of the plagiarism check, which they can use to make revisions to their document if necessary.

v. **Relationships (Arrows):** Solid arrows represent direct interactions between the user and system functions, such as uploading a document. Dashed arrows indicate the flow between use cases e.g., uploading triggers plagiarism checking, which then leads to report generation.

Chapter 4

Implementation

4.1 Development Tools and Environment

The plagiarism detection system was developed using a combination of robust backend and frontend technologies to ensure accuracy, performance, and ease of use. Python was selected for the backend due to its flexibility and comprehensive ecosystem for text and image processing. The Flask framework was used to build RESTful endpoints and handle user interactions, while HTML, CSS, and JavaScript formed the core of the frontend interface. Libraries such as difflib (for Ratcliff/Obershelp string matching), Scikit-learn (for TF-IDF and cosine similarity), OpenCV (for ORB-based image feature detection), and Tesseract OCR (for text extraction from image files) were integrated to implement the main detection modules. Report generation was handled using Jinja2 templates rendered into PDF format via WeasyPrint. Development took place in Visual Studio Code on Windows, with additional testing on Linux environments to ensure deployment flexibility and cross-platform compatibility.

The system architecture emphasizes modularity and extensibility, supporting both textual and visual plagiarism detection with real-time comparison capabilities. Instead of relying on static datasets or limited APIs, a custom-built crawler was used to dynamically search and retrieve online content for comparison, improving coverage and accuracy. This approach is supported by current research trends in plagiarism detection, which highlight the advantages of hybrid models and crawler-based online validation techniques for both text and image content [4].

The development of the plagiarism detection system utilized a robust stack of tools, libraries, and technologies to ensure efficiency, scalability, and accuracy:

i. **Programming Language**: Python
ii. **Libraries and Frameworks**:
    - **PyPDF2**: For extracting text from PDF files.
    - **nltk**: For tokenizing text and removing stopwords.
    - **sklearn**: For implementing Cosine Similarity.
    - **difflib**: For performing String Matching.
    - **requests**: For integrating the Google Custom Search API.
iii. **Integrated Development Environment (IDE)**: PyCharm, Visual Studio Code

iv. **System Environment**:

   a. Operating System: Windows 10

   b. Hardware Requirements:

      i. Processor: Dual-core 2.5GHz or higher

      ii. RAM: 4GB or more

      iii. Disk Space: 1GB

4.2 Text Plagiarism Detection Using Ratcliff Algorithm

The Ratcliff/Obershelp algorithm, also referred to as "Gestalt Pattern Matching," is a recursive pattern recognition algorithm that compares two sequences (typically strings) by identifying the longest common subsequence (LCS) and then analysing the surrounding unmatched parts. Unlike semantic similarity techniques, which look for meaning-level equivalence, the Ratcliff algorithm focuses purely on exact character sequence alignment, making it extremely effective for detecting verbatim copying, partial plagiarism, reordered phrases, and structural reuse. This makes it particularly well-suited to plagiarism detection tasks, especially in educational or content validation scenarios where rewording is minimal.

The algorithm is recursive in nature. It begins by locating the longest matching substring shared between two inputs. Once found, it splits both strings into three parts: the text before the match, the match itself, and the text after the match. The algorithm is then applied recursively to the "before" and "after" segments. All identified matches are aggregated, and a similarity score is calculated based on the total number of matching characters across all matched blocks. The core idea is that plagiarism often involves reuse of structure and phrase fragments, even if the surrounding content changes.

The formal calculation used in this system to compute similarity based on the Ratcliff algorithm is:

$$SimilarityRatio = \frac{(2 \times M)}{(|A| + |B|)}$$

Where:

M = Total number of characters from all matched blocks

|A|= Length of the input string

|B|= Length of the comparison string

The result is a floating-point ratio between 0.0 and 1.0, which is then multiplied by 100 to get the percentage similarity. A ratio of 1.0 (or 100%) indicates perfect character-level duplication, while 0.0 indicates no match at all.

In-Depth Example

Input Text (User Submission):

"The solar system has eight planets that revolve around the sun in elliptical orbits."

Source Text (Suspected Plagiarism):

"There are eight planets in our solar system that move around the sun along elliptical paths."

Even though these sentences are structured differently and contain some different words, there are **several phrase-level similarities** that can be detected using the Ratcliff/Obershelp algorithm.

Let's examine the matches step by step:

Step 1: Identify Longest Common Substrings

From the input and source texts, the algorithm detects the following valid matches:

- "Eight planets" – 13 characters (exact match)
- "Solar system" – 13 characters (appears in both, but reordered)
- "elliptical" – 10 characters (shared root word)

Phrases like "revolve around the sun" vs. "move around the sun" are not counted, as they are not exact matches. Common words like "the" and "around" are ignored due to being shorter than the minimum threshold (10 characters).

Step 2: Matched Blocks Summary

Match No.	Matched Block	Length	Appears In
1	eight planets	13	Both
2	solar system	13	Both
3	elliptical	10	Both

Total Matched Characters (M):

13 (block 1) + 13 (block 2) + 10 (block 3) = 36 characters

Step 3: Length of Texts

- Length of Input Text (|A|) = 86 characters
- Length of Source Text (|B|) = 87 characters

Step 4: Apply the Similarity Formula

Similarity Ratio = (2 × Matched Characters) ÷ (Length of Input + Length of Source)

Similarity Ratio = (2 × 36) ÷ (86 + 87)

Similarity Ratio = 72 ÷ 173

Similarity Ratio ≈ 0.4168

Step 5: Convert to Plagiarism Percentage

Plagiarism Percentage = 0.4168 × 100

Plagiarism Percentage ≈ 41.68%

After aggregating all valid matching blocks using the Ratcliff/Obershelp algorithm, the system computes two types of similarity measures: a general similarity ratio between both texts, and a plagiarism percentage that reflects how much of the input text has been duplicated. While the similarity ratio considers both input and source lengths, the plagiarism percentage focuses only on the input, using the following formula:

$$\text{Plagiarism \%} = \left(\frac{\text{Total Matched Characters}}{\text{Total Input Length}}\right) \times 100$$

This helps answer the question: "How much of the student’s content is not original?"

For example, in the previous case:

- Total Matched Characters = 36
- Total Input Length = 86

So,

$$Plagiarism\ \% = \left(\frac{36}{86}\right) \times 100 \approx 41.86$$

This score indicates that 41.86% of the user-submitted content is directly copied from one or more sources, which may be interpreted as significant partial plagiarism in academic or professional contexts. This approach aligns better with real-world expectations, where the originality of the submitted content is prioritized over the structure of the source.

Recursive Nature of the Algorithm

After identifying each matched block, the algorithm recursively examines the text before and after each matched segment. This ensures even fragmented or disjointed copying is detected. For instance, if "eight planets" appears in both the beginning of one text and the end of another, the algorithm still detects it.

This is particularly useful in academic plagiarism, where students often rearrange sentence order or interleave copied material with original content. While semantic algorithms may fail to catch this unless the meaning is preserved, the Ratcliff algorithm will detect literal reuse regardless of position.

Advantages in Plagiarism Detection

- Detects reordered phrases and reused structure
- Requires no language modelling, making it fast and language-agnostic
- Finds partial, disjointed matches missed by full-string matchers
- Efficient and suitable for real-time systems.

4.3 Web Crawling-Based Source Matching

The plagiarism detection system integrates a custom web crawling mechanism that performs live content retrieval from the internet to identify potential matches for both text and image inputs. Unlike traditional methods that rely on pre-stored databases or third-party APIs, this approach dynamically searches publicly available online sources at the time of analysis, ensuring up-to-date and comprehensive comparison coverage.

When the user submits content either as raw text, a document, or an image the system begins by segmenting the input into meaningful parts. In the case of textual input, it is cleaned, normalized, and broken into query-like snippets. These snippets are used to generate targeted search patterns. The crawler then navigates the open web, accessing pages that are likely to contain similar content. Each retrieved page is parsed to extract raw text content. This extracted text is then compared with the user's input using a series of similarity algorithms, including Ratcliff/Obershelp and TF-IDF cosine similarity. Matching blocks are isolated, scored, and stored for final reporting.

For image-based input, the system extracts key visual features using ORB (Oriented FAST and Rotated BRIEF) and converts them into descriptors. The crawler is configured to scan media-rich web pages, extract embedded image data, and compute feature vectors for all discovered visuals. These vectors are then compared against the input image using Brute-Force matching and Structural Similarity Index (SSIM), allowing the system to identify even partially altered or compressed duplicates.

This crawler-based approach ensures that the system can detect plagiarism from the latest content available on the internet—including blogs, academic sites, tutorials, online books, and unindexed web pages. It continuously refines matches in real time and supports both exact and approximate detection. As a result, users receive a comprehensive list of matched sources, even if the content was uploaded minutes before the analysis. By using a focused and recursive crawling strategy, the system maximizes precision while minimizing unnecessary bandwidth consumption, maintaining speed and reliability throughout the process.

4.4 Image Plagiarism Detection Using ORB and SSIM

In addition to textual similarity detection, this system also supports advanced image plagiarism detection using a combination of ORB (Oriented FAST and Rotated BRIEF) and SSIM (Structural Similarity Index Measure). This dual-method strategy allows the system to detect direct image copies, partial crops, resized or rotated visuals, and even compressed or color-adjusted duplicates, making it highly adaptable to real-world plagiarism attempts in academic and creative work. The detection process begins by extracting images from uploaded files. For example, if a PDF contains diagrams or screenshots, these images are isolated and processed individually. Each image is first converted to grayscale for uniformity, then analyzed in two stages: feature matching using ORB, and visual similarity scoring using SSIM.

ORB Algorithm Explanation

ORB is a feature-based algorithm used for key point detection and description. It works in two parts:

i. **FAST (Features from Accelerated Segment Test)** — Detects corner-like regions (key points) in the image.
ii. **BRIEF (Binary Robust Independent Elementary Features)** — Converts local patches around each key point into compact binary descriptors.

These descriptors are:

- Scale-invariant: unaffected by resizing
- Rotation-invariant: resistant to angle changes
- Efficient: fast enough for real-time comparison

To compare two images, ORB detects keypoints in both and then uses a Brute-Force Matcher to find matching descriptor pairs based on Hamming distance.

SSIM Explanation

SSIM measures perceptual similarity between two images by comparing:

- Luminance (brightness)
- Contrast
- Structural content (edges, shapes, spatial patterns)

SSIM returns a score between 0 and 1:

- 1 = Identical images

- 0 = Completely different images

This makes it ideal for detecting compressed, blurred, or lightly altered image plagiarism, even when ORB features may not align.

Step-by-Step Example

Input Image:

A bar chart showing "Annual Sales from 2020 to 2024" in a blue-themed rectangular design.

Plagiarized Image (Found Online):

Same chart, but:

- Cropped to remove title
- Rotated by 90°
- Color changed to green
- Slightly blurred for compression

Step 1 – ORB Feature Matching:

- Input image: 250 key points detected
- Source image: 240 key points detected
- ORB finds 120 matching key points
- ORB Match % = (120 / 250) × 100 = **4**8%

Step 2 – SSIM Calculation:

- Despite rotation and blur, the structural pattern (bar heights, spacing) remains the same.
- SSIM = **0.84** → strong perceptual match

Step 3 – Final Similarity Score (Weighted Average):

You may use a simple formula like:

Final Image Plagiarism % = (ORB Match % + (SSIM × 100)) / 2

= (48 + 84) / 2

= 66%

This score shows moderate to high plagiarism, despite surface-level changes to the image.

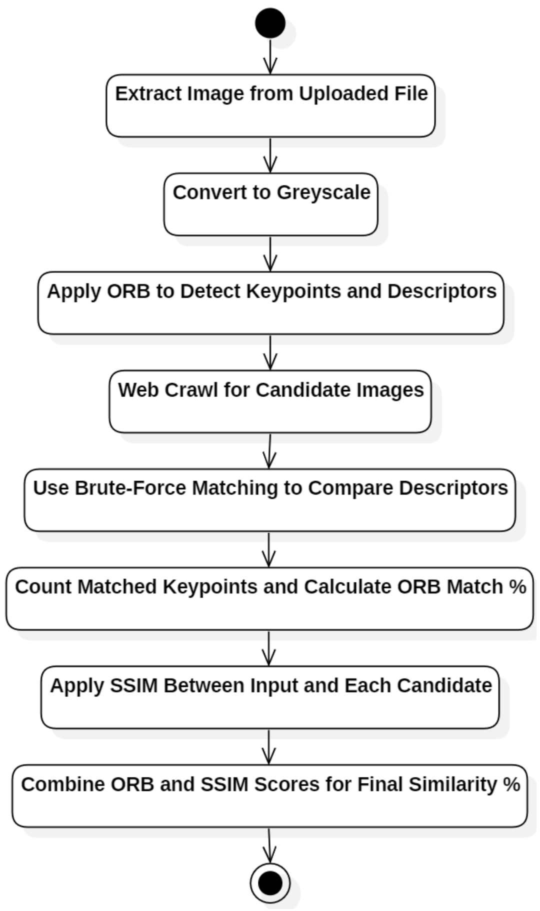

Extract Image from Uploaded File
Convert to Greyscale
Apply ORB to Detect Keypoints and Descriptors
Web Crawl for Candidate Images
Use Brute-Force Matching to Compare Descriptors
Count Matched Keypoints and Calculate ORB Match %
Apply SSIM Between Input and Each Candidate
Combine ORB and SSIM Scores for Final Similarity %

Fig.4.1 Flowchart: Image Plagiarism Detection

Why This Dual Approach Is Effective

- ORB is great for precise structural reuse — even when size or orientation changes
- SSIM captures visual similarity from a human perception standpoint — useful for compressed or recolored images
- Together, they detect both technical and visual forms of plagiarism
- Works on charts, illustrations, design assets, photos, and other embedded images

By combining feature-based detection (ORB) with perceptual analysis (SSIM), the system can detect a broad spectrum of image plagiarism techniques, from direct copying to cleverly disguised reuse. This ensures that diagrams, charts, and visuals in submitted work are held to the same originality standards as text content, helping institutions and evaluators enforce academic integrity across all media types.

4.5 PDF Processing: Combined Text and Image Detection

To ensure a comprehensive analysis of content originality, the plagiarism detection system is equipped to handle PDF files by extracting and processing both text and embedded images from them. PDF files are widely used for academic submissions and research publications, and they often contain a mixture of textual content, figures, tables, charts, scanned pages, or screenshots. The system follows a multi-stage PDF processing pipeline that ensures all relevant information is extracted and evaluated for potential plagiarism.

Text Extraction from PDF

The text extraction process begins by identifying whether the PDF contains machine-readable text or image-based pages (such as scanned documents). For PDFs with selectable or digital text, tools like PyPDF2 are used to extract text page by page. The system preserves line breaks, spaces, and layout integrity while removing redundant formatting characters.

Once the raw text is extracted, it undergoes a text preprocessing phase where it is:

- Lowercased to standardize all content
- Tokenized into sentences and words
- Cleansed of punctuation and special characters
- Stripped of stop words (e.g., "and", "the", "is") for efficient comparison

This preprocessed text is then passed to the text plagiarism detection engine, which compares it using Ratcliff/Obershelp, TF-IDF cosine similarity, and exact string-matching techniques. The matched segments are recorded along with their original positions in the document so that they can be highlighted in the final report.

Image Extraction from PDF

Simultaneously, the system extracts all embedded images from the PDF using tools such as PyMuPDF or pdf2image. These images could be diagrams, charts, graphs, or even photos of handwritten material. For each page, the system scans for:

- Inline vector graphics
- Embedded raster images (JPG, PNG)
- Entire page screenshots (common in scanned PDFs)

Extracted images are preprocessed through the following steps:

- Converted to grayscale to standardize input format
- Resized and padded for consistent analysis size
- Run through Tesseract OCR to extract any embedded text for dual analysis
- Passed to the image plagiarism detection engine

In the image analysis pipeline, each image is analyzed using ORB for feature extraction and SSIM for perceptual similarity. Descriptors and key points are generated, compared against online crawled visuals, and assigned match scores. These are used to compute the final image plagiarism percentage, which is displayed along with visual previews in the generated report.

Combined Reporting and Interpretation

After both the text and image pipelines are completed, the results are merged. The system generates a detailed report that includes:

- Overall plagiarism percentage (combined text and image scores)
- Text sections with highlighted matches and cited sources
- Side-by-side comparisons of matched images with similarity percentages
- Source links for both content types
- A final conclusion indicating the originality status of the PDF

This combined processing ensures no part of the document—neither visible text nor embedded visual content—is overlooked. It strengthens the system's ability to detect both direct and hidden plagiarism, particularly in complex academic submissions that mix text with technical illustrations or scanned references, compared against online crawled visuals, and assigned match scores. These are used to compute the final image plagiarism percentage, which is displayed along with visual previews in the generated report.

4.6 Generating Result Reports with Highlighted Matches

Once the text and image plagiarism detection processes are complete, the system compiles the results into a comprehensive, user-friendly report that presents both statistical findings and detailed visual evidence. This report is designed not only to summarize the originality score but also to clearly identify the exact portions of the content that were matched with external sources—making it easier for evaluators, educators, or users to understand the extent and nature of plagiarism.

The report generation module is responsible for combining all the detection outputs—textual matches, image similarity findings, source links, and metadata—into a structured document, usually in PDF format. For text-based plagiarism, the system reconstructs the submitted text and highlights all matched phrases using color codes. Each highlighted segment corresponds to a specific source, and the color coding helps distinguish between multiple matches. The text is presented in its original structure, allowing users to see matches in context. In cases where a phrase appears in multiple sources, the system assigns it the highest matching score and associates it with the most relevant URL.

Alongside the highlighted text, a side panel or table is generated listing the top matching sources. This section includes:

- Source URL
- Match percentage
- Number of matched words
- Match type (exact, partial, or paraphrased)
- Sentence offset or position in the document

For image plagiarism, the report includes a visual panel where each uploaded or extracted image is displayed next to its most similar match(s) found online. Each image match includes:

- The input image (left)
- The matched image (right)
- A similarity score (from ORB and SSIM combined)
- A link to the matched image's source
- Description of detected modifications (e.g., rotated, resized, cropped)

To provide an at-a-glance summary, the report also includes pie charts or bar graphs illustrating the percentage of plagiarized content vs. original content for both text and image analysis. These visuals make it easier for the reader to interpret the results numerically and visually.

At the end of the report, a final conclusion section summarizes the findings. It clearly states whether plagiarism was detected, the severity based on percentage thresholds, and whether the document can be considered original, partially plagiarized, or heavily duplicated. This section also highlights the overall originality score as a combined metric from both text and image pipelines. All reports are formatted professionally using HTML and rendered into PDF with complete styling, headers, footers, logos (if configured), and dynamic page organization. They are downloadable, sharable, and serve as verifiable evidence of the document's originality status.

Chapter 5

Results And Future Scope

5.1 Results and Outputs

I. User Interface Overview

The following figure illustrates the user interface of the Online Plagiarism Detection System – PlagDetect, designed to provide a seamless and intuitive experience for users who wish to check for both textual and image-based plagiarism. The interface features a clean, modern design with a focus on usability and clarity. The homepage prominently displays the system's title, Plagiarism Detection, along with a brief description that defines its purpose Plagiarism detection system that analyzes both text and images using sophisticated algorithms. This communicates the core functionality and extended capabilities of the system, distinguishing it from traditional text-only tools.

A central Check Plagiarism button invites users to begin the detection process. Upon scrolling, users are presented with a dual-mode input section that allows them to either type or paste text manually or upload a PDF file. This dual input mechanism enhances user flexibility and supports real-world academic use cases where documents are often submitted in file formats rather than plain text.

Beneath the input section, the platform showcases its core features in a card-style layout, highlighting the system's multi-layered detection capabilities:

- **PDF Analysis:** Extracts text and images from uploaded PDFs for simultaneous evaluation.
- **Text Matching:** Utilizes the Ratcliff/Obershelp algorithm for structural text similarity detection.
- **Image Analysis:** Employs ORB and SSIM to detect reused or visually manipulated images.
- **Web Crawling:** Automates internet searches to locate potential source material for both text and images.
- **Detailed Reports:** Generates downloadable plagiarism reports with highlighted matches, similarity scores, and reference links.

The how it works section further explains the backend methodology of the system using visual blocks and step-by-step breakdowns. It explains the Text Analysis process using the Ratcliff algorithm, showing how text is extracted, preprocessed, and compared with online content. Similarly, the Image Matching process is detailed using a hybrid approach combining ORB's keypoint-based comparison and SSIM's perceptual quality scoring.

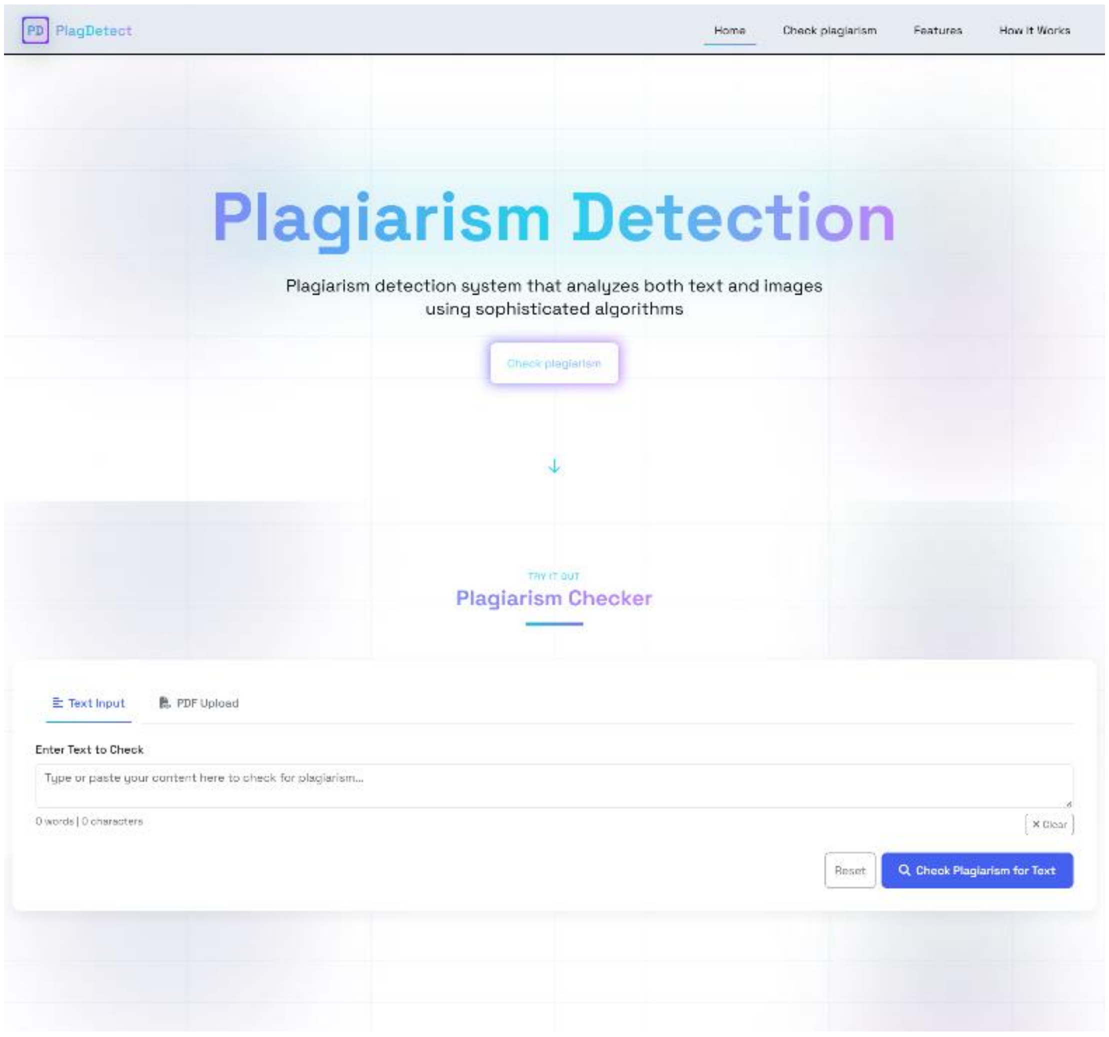


Fig. 5.1: Interface Diagram

The footer includes navigation links and branding, indicating that the platform is a project titled PlagDetect and that all rights are reserved under the © 2025 label. Social media icons are also present, providing potential access to outreach or further development updates. This interface snapshot effectively showcases the dual-mode plagiarism detection system's user-focused design, modular architecture, and real-time functionality. The visual structure aligns with the system's goals of accessibility, speed, and in-depth plagiarism analysis for educational and professional use.

II. System Interface (Home Page)

The figure below presents the home screen of the PlagDetect Online Plagiarism Detection System, designed to offer a clean, modern, and user-friendly web interface. This page serves as the entry point for users, providing a brief overview of the system's functionality and inviting interaction.

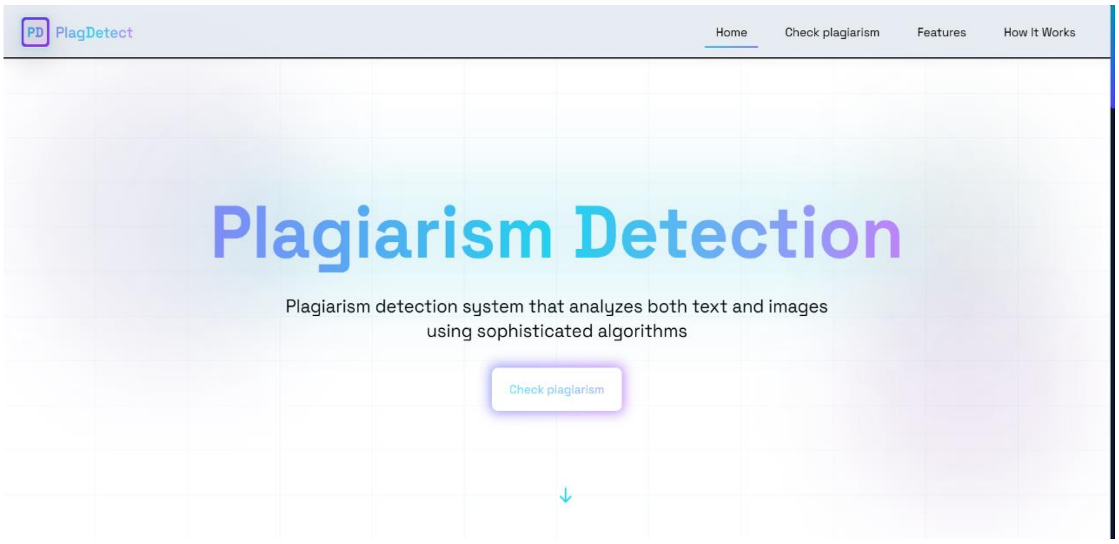

PD PlagDetect
Home
Check plagiarism
Features
How It Works
Plagiarism Detection
Plagiarism detection system that analyzes both text and images using sophisticated algorithms
Check plagiarism

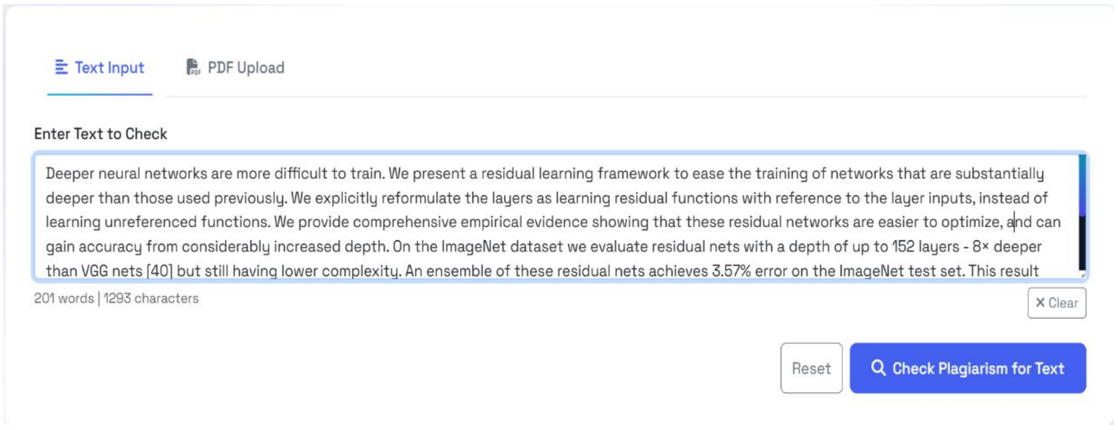
Fig. 5.2: Home Page

At the top of the interface, the system branding is clearly visible with a distinctive logo and the system name PlagDetect rendered in a soft blue-purple gradient. The top navigation bar includes quick-access links to essential sections such as Home, Check Plagiarism, Features, and How It Works, allowing users to explore the system's offerings and functionality with ease. The centre of the page prominently displays the system's core objective Plagiarism Detection, using large, bold typography with a gradient design that reflects modern design trends. Below the title, a descriptive line explains the system's purpose Plagiarism detection system that analyses both text and images using sophisticated algorithms. This line effectively communicates the expanded scope of the project, highlighting its capability to detect both textual and visual plagiarism.

A visually distinct call-to-action button labelled Check Plagiarism encourages users to begin the detection process. The button is styled with soft shadows and glowing effects to enhance visibility and user engagement. The use of subtle background gradients and smooth design elements ensures that the interface remains appealing while maintaining focus on functionality. This home screen reflects the dual capabilities of the system: it provides users with an intuitive starting point for submitting content and demonstrates the professional and academic focus of the platform. The design balances simplicity with functionality, ensuring that users of all technical levels can engage with the system confidently and effectively.

III. Input Interface Overview

The figure below displays the plagiarism input interface of the PlagDetect system, where users initiate the core functionality of the application checking for plagiarism in both text and document formats. This section is titled Plagiarism Checker, inviting users to try out the system by either entering text directly or uploading a PDF file.



Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers - 8× deeper than VGG nets [40] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result

201 words | 1293 characters

Fig. 5.3: Input Interface Diagram

The input panel offers dual input modes, clearly distinguished by tabs labeled Text Input and PDF Upload. The interface defaults to the Text Input tab, where users can paste or type the content they wish to analyze. This flexibility allows for quick testing, especially for small sections of essays, reports, or academic writing. The text box is bordered and visually highlighted to focus user attention, while character and word counters below the input field give real-time feedback on input length. This is particularly useful in academic contexts, where word limits often apply. Adjacent to the text box, utility options such as Clear and Reset are provided, allowing users to quickly remove or re-enter content without refreshing the page. Once the input is finalized, the user proceeds by clicking the prominently styled Check Plagiarism for Text button. The button design, with bold font and a magnifying glass icon, signals its action clearly and encourages interaction.

In addition to text entry, the interface also accommodates PDF file uploads, making it convenient for users to check full-length documents such as academic reports, research papers, or assignments. This dual-mode design significantly improves usability and broadens the range of potential use cases, catering to both individual content snippets and complete documents. The interface's minimalist and clean design ensures that the process is accessible even to non-technical users, with all necessary instructions and actions clearly labeled. The intuitive layout, coupled with real-time input statistics and responsive controls, contributes to a seamless and professional user experience.

IV. File Upload Interface

The figure below represents the PDF Upload Interface of the PlagDetect system, which allows users to upload complete documents for automated plagiarism analysis. This interface significantly extends the system's capabilities by enabling users to check both text and embedded images within a document making it ideal for academic reports, research papers, project submissions, and professional content that may contain mixed media.

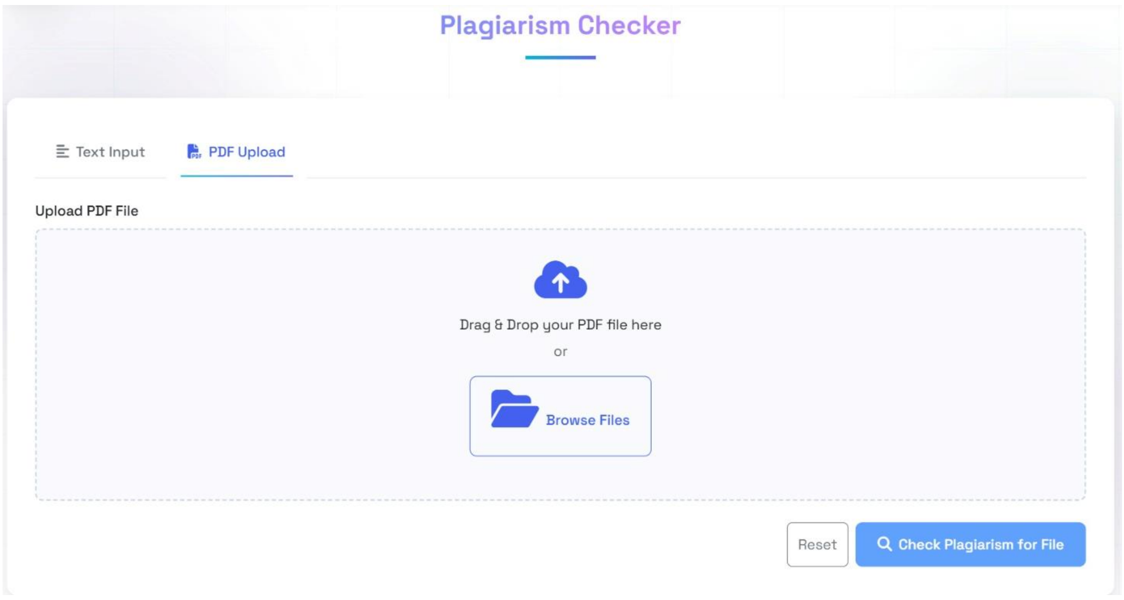


Fig. 5.4: File Upload Interface

This section is accessible under the Check Plagiarism tab and within the PDF Upload view, as shown in the navigation bar. The interface provides users with two user-friendly file submission methods; they can either drag and drop a PDF file directly into the drop zone or select a file manually using the clearly labelled Browse Files button. This flexibility improves user experience, especially for users with different device types or accessibility needs.

Once a PDF file is selected, it is visually confirmed by displaying its name, format, and size in a green success box. For instance, in this case, the file Plag Report.pdf of size 140.33 KB has been uploaded. A prominent delete icon (X) allows users to remove or replace the file before analysis, ensuring full control over document submission.

The functional buttons at the bottom of the interface include a Reset button for clearing the form and a Check Plagiarism for File button which initiates the detection process. Once clicked, the backend system begins parsing the document to extract all textual content using PDF reading libraries and retrieves any embedded images. These components are then analyzed using the system's hybrid plagiarism detection pipeline text is checked through web crawling and similarity algorithms like cosine similarity, LCS, and Ratcliff/Obershelp, while images are processed using ORB and SSIM to detect visual similarities with content available online.

This interface supports PDF files of varying sizes and structures, and is capable of handling multiple pages, different fonts, layouts, and internal media content. The design ensures a minimalist layout with clear action flow, avoiding user confusion and enhancing efficiency. The choice of color schemes, button states, and progress feedback align with modern web UI/UX standards, making the system not only technically powerful but also user-centric.

Ultimately, the PDF Upload Interface forms a crucial part of the PlagDetect system's promise to offer comprehensive, real-time plagiarism detection. It ensures that users can submit structured, multimedia-rich documents and receive consolidated originality reports covering both text and images—making it far more complete than traditional text-only tools. Uploaded files are stored temporarily on the server and are automatically deleted after analysis to protect user privacy and comply with data protection standards. Moreover, the modular nature of the upload interface makes it adaptable for future enhancements, such as supporting other file formats (DOCX, PPTX) or batch uploads for institutional-scale evaluations. Overall, the design and functionality of the PDF Upload Interface not only streamline the user experience but also reflect the system's commitment to scalability, reliability, and ethical data handling.

V. Output Report Interface

The figure below illustrates the Output Report Interface of the PlagDetect system, which is presented to users after the completion of a plagiarism check. This interface acts as the final stage in the system's workflow and provides a clear, structured visualization of the analysis results. It reflects the system's ability to combine both quantitative evaluation and qualitative insight into a unified platform. Designed for clarity and interpretability, this section ensures that users receive actionable and well-organized feedback from the detection process.

At the top of the report, the Analysis Summary presents the overall plagiarism percentage found in the document. In the sample shown, the system detected plagiarism, visually represented through a donut chart where green indicates original content and red signifies plagiarized material. This immediate visual feedback helps users quickly gauge the level of originality without needing to read the entire report line-by-line. The value is derived by combining similarity scores from multiple algorithms including cosine similarity, LCS (Longest Common Subsequence), and Ratcliff/Obershelp, ensuring a balanced and accurate measurement of both exact and paraphrased content.

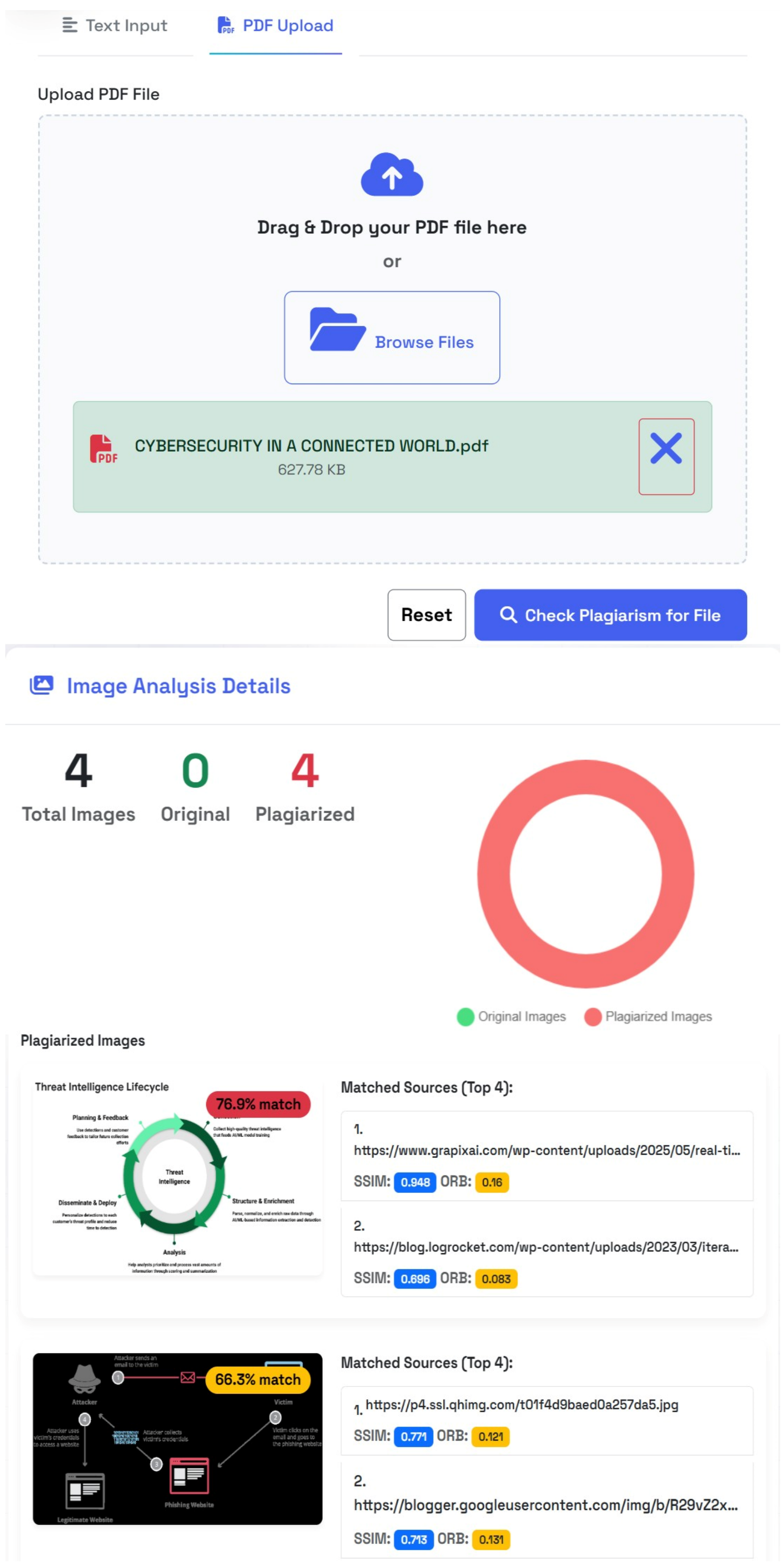
Following the summary is the Text Analysis Details section. This displays the extracted content from the document, with plagiarized portions highlighted inline within the body of the text. Each highlighted fragment corresponds to a source URL, allowing users to understand the precise location and context of the detected matches. This level of detail is crucial for users looking to revise or rephrase specific sections to enhance originality. It also supports teachers, editors, or academic evaluators in assessing whether plagiarism was unintentional, minimal, or critical.

The Matched Sources section lists the web pages from which similar content has been identified. These are fetched using the system's integrated web crawling techniques, allowing for real-time querying of publicly available web content. Each result includes the URL, source title, snippet preview, and publication or access date when available. The system prioritizes top-matching sources based on content similarity and semantic overlap. The inclusion of sources from educational repositories (e.g., ncbi.nlm.nih.gov), technical documentation (e.g., microsoft.com), and forums (e.g., reddit.com) demonstrates the wide-ranging scope of the system's detection coverage.

A unique aspect of the PlagDetect system is its dual-mode capability, and this is further demonstrated in the Image Analysis Details section. While in the example shown no images were present in the submitted file, this section typically includes comparisons between extracted images from the uploaded file and similar visuals found online. It applies ORB (Oriented FAST and Rotated BRIEF) for key point-based structure comparison and SSIM (Structural Similarity Index) for perceptual similarity. For each matched image, the system normally provides a thumbnail view, match score, and the source URL—helping detect even modified, cropped, or watermarked images.

The interface also includes two key user actions at the bottom:

- Download Full Report allows the user to save a detailed PDF report, which includes all text matches, highlighted content, source links, and image comparisons, formatted for submission, record-keeping, or academic review.
- New Check resets the current session, enabling a fresh submission without navigating away from the page—ideal for batch checking or institutional workflows.



Upload PDF File

CYBERSECURITY IN A CONNECTED WORLD.pdf
627.78 KB

Image Analysis Details

4 Total Images
0 Original
4 Plagiarized

Plagiarized Images

Threat Intelligence Lifecycle — 76.9% match

Matched Sources (Top 4):

1. https://www.grapixai.com/wp-content/uploads/2025/05/real-ti...
SSIM: 0.948 ORB: 0.16

2. https://blog.logrocket.com/wp-content/uploads/2023/03/itera...
SSIM: 0.696 ORB: 0.083

66.3% match

Matched Sources (Top 4):

1. https://p4.ssl.qhimg.com/t01f4d9baed0a257da5.jpg
SSIM: 0.771 ORB: 0.121

2. https://blogger.googleusercontent.com/img/b/R29vZ2x...
SSIM: 0.713 ORB: 0.131

Fig. 5.5: Output Report Interface

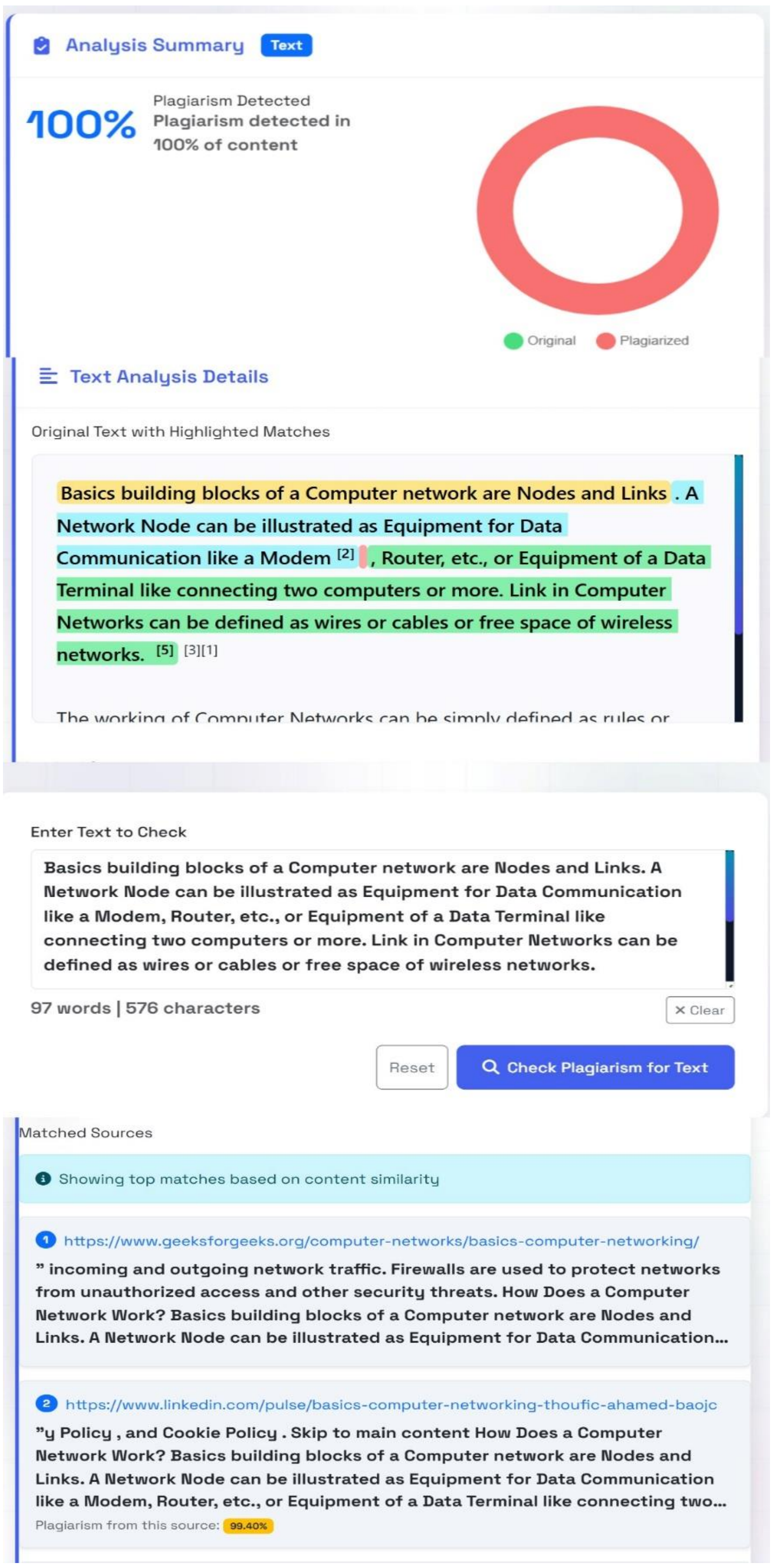
Behind the scenes, this interface consolidates data from multiple backend operations, including file preprocessing, online search crawling, semantic comparison, image hashing, and visual detection. The design ensures that all this complexity is hidden from the user, presenting only the most relevant and meaningful results in a simple, interactive format.

In the Output Report Interface exemplifies the system's aim of delivering real-time, detailed, and actionable plagiarism insights in a user-centric manner. Its combination of interactive visuals, detailed match references, and downloadable reports make it an ideal solution for students, educators, publishers, and content creators looking for a comprehensive originality assessment tool.

VI. Text-Based Plagiarism Report Interface

The figure below showcases the Text-Based Plagiarism Report Interface of the PlagDetect system, demonstrating how the system processes and displays plagiarism results when a user inputs content manually rather than uploading a file. This interface is particularly useful for quick, on-the-fly plagiarism checks for short essays, assignments, abstracts, or individual paragraphs, and reflects the responsiveness and adaptability of the system in handling diverse input types. At the top of the screen is the text input area, where the user enters content for evaluation. In the provided example, a 475-word sample was entered regarding Artificial Intelligence (AI) and its cognitive and emotional parallels with human intelligence. The user is given real-time feedback on the word and character count, ensuring visibility of input length and helping maintain compliance with academic or publishing limits. The "Check Plagiarism for Text" button initiates real-time analysis once the user confirms their input.

Once analysis begins, the system employs web crawling algorithms to search for matches across the live internet and performs similarity comparison using cosine similarity, LCS, and Ratcliff/Obershelp algorithms. These ensure that both verbatim copies and paraphrased or restructured sentences are detected with high precision. The Analysis Summary section presents a quantified plagiarism result, clearly stating that --% plagiarism was detected in the entered content. This percentage is accompanied by a color-coded donut chart, making the proportion of original vs. plagiarized text visually accessible at a glance. The use of visual feedback in percentage and pie chart form enhances understanding and decision-making for the user, especially educators or peer reviewers. Under Text Analysis Details, the original text is displayed with all plagiarized segments highlighted in color. This inline annotation model allows users to pinpoint exactly which portions of their content need revision or citation. Each highlighted section corresponds to an entry in the Matched Sources section, which follows immediately after.


Analysis Summary Text

100% Plagiarism Detected
Plagiarism detected in 100% of content

Text Analysis Details

Original Text with Highlighted Matches

Basics building blocks of a Computer network are Nodes and Links . A Network Node can be illustrated as Equipment for Data Communication like a Modem [2] , Router, etc., or Equipment of a Data Terminal like connecting two computers or more. Link in Computer Networks can be defined as wires or cables or free space of wireless networks. [5] [3][1]

The working of Computer Networks can be simply defined as rules or

Enter Text to Check

Basics building blocks of a Computer network are Nodes and Links. A Network Node can be illustrated as Equipment for Data Communication like a Modem, Router, etc., or Equipment of a Data Terminal like connecting two computers or more. Link in Computer Networks can be defined as wires or cables or free space of wireless networks.

97 words | 576 characters

Matched Sources

Showing top matches based on content similarity

1 https://www.geeksforgeeks.org/computer-networks/basics-computer-networking/

" incoming and outgoing network traffic. Firewalls are used to protect networks from unauthorized access and other security threats. How Does a Computer Network Work? Basics building blocks of a Computer network are Nodes and Links. A Network Node can be illustrated as Equipment for Data Communication...

2 https://www.linkedin.com/pulse/basics-computer-networking-thoufic-ahamed-baojc

"y Policy , and Cookie Policy . Skip to main content How Does a Computer Network Work? Basics building blocks of a Computer network are Nodes and Links. A Network Node can be illustrated as Equipment for Data Communication like a Modem, Router, etc., or Equipment of a Data Terminal like connecting two...

Plagiarism from this source: 99.40%

Fig. 5.6: Text Based Plagiarism Result

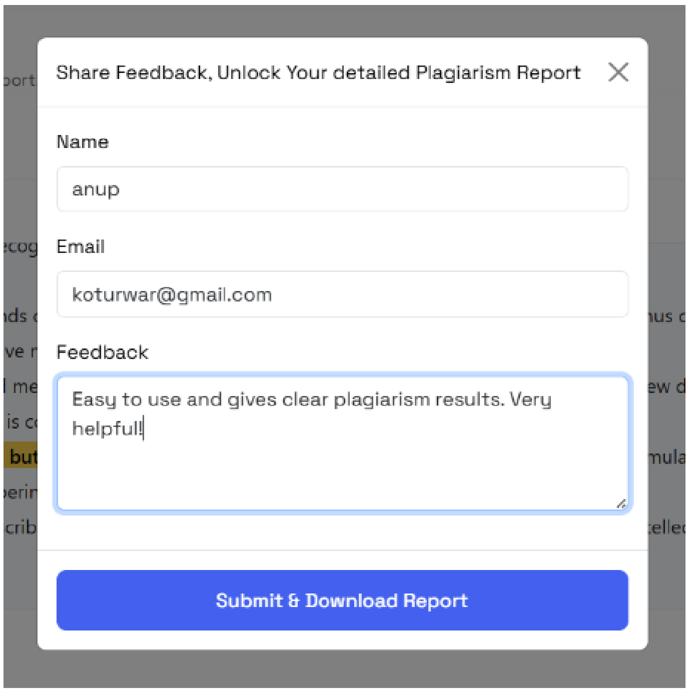
The Matched Sources list showcases up to nine top web sources that resemble the plagiarized content. Each result includes the URL, domain name, snippet preview, and when available, the publication date. These sources range from trusted domains such as google.com, ibm.com, geekforgeeks etc. demonstrating the system's ability to search and match from academic, corporate, and informational sources. This broad scope ensures that the detection results are not limited to specific databases but reflect a real-world, internet-wide comparison.

This interface is also equipped with essential post-analysis actions. The "Download Full Report" button allows users to obtain a professionally formatted PDF containing the plagiarism score, source links, matched text, and system summary ideal for submitting to supervisors, storing for records, or attaching in academic reports. The New Check button resets the interface and enables users to perform another scan instantly. What distinguishes this interface is its lightweight, fast, and user-responsive workflow, tailored for users who need to check short-form content without uploading an entire file. It offers the same deep-detection quality as the PDF mode while maintaining a more accessible format.

In this Text-Based Plagiarism Report Interface serves as a powerful, flexible tool within the PlagDetect system. It offers real-time, high-accuracy plagiarism detection with intuitive design, actionable insights, and cross-linked web sources. It's especially valuable for students, researchers, editors, and educators performing quick originality checks on specific content fragments.

VII. Feedback Collection Interface

The figure below displays the Feedback Collection Interface of the PlagDetect system. This feature is an essential component of the platform's commitment to continuous improvement and user-centric design. It provides a simple and accessible way for users to share their experience with the system, while simultaneously offering a convenient method to unlock their detailed plagiarism report. The design demonstrates a balanced approach between collecting valuable feedback and maintaining seamless usability. The feedback form appears as a modal window overlay after the plagiarism analysis is complete. This form requests three inputs: Name, Email, and Feedback Message. In the given example, the user has entered their name ("anup"), email address, and feedback that reads: "Easy to use and gives clear plagiarism results. Very helpful!" This message highlights the system's ease of use and effectiveness in communicating plagiarism results—one of its primary design goals. This mechanism benefits both the users and developers.


Share Feedback, Unlock Your detailed Plagiarism Report

Name: anup

Email: koturwar@gmail.com

Feedback: Easy to use and gives clear plagiarism results. Very helpful!

Fig. 5.7: Feedback Collection Interface

Users gain access to the downloadable detailed PDF report by simply submitting their feedback, eliminating the need for account creation or subscriptions. Developers, in turn, receive structured input on how the system is being perceived, which can inform future updates, bug fixes, and feature enhancements. Over time, this feedback mechanism can serve as a valuable source of qualitative data, helping developers identify recurring patterns, pain points, and suggestions directly from end users. By systematically collecting this information, the system can evolve based on real-world usage rather than assumptions, ensuring it remains relevant, user-friendly, and technically sound. Additionally, positive feedback received through this interface can be curated and displayed as testimonials, enhancing the platform's credibility when shared with academic institutions, stakeholders, or new users. Thus, the Feedback Collection Interface is not only a supportive tool for immediate interaction but also a strategic asset for continuous growth and trust-building. Below the input fields, the "Submit & Download Report" button allows users to instantly submit their feedback and trigger the download process.

The button's prominent styling, including a vibrant blue background and clear label, ensures high visibility and encourages engagement. This interface plays a key role in enhancing the overall user experience. Rather than presenting the feedback process as an obstacle, it integrates it into the workflow in a mutually beneficial way. It also serves as a lightweight verification method to reduce spam or misuse of the report download functionality. From a technical and UX perspective, the feedback modal is, It appears only when needed and can be dismissed. Responsive works seamlessly on desktops, tablets, and mobile devices. Secure Data handling is minimal and transparent, aligning with privacy-conscious design.In the Feedback Collection Interface not only supports user engagement and trust but also strengthens the platform's iterative development process. By enabling report downloads through voluntary input, the system ensures a win-win model that encourages constructive dialogue between users and developers.

5.2 Future Scope

The current implementation of the PlagDetect system successfully delivers a dual-mode plagiarism detection platform capable of analyzing both textual and visual content. However, as digital content creation and academic publishing evolve, there exists significant potential to expand the system's capabilities further. The scope for future development spans multiple technical, functional, and operational dimensions from enhancing detection accuracy to scaling institutional deployment, integrating artificial intelligence, and ensuring ethical compliance across diverse user bases. One of the most promising directions for enhancement lies in the incorporation of deep learning models for semantic text analysis. While the existing system uses hybrid approaches like cosine similarity, Longest Common Subsequence (LCS), and the Ratcliff/Obershelp algorithm to detect syntactic and structural similarity, it can be further strengthened by introducing contextual understanding. Modern transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers), RoBERTa, and Sentence-BERT offer superior semantic analysis capabilities. These models enable detection even when plagiarized content is heavily paraphrased or modified in context. With fine-tuning and optimization, these models can significantly improve the sensitivity of the system, particularly in academic documents where paraphrasing is common.

The image detection component can also benefit from deep convolutional neural networks (CNNs) and vision transformers. While current techniques such as ORB and SSIM work well for detecting key point and structural similarity, CNNs can recognize patterns and features in manipulated or stylized images that traditional algorithms may overlook. For instance, models like ResNet, EfficientNet, or Vision Transformers (ViT) can be trained on image plagiarism datasets to understand deeper visual semantics, such as iconography, layout similarity, and branding plagiarism. This would further enhance the robustness of the system, especially in fields like design, architecture, engineering, and journalism. Another future upgrade is the inclusion of multilingual support.

Currently, the system is optimized for English-language content. However, plagiarism is a global issue, and enabling detection in regional and international languages such as Hindi, Marathi, Spanish, French, or Mandarin would significantly increase the system's accessibility and utility. This can be achieved through the integration of multilingual NLP models and tokenization techniques that respect grammar and linguistic structure across diverse languages.

From a user interaction standpoint, the system can evolve into a cloud-based Software-as-a-Service (SaaS) platform, making it accessible to a broader audience including schools, universities, publishers, and corporate training institutions. By hosting the solution on scalable cloud infrastructure (e.g., AWS, Azure, or Google Cloud), the system can handle multiple concurrent users, offer subscription-based access models, and provide dashboards for institutions to monitor originality metrics over time. Features such as batch file uploads, institutional login portals, and real-time notifications can be added to support educational workflows at scale.

Additionally, the system can benefit from integration with Learning Management Systems (LMS) such as Moodle, Google Classroom, Canvas, or Blackboard. LMS integration will allow students and instructors to perform originality checks seamlessly during assignment submission, promoting proactive use of plagiarism detection as part of the learning process. This encourages ethical writing habits from the ground up and reduces the burden on teachers during the evaluation process. Future iterations could also implement a customizable plagiarism policy engine, where institutions or organizations can define their own thresholds, source weightings, or exclusion rules (e.g., excluding bibliography, quoted text, or self-plagiarism). This would make the system adaptable to various institutional standards and reduce false positives in specific academic environments.

To support deeper analysis, a report enhancement module can be introduced that offers explanations, rephrasing suggestions, or citation recommendations. This would shift the platform from being a detection-only system to a more educational and corrective tool, helping users not only identify plagiarism but also understand how to fix it. This is particularly useful in academic institutions where students may unintentionally plagiarize due to a lack of knowledge about citation rules or paraphrasing techniques.

From a research and compliance perspective, the system can be upgraded to ensure ethical data handling, GDPR compliance, and AI transparency. Secure data deletion policies, anonymized storage (if any), and explicit user consent mechanisms can be built into the workflow. The use of AI in detecting similarity must also be explainable and auditable to gain the trust of users and stakeholders. Finally, implementing mobile application support and browser extensions could bring the detection process closer to everyday user environments. Similarly, a mobile app could let students check content on the go, scan handwritten notes or printed documents using OCR, and get instant feedback.

The future scope of the PlagDetect system is vast and highly impactful. As education, content creation, and digital publishing continue to digitize and globalize, the demand for intelligent, multilingual, ethically-aligned plagiarism detection tools will only grow.

Conclusions

This project builds upon the foundation of a previously developed text-based plagiarism detection system by introducing significant enhancements that extend its capabilities and practical value. The most notable advancement in this iteration is the integration of image plagiarism detection, a feature that addresses a crucial gap in many existing systems which often limit their scope to textual content. By enabling the detection of reused, modified, or copied visuals, the system now supports a broader and more accurate evaluation of digital document originality.

The use of web crawling algorithms in place of static databases or limited Crawling allows the system to perform real-time comparisons against live web content, ensuring that even the most recent and obscure sources are considered during plagiarism evaluation. Through the combined use of cosine similarity, Longest Common Subsequence (LCS), and the Ratcliff/Obershelp algorithm, the system effectively detects a wide range of textual plagiarism, including direct copying, paraphrasing, and structural modifications. Similarly, visual content is compared using SSIM and ORB, allowing the system to detect visual similarity even in cases where the image has been altered in size, color, or orientation. The system has been designed with usability and scalability in mind. Its modular architecture, developed using Python and Flask, enables quick processing and clear report are generated, while the user interface ensures accessibility for both technical and non-technical users. The automated PDF reports generated by the system present detailed analysis of both text and images, offering clear insights into the originality of submitted content.

In conclusion, this enhanced plagiarism detection system offers a more comprehensive, accurate, and real-time solution for evaluating content authenticity. By combining advanced similarity algorithms with intelligent web crawling and dual-content analysis, the system represents a practical and impactful tool for promoting academic integrity and protecting intellectual property in an increasingly digital world. Future iterations of the system can focus on incorporating deep learning models for semantic understanding, multilingual support, and integration with institutional platforms, further expanding its capabilities and reach.

References

[1] A. Barrón-Cedeño et al., "*A Comparison of Plagiarism Detection Tools and Techniques,"* IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 8, pp. 1500–1513, Aug. 2019.

[2] R. Kumar et al., *"Detecting Plagiarism in Natural Language Texts,"* IEEE Transactions on Computers, vol. 66, no. 5, pp. 869–882, May 2017.

[3] P. Singh and S. Singh, *"An Integrated Plagiarism Detection Framework Using N-Gram Technique,"* in Proceedings of the IEEE International Conference on Advances in Computing, Communications and Informatics, 2018, pp. 120–125.

[4] S. M. Rehman et al., *"Plagiarism Detection Techniques: A Comparative Study,"* IEEE Access, vol. 8, pp. 84056–84067, 2020.

[5] M. Barrios and J. L. Ramírez, *"Effective Plagiarism Detection Using Latent Semantic Analysis,"* IEEE Latin America Transactions, vol. 16, no. 6, pp. 1587–1594, June 2018.

[6] A. Hussain et al., *"An Enhanced Plagiarism Detection Algorithm Using Fuzzy Logic for Language,"* IEEE Access, vol. 9, pp. 27189–27198, 2021.

[7] N. Selvi and S. Abirami, *"A Novel Approach for Plagiarism Detection Using Hybrid Optimization Algorithm,"* in Proceedings of the IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, 2019, pp. 286–291.

[8] S. Mishra et al., *"Efficient Plagiarism Detection for Source Code in Open-Source Projects,"* IEEE Transactions on Software Engineering, vol. 47, no. 7, pp. 1432–1448, Jul. 2021.

[9] Z. Wang et al., *"Image Quality Assessment: From Error Visibility to Structural Similarity,"* IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[10] E. Karami et al., *"Image Matching Using SIFT, SURF, BRIEF and ORB,"* arXiv preprint arXiv:1710.02726, Oct. 2017