PLAGIARISM DETECTION USING NLP

A DESIGN PROJECT REPORT

submitted by

ABINAYA SHREE J

CHARULATHA K

HARSHITHA K

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

JUNE 2025

PLAGIARISM DETECTION USING NLP

A DESIGN PROJECT REPORT

submitted by

ABINAYA SHREE J (811722104004)

CHARULATHA K (811722104023)

HARSHITHA K (811722104052)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

JUNE 2025

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"PLAGIARISM DETECTION USING NLP"** is Bonafide work of **ABINAYA SHREE J (811722104004), CHARULATHA K (811722104023), HARSHITHA K (811722104052)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr. A Delphin Carolina Rani, M.E.,Ph.D.,	Mrs. R Sathya, M.E.,(Ph.D.,)
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	Assistant Professor
Department of CSE	Department of CSE
K Ramakrishnan College of Technology	K Ramakrishnan College of Technology
(Autonomous)	(Autonomous)
Samayapuram – 621 112	Samayapuram – 621 112

Submitted for the viva-voice examination held on ………………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **“PLAGIARISM DETECTION USING NLP”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of Bachelor of Engineering. This project report is submitted on the partial fulfilment of the requirement of the awardof Degree of Bachelor of Engineering.

Signature

ABINAYA SHREE J

CHARULATHA K

HARSHITHA K

Place: Samayapuram

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtedness to our institution **"K RAMAKRISHNAN COLLEGE OF TECHNOLOGY"**, for providing us with the opportunity to do this project.

We are glad to credit and praise our honorable and respected chairman sir **Dr. K RAMAKRISHNAN, B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering adequate duration to complete it.

We would like to thank **Dr. N VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project with full satisfaction.

We heartily thank **Dr. A DELPHIN CAROLINA RANI, M.E., Ph.D.,** Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing her support to pursue this project.

We express our deep and sincere gratitude and thanks to our project guide **Mrs. R SATHYA, M.E., (Ph.D.,)** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry our this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

Plagiarism detection is vital in academic and professional settings where originality and integrity are paramount. An AI-powered system using Natural Language Processing (NLP) techniques offers an effective solution by analyzing user-submitted documents against a reference database to identify similarities. Built with the Flask web framework, the system provides a user-friendly interface for document uploads and displays detailed plagiarism reports, highlighting matched sections and showing overall similarity percentages.

By incorporating text preprocessing, TF-IDF vectorization, and cosine similarity, the system detects both exact matches and semantically similar content, including paraphrased material. This advanced approach ensures accurate, efficient, and transparent plagiarism evaluation, offering significant value in validating academic work and professional content.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	Proposed System Architecture	12
3.2	Flow Chart	13

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
TF-IDF	Term Frequency–Inverse Document Frequency
UI	User Interface
AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
IDE	Integrated Development Environment
FTLM	Full Transformer Language Model
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution

CHAPTER 1

INTRODUCTION

1.1OVERVIEW

Plagiarism detection has become increasingly vital in both academic and professional settings, where maintaining originality and ethical standards is a fundamental requirement. With the exponential growth of digital content and easy access to vast online resources, the unintentional or deliberate replication of content has become a common concern. Addressing this issue requires an intelligent, automated system capable of analyzing documents and identifying similarities beyond mere word-for-word copying. This project presents an AI-powered plagiarism detection system that leverages advanced Natural Language Processing (NLP) techniques to evaluate the originality of submitted content. By utilizing text preprocessing, Term Frequency-Inverse Document Frequency (TF- IDF) vectorization, and cosine similarity measures, the system can accurately detect not only direct copying but also paraphrased or semantically similar text. The core functionality involves comparing user-uploaded documents against a reference database to compute a plagiarism percentage and highlight overlapping content. Built using the Flask web framework, the system features a user-friendly interface that simplifies the document submission process and clearly presents the results. The integration of NLP with a web-based platform ensures that the evaluation process is both efficient and accessible, supporting educators, researchers, and content creators in upholding academic and professional integrity. This innovative solution enhances the reliability of plagiarism detection while promoting a culture of authenticity and responsible authorship.

1.2PROBLEM STATEMENT

In today’s digital age, the widespread availability of online content has made it increasingly easy to copy, reuse, and rephrase existing material, leading to a significant rise in plagiarism across educational, academic, and professional domains. Traditional plagiarism detection methods often fall short in identifying paraphrased content or semantically similar text, relying mainly on surface-level comparisons that lack contextual understanding. This creates a critical need for a more advanced, accurate, and efficient system that can detect not only direct copying but also subtle forms of content duplication. The challenge lies in designing a scalable solution that can analyze large volumes of text, process and compare them intelligently, and present the results in an interpretable and user-friendly manner. Addressing this issue, the proposed project aims to develop an AI-powered plagiarism detection system using NLP techniques such as text preprocessing, TF-IDF vectorization, and cosine similarity to detect content overlap and similarity with greater precision, thereby ensuring content integrity and academic honesty.

1.3OBJECTIVE

The primary objective of this project is to develop an intelligent plagiarism detection system that ensures content originality and upholds academic and professional integrity.

- The system aims to accurately identify both exact matches and paraphrased similarities in user-submitted documents by leveraging Natural Language Processing (NLP) techniques.
- Key goals include implementing efficient text preprocessing, TF-IDF vectorization, and cosine similarity methods to perform robust content comparison. Additionally, the project seeks to create a user-friendly web interface using the Flask framework that allows users to upload documents seamlessly and view plagiarism results with clear percentage scores and highlighted matches.

- Another important objective is to offer a reliable and transparent evaluation tool that can support academic institutions, content creators, and organizations in detecting and minimizing plagiarism effectively. By achieving these goals, the system will serve as a valuable asset in promoting ethical content creation and maintaining high standards of originality.

CHAPTER 2

LITERATURE SURVEY

1.P. Sharmila; KalaiarasiSonaiMuthuAnbananthen; NithyakalaGunasekaran; BaarathiBalasubramaniam; DeisyChelliah et all proposed to "FTLM: A Fuzzy TOPSIS Language Modeling Approach for Plagiarism Severity Assessment" IEEE-2024

Detecting plagiarism poses a significant challenge for academic institutions, research centers, and content-centric organizations, especially in cases involving subtle paraphrasing and content manipulation where conventional methods often prove inadequate. Our paper proposes FTLM (Fuzzy TOPSIS Language Modeling), a novel method for detecting plagiarism within decision science. FTLM integrates language models with fuzzy sorting techniques to assess plagiarism severity by evaluating the similarity of potential solutions to a reference. The method involves two stages: leveraging language modeling to define criteria and alternatives and implementing enhanced fuzzy TOPSIS. Word usage patterns, grammatical structures, and semantic coherence represent fuzzy membership functions. Moreover, pre-trained language models enhance semantic similarity analysis. This approach highlights the benefits of combining fuzzy logic's tolerance for imprecision with the semantic evaluation capabilities of advanced language models, thereby offering a comprehensive and contextually aware method for analyzing plagiarism severity. The experimental results on the benchmark dataset demonstrate effective features that enhance performance on the user-defined severity ranking order.

2.Hayden Cheers; Yuqing Lin; Shamus P. Smith et all proposed to "Academic Source Code Plagiarism Detection by Measuring Program

Behavioral Similarity" IEEE-2021

Source code plagiarism is a long-standing issue in tertiary computer science education. Many source code plagiarism detection tools have been proposed to aid in the detection of source code plagiarism. However, existing detection tools are not robust to pervasive plagiarism-hiding transformations and can be inaccurate in the detection of plagiarised source code. This article presents BPlag, a behavioural approach to source code plagiarism detection. BPlag is designed to be both robust to pervasive plagiarism-hiding transformations and accurate in the detection of plagiarised source code. Greater robustness and accuracy is afforded by analyzing the behavior of a program, as behavior is perceived to be the least susceptible aspect of a program impacted upon by plagiarism-hiding transformations. BPlag applies symbolic execution to analyses execution behavior and represents a program in a novel graph-based format. Plagiarism is then detected by comparing these graphs and evaluating similarity scores. BPlag is evaluated for robustness, accuracy and efficiency against five commonly used source code plagiarism detection tools. It is then shown that BPlag is more robust to plagiarism-hiding transformations and more accurate in the detection of plagiarised source code, but is less efficient than the compared tools.

3.Muhammad FarazManzoor; Muhammad Shoaib Farooq; Muhammad Haseeb; Uzma Farooq; Sohail Khalid; Adnan Abid et all proposed to "Exploring the Landscape of Intrinsic Plagiarism Detection: Benchmarks, Techniques, Evolution, and Challenges" IEEE-2022.

In the realm of text analysis, intrinsic plagiarism detection plays a crucial role by aiming to identify instances of plagiarized content within a document and determining whether parts of the text originate from the same author. As the development of Large Language Models (LLMs) based content generation tools such as, ChatGPT is publicly available, the challenge of intrinsic plagiarism has become increasingly significant in various domains. Consequently, there is a growing demand for robust and accurate detection methods to address this evolving landscape. This study conducts a comprehensive Systematic Literature Review (SLR), analyzing 44 research papers that explore various facets of intrinsic plagiarism detection, including common datasets, feature extraction techniques, and detection methods. This SLR also highlights the evolution of detection approaches over time and the challenges faced in this context especially challenges associated with low-resource languages. To the best of our knowledge, there is no SLR exclusively based on the intrinsic plagiarism detection that bridge the gap in existing literature and offering valuable insights to researchers and practitioners. By consolidating the state-of-the-art findings, this SLR serves as a foundation for future research, enabling the development of more effective and efficient plagiarism detection solutions to combat the ever-evolving challenges posed by plagiarism in today’s digital age.

4.Zhenzhou Tian; Qing Wang; Cong Gao; Lingwei Chen; Dinghao Wu et all proposed to "Plagiarism Detection of Multi-Threaded Programs via Siamese Neural Networks" IEEE-2020

Widespread intentional or unintentional software plagiarisms have posed serious threats to the healthy development of software industry. In order to detect such evolving software plagiarism, software dynamic birthmark techniques of better anti-obfuscation ability serve as one of the most promising methods. However, due to the perturbation caused by non-deterministic thread scheduling in multi-threaded programs, existing dynamic approaches optimized for sequential programs may suffer from the randomness in multi-threaded program plagiarism detection. Some thread-aware birthmarking methods have been then proposed to address this issue, which nevertheless largely rely on manual feature engineering and empirical observations without any ground-truth training, and thus require domain knowledge, making them inflexible to be deployed in the wild. Inspired by the success of self-guided optimization using deep neural networks and their superior feature learning ability, in this article, we transform multiple execution traces for each multi-threaded program under a specified input to the plain feature matrix, and feed it to the deep learning framework to learn latent representation as thread-aware birthmark that enjoys better semantic richness and perturbation resistance; instead of empirically determining the plagiarism over direct birthmark similarity metric, we further build up sophisticated siamese neural networks to supervise birthmark construction, similarity measurement, and decision making. Integrating our proposed method, a system called NeurMPD is developed to perform Neural network-based Multi- threaded program Plagiarism Detection. The experimental results based on a public software plagiarism sample set demonstrate that NeurMPD copes better with multi-threaded plagiarism detection than alternative approaches.

5.VedranLjubovic; EnilPajic et all proposed to "Plagiarism Detection in Computer Programming Using Feature Extraction From Ultra-Fine-Grained Repositories" IEEE-2020

Detecting instances of plagiarism in student homework, especially programming homework, is an important issue for practitioners. In the past decades, several tools have emerged that are able to effectively compare large corpora of homeworks and sort pairs by degree of similarity. However, those tools are available to students as well, allowing them to experiment and develop elaborate methods for evading detection. Also, such tools are unable to detect instances of “external plagiarism” where students obtained unethical help from sources not among other students of the same course. One way to battle this problem is to monitor student activity while solving their homeworks using a cloud-based integrated development environment (IDE) and detect suspicious behaviours. Each editing event in program source can be stored as a new commit to create a form of ultra-fine-grained source code repository. In this paper, the authors propose several new features that can be extracted from such repositories with the purpose of building a comprehensive profile of each individual developer. Machine learning techniques were used to detect suspicious behaviours, which allowed the authors to significantly improve upon the performance of more traditional plagiarism detection tools.

CHAPTER 3

SYSTEM ANALYSIS

3.1EXISTING SYSTEM

In the current educational landscape, code evaluation is typically carried out through manual grading or the use of basic online code submission platforms. Most existing systems focus primarily on output-based evaluation, where the submitted code is tested against a predefined set of input-output pairs. While this approach can validate the correctness of a program's output, it fails to assess other critical aspects such as code structure, syntax quality, programming style, and originality. Manual grading, on the other hand, is time-consuming, prone to human bias, and lacks consistency, especially when dealing with large numbers of students. Furthermore, these systems rarely provide detailed feedback, making it difficult for students to understand their mistakes and improve. Some platforms include limited plagiarism detection features, but they are not deeply integrated into the evaluation pipeline. As a result, the existing systems do not offer comprehensive and timely evaluation mechanisms that address the diverse needs of both educators and students in modern programming education.

3.1.1DISADVANTAGES

- Limited understanding of creative solutions: The model might not effectively assess unconventional coding approaches or creative problem-solving techniques that deviate from standard patterns.
- Challenges with complex algorithmic logic: The model may struggle to fully understand or assess the intricacies of complex algorithms and design choices, potentially missing the intent behind a solution.

- False positives in plagiarism detection: Automated plagiarism detection may flag common code structures or standard library functions as plagiarized, leading to unfair penalties for students who are not copying others' work.
- Initial setup and maintenance effort: Setting up the system requires significant technical expertise, time, and resources to ensure it can handle diverse programming languages and continually adapt to evolving evaluation criteria.
- Reduced personalized mentoring: The system's automated feedback might decrease the opportunities for one-on-one mentoring, which is important for developing critical thinking and problem-solving skills in students.
- Over-reliance on automated feedback: Students might focus more on meeting system requirements rather than truly understanding programming principles, leading to surface-level learning and a lack of deeper comprehension.

3.2PROPOSED SYSTEM

The proposed system is an AI-powered plagiarism detection application designed to efficiently analyze and compare textual content using advanced Natural Language Processing (NLP) techniques. The system allows users to upload documents through a web-based interface developed using the Flask framework. Once a document is submitted, it undergoes several stages of processing, including text cleaning, tokenization, and normalization to prepare the data for analysis. The core of the system uses TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert the text into numerical representations, which are then compared using cosine similarity to identify exact and semantically similar content. The system cross-references the submitted document against a predefined repository of reference documents to assess the degree of similarity. It calculates and displays a plagiarism percentage along with the highlighted portions that match existing content, allowing users to clearly understand the areas of concern. This intelligent approach not only detects verbatim copying but also uncovers paraphrased content, ensuring a more robust and comprehensive evaluation. By integrating machine learning techniques within a user-friendly web interface, the proposed system provides an effective, automated solution for maintaining content integrity in academic and professional domains.

3.2.1ADVANTAGES

- **Detects paraphrased or restructured content, not just copy-paste**

  Traditional plagiarism tools rely heavily on string matching and fail when sentences are reworded. Your use of TF-IDF and cosine similarity allows the system to understand contextual relationships, capturing nuanced semantic overlaps. This empowers users to identify intellectual plagiarism where ideas are subtly copied.

- **Eliminates manual review bottlenecks with fast, automated comparisons.**

  Once the document is uploaded, the system automatically processes and evaluates it against a corpus, regardless of size. This scalability is critical for institutions managing thousands of submissions—offering results in seconds, not hours**.**

- **Enhances user trust and usability through clear visual reporting.**

  By presenting matched text directly within the document, users gain immediate insight into which sections are flagged and why. This encourages responsible revision, learning, and understanding, particularly valuable in academic settings.

- **Easy integration, maintenance, and deployment.**

  The use of Flask ensures a lightweight, extensible web framework.

Your system's modular design supports future enhancements—like adding multilingual support, synonym detection, or deep learning-based embeddings—without overhauling the base structure.

- **Protects academic and professional credibility.**
  For universities, publishers, and enterprises, undetected plagiarism can lead to reputational damage and legal issues. Your system helps safeguard against these risks by ensuring originality in submissions, fostering a culture of ethical authorship and innovation.

3.3BLOCK DIAGRAM OF PROPOSED SYSTEM

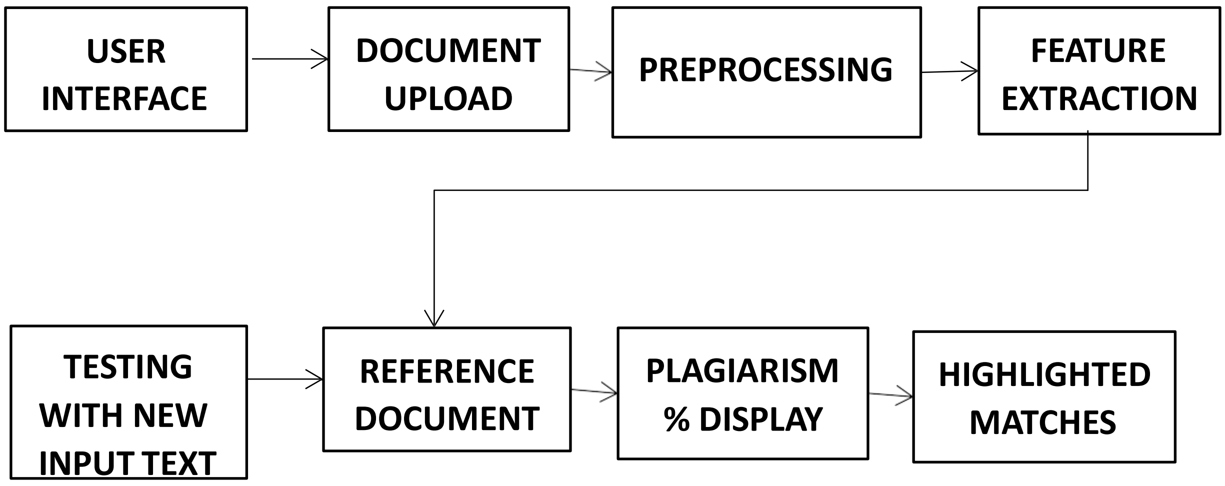


Fig 3.1: Proposed System Architecture

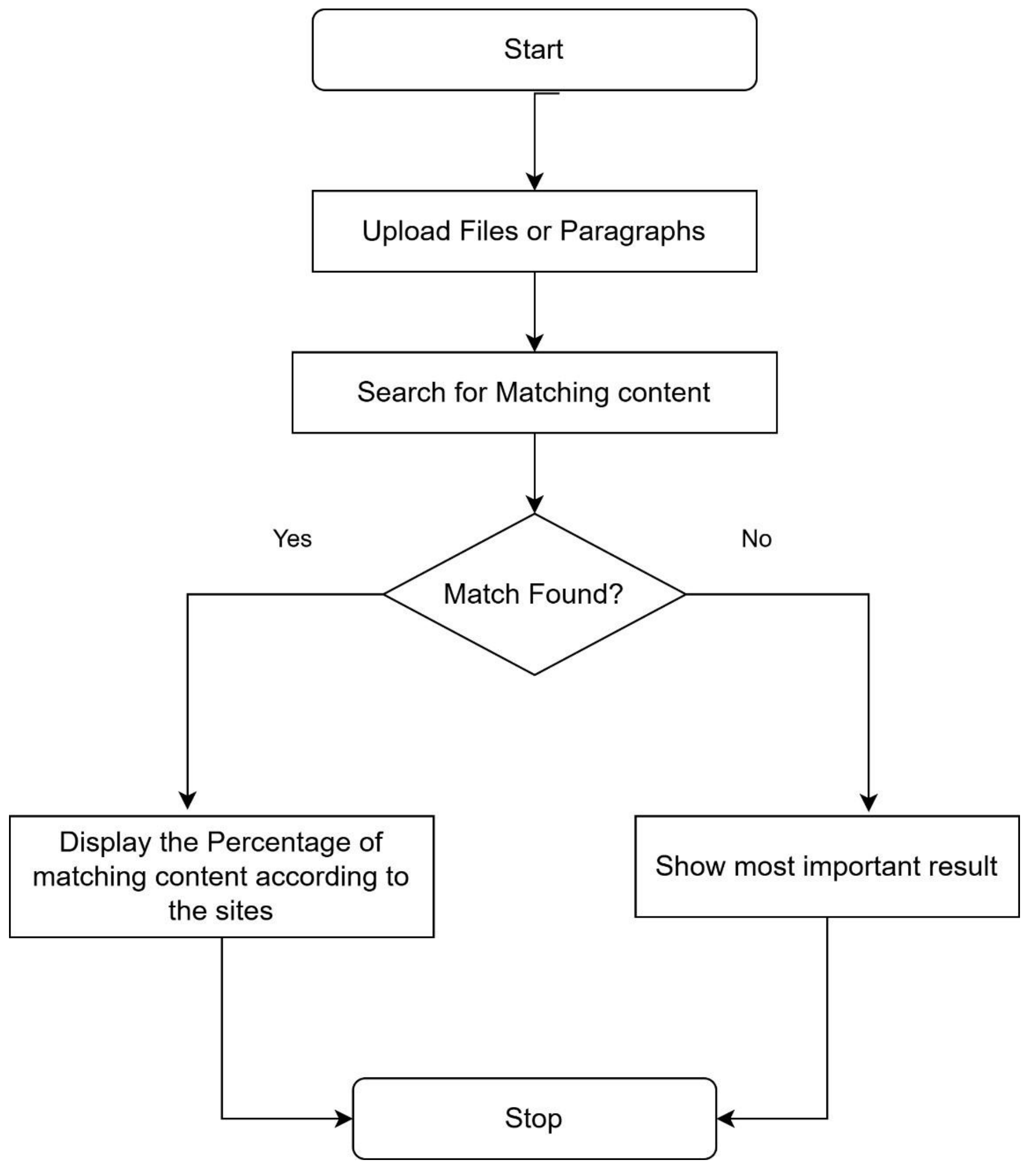

Start
Upload Files or Paragraphs
Search for Matching content
Yes
No
Match Found?
Display the Percentage of matching content according to the sites
Show most important result
Stop

Fig 3.2 Flowchart

CHAPTER 4

MODULES

4.1MODULE DESCRIPTION

- USER INTERFACE
- DOCUMENT UPLOAD MODULE
- PREPROCESSING
- TESTING
- REFERENCE DOCUMENT MODULE

- OUTPUT FOR PLAGIARISM DETECT MODULE

4.1.1.User Interface

The User Interface serves as the entry point for users to interact with the plagiarism detection system. Designed using the Flask web framework, it provides a clean and user-friendly environment where users can navigate through the platform with ease. Through this interface, users can upload documents for analysis, view the results of the plagiarism check, and interpret highlighted text and percentage scores. The UI ensures seamless communication between the user and the backend processes.

4.1.2.Document Upload Module

This module allows users to upload their documents in supported formats such as .txt, .pdf, or .docx. Once a document is uploaded, it is temporarily stored for processing. This module is responsible for validating the file type and size, and for securely passing the document to the preprocessing stage. It acts as a bridge between user input and backend analysis.

4.1.3.Preprocessing

The preprocessing module is the foundation of accurate plagiarism detection, responsible for cleaning and standardizing both user-uploaded documents and reference texts. It begins with text extraction from various formats such as PDF or DOCX, converting the content into raw text. All characters are then transformed to lowercase to ensure uniformity during comparison. Special characters, punctuation, and irrelevant symbols are stripped away to eliminate noise. Common stop words—terms that occur frequently but carry minimal semantic value—are removed to focus the analysis on meaningful content. The text is then tokenized, breaking it down into individual words or phrases, and lemmatization is applied to reduce each word to its root form, preserving semantic integrity while minimizing linguistic variation. These preprocessing steps ensure the input is normalized and optimized for accurate vectorization and similarity analysis in subsequent stages.

4.1.4.Testing

In the testing phase, the cleaned and preprocessed text from the uploaded document is converted into numerical form using TF-IDF (Term Frequency–Inverse Document Frequency) vectorization. This technique quantifies the importance of each term relative to the entire corpus, enabling nuanced text comparisons. The resulting vectors are then analyzed using cosine similarity, which measures the angle between vector representations to determine how closely related two documents are. This mathematical approach is effective not only for detecting exact word-for-word copying but also for identifying paraphrased or semantically similar content. By evaluating the degree of similarity between the submitted text and the reference corpus, the testing module plays a critical role in flagging potential plagiarism across a spectrum of linguistic variations.

4.1.5.Reference Document Module

This module serves as the backbone of the detection engine by maintaining a well-organized, curated repository of reference materials. These materials may include academic publications, publicly available articles, internal reports, or previously submitted student assignments. To ensure accurate and efficient comparison, each reference document undergoes the same preprocessing pipeline—text extraction, normalization, tokenization, and lemmatization—and is stored in its vectorized form using TF-IDF. This preemptive vectorization significantly reduces processing time during the testing phase and allows for rapid similarity computation. By maintaining a clean, scalable, and searchable database of vetted reference texts, this module ensures that every user submission is compared against a reliable baseline of authoritative content.

4.1.6.Output for Plagiarism Detect Module

The final module is responsible for interpreting and presenting the results of the plagiarism check in a clear and actionable manner. It calculates a plagiarism percentage based on the similarity scores derived from the testing phase, offering a quantifiable metric of content originality. In addition to the percentage score, the system highlights specific segments of the uploaded document that closely match the reference texts, using visual cues to distinguish levels of similarity. Each highlighted section is linked to the corresponding reference source, providing transparency and traceability. The system may also generate a comprehensive report summarizing all findings, which can be downloaded for documentation or further review. This output empowers users—be they students, educators, or professionals—with detailed insight into the originality of their work, enabling informed decisions and promoting content integrity.

CHAPTER 5

SOFTWARE DESCRIPTION

5.1HARDWARE REQUIREMENTS

System	:	PC OR LAPTOP
Processor	:	INTEL / AMD
RAM	:	4 GB Recommended
ROM	:	2 GB

5.2SOFTWARE REQUIREMENTS

OPERATING SYSTEM	:	WINDOWS 10/11
LANGUAGE USED	:	PYTHON
BACKEND	:	PYTHON SCRIPT
FRONTEND	:	HTML, CSS, FLASK

CHAPTER 6

TEST RESULT AND ANALYSIS

6.1TESTING

The AI-powered plagiarism detection system was successfully implemented and tested using a diverse set of documents, including academic essays, research articles, and general content with varying degrees of similarity. The system demonstrated strong performance in identifying both exact text matches and paraphrased content, thanks to the integration of TF-IDF vectorization and cosine similarity for semantic comparison. Documents containing copied content from the reference database were flagged accurately, with a clear plagiarism percentage displayed to the user. Additionally, the system highlighted the matching sections, allowing users to visually identify problematic areas within the submitted text.

In terms of performance, the system achieved fast response times, even with moderately large documents, due to efficient text preprocessing and vectorization. The use of Flask for the web interface ensured a smooth and user-friendly experience, with successful uploads and result retrieval across multiple browsers and devices. The system's ability to differentiate between original and altered phrases provided a clear advantage over traditional keyword-based methods, which often fail to detect reworded plagiarism.

Testing also revealed that the accuracy of plagiarism detection improves with the expansion of the reference document database. As the corpus of comparison texts grows, the system becomes more robust in detecting subtle instances of reused content. However, like most NLP-based systems, detection accuracy may vary slightly for documents that contain highly specialized jargon or non-standard language.

Overall, the results confirm that the proposed system is effective, reliable, and well-suited for educational and professional use. It not only aids in maintaining academic integrity but also empowers users with a tool to self-assess and improve the originality of their work before submission.

6.2TEST OBJECTIVES

These are several rules that can save as testing objectives they are: Testing is a process of executing program with the intent of finding an error. A good test case is one that has a high probability of finding an undiscovered error. If testing is conducted successfully according to the objectives as stated above it would in cover errors in the software also testing demonstrator that software functions appear to the working according to specification that performance requirements appear to have been met.

CHAPTER 7

RESULT AND DISCUSSION

7.1RESULT

The AI-powered plagiarism detection system was successfully implemented and tested using a diverse set of documents, including academic essays, research articles, and general content with varying degrees of similarity. The system demonstrated strong performance in identifying both exact text matches and paraphrased content, thanks to the integration of TF-IDF vectorization and cosine similarity for semantic comparison. Documents containing copied content from the reference database were flagged accurately, with a clear plagiarism percentage displayed to the user. Additionally, the system highlighted the matching sections, allowing users to visually identify problematic areas within the submitted text.

In terms of performance, the system achieved fast response times, even with moderately large documents, due to efficient text preprocessing and vectorization. The use of Flask for the web interface ensured a smooth and user-friendly experience, with successful uploads and result retrieval across multiple browsers and devices. The system's ability to differentiate between original and altered phrases provided a clear advantage over traditional keyword-based methods, which often fail to detect reworded plagiarism.

Testing also revealed that the accuracy of plagiarism detection improves with the expansion of the reference document database. As the corpus of comparison texts grows, the system becomes more robust in detecting subtle instances of reused content. However, like most NLP-based systems, detection accuracy may vary slightly for documents that contain highly specialized jargon or non-standard language.

7.2CONCLUSION

In conclusion, the proposed AI-powered plagiarism detection system offers an effective and intelligent solution for identifying content similarity at the document level. By leveraging advanced Natural Language Processing (NLP) techniques such as text preprocessing, TF-IDF vectorization, and cosine similarity, the system accurately detects both direct copying and paraphrased content. The integration of these methods enables a more nuanced analysis of textual data, going beyond surface-level matches to capture semantic similarities. Developed using the Flask web framework, the application ensures user accessibility through a simple and interactive interface, making it suitable for academic institutions, content creators, and professional organizations. The system not only promotes originality and integrity but also empowers users to evaluate and improve the uniqueness of their work before final submission. Overall, the project successfully demonstrates the potential of AI and NLP in automating plagiarism detection with transparency, efficiency, and precision.

7.3FUTURE ENHANCEMENT

The current AI-powered plagiarism detection system can be further enhanced to improve its accuracy, scalability, and user experience. One potential enhancement is the integration of advanced deep learning models like BERT or RoBERTa, which would offer a deeper contextual understanding of text and significantly improve the detection of paraphrased content.

Additionally, expanding the system to support multiple languages would broaden its applicability across diverse academic and professional settings, making it useful on a global scale. Another area of improvement could be the implementation of real-time web crawling, enabling the system to compare documents against dynamic online content and continually update its reference database. To detect more complex plagiarism techniques, future versions could incorporate cross-document analysis, allowing the system to identify content fragments copied from multiple sources. For greater functionality, adding user authentication and the ability to generate detailed reports could help institutions track plagiarism instances and provide actionable insights.

It Enhances the visualization tools by offering features like side-by-side document comparisons or similarity heatmaps could make the results more intuitive and user-friendly. Furthermore, deploying the system on a cloud platform would provide the scalability required for handling larger datasets and increased user loads, making it suitable for enterprise or institutional deployment. Finally, developing a mobile application would allow users to check documents for plagiarism on the go, improving accessibility and convenience. These enhancements would transform the system into a more powerful, flexible, and widely adopted tool for plagiarism detect

APPENDIX A
SOURCE CODE

Index.html

```
<!DOCTYPE html>

<html>

<head>

<title>Plagiarism Checker</title>

<style>        body {            font-

family: Arial;           background-

color: #CDFCF6;           padding:

50px;           height: 100vh;
```

```
        }         form {              background:
#F5EFFF;              padding: 25px;
border-radius: 8px;              box-shadow: 0 0
10px rgba(0,0,0,0.1);              width: 500px;
margin: auto;              box-shadow: 0 0 15px
black;
        }
        input[type=file], input[type=submit] {
margin: 10px 0;              padding: 8px;
width: 100%;


        }
        input[type=submit] {
background: #007BFF;
color: white;              width:
300px;              border: none;
justify-content: center;
display: flex;
        }        .dd{
display: flex;              justify-
content: center;              align-
items: center;              height:
```

```
100vh;            margin: 0;
border-radius: 15px;
        }
</style>
</head>
<body>
<div class="dd">
<form action="/check" method="post" enctype="multipart/form-data">
<h2>Upload Document to Check for Plagiarism</h2>
<label>Upload File (.docx):</label>
<input type="file" name="upload" required><br>
<input type="submit" value="Check Plagiarism">
</form>
</div>
</body>
</html>
```

Login.html

```
<!DOCTYPE html>
<html>
<head>
<title>Login - Plagiarism Checker</title>
<style>
```

```
        .main{     text-align:
           center;

           color: black;
        }        body {             font-
family: Arial, sans-serif;
background: #C9D6DF;
           display: flex;
justify-content: center;
align-items: center;
height: 100vh;
        }
        .head{            text-
align: center;
font-size: 30px;
font-weight: bold;
padding-top: 0px;
        }        form {             background-color:
#F0F5F9;            padding: 30px;
border-radius: 20px;            box-shadow: 0 0
15px rgba(0,0,0,0.2); width: 500px;
        }
        .us{                    padding-
bottom: 15px;                   font-
weight: bold;
        }
        #inp{             border-radius: 10px;
margin-right: 20px;             width: 350px;
box-shadow: 0 0 15px rgba(0,0,0,0.2);
        }
        input[type=text], input[type=password] {
width: 100%;             padding: 8px;
margin: 8px 0;             border: 1px solid #ccc;
        }
        input[type=submit] {
background: #2b909b;
color: white;             padding:
10px; border: none;
            cursor: pointer;
border-radius: 11px;
width: 150px;             font-
size: 20px;
        }
        .button:hover{
background-color: #A66CFF;
        }        p {
color: red;
        }
</style>
</head>
<body>
<div class="bg-container">
<h1 class="main">Plagiarism Checker</h1>
<form method="post" action="/login">
<h2 class="head">Login</h2>
<label class="us">Username:</label><br><br>
<input    type="text"    name="username"    placeholder="Enter    the    UserName"
id="inp"><br><br>
<label class="us">Password:</label><br><br>
<input   type="password"   name="password"   placeholder="Enter   the   Password"
id="inp"><br><br>
<div style="text-align: center;"><br>
<input type="submit" class="button us" value="Submit">
</div>
</form>
```

```
</div>

</body>

</html>
```

Result.html

```
<!DOCTYPE html>

<html>

<head>

<title>Plagiarism Result</title>

<style>         body {

font-family: Arial;

padding: 30px;

background: #f8f9fa;

        }

        .result                     {

background:            white;

padding:                   25px;

border-radius:           10px;

box-shadow:   0   0   10px

rgba(0,0,0,0.1);

margin-top: 30px;

        }                  span  {

font-weight: bold;
```

```
        }
        #container {
background-color: #fff;
border-radius: 8px;
width: 420px;             padding:
50px 0;             display: flex;
flex-direction: column;
align-items: center;
margin-left: 500px;
        }
        #circle-progress {
position: relative;
height: 250px;             width:
250px;          border-radius:
50%;          background:
conic-gradient(rgb(210, 100,
237) 0deg, #ededed 0deg);
            display: flex;
justify-content: center;
align-items: center;
        }
```

```
#circle-progress::before {
content: "";            position:
absolute;            width:
210px;            height: 210px;
border-radius: 50%;
background-color: #fff;
        }
        .progress-value {
position: relative;            font-
size: 50px;            color:
rgb(102, 37, 166);
        }
</style>
</head>
<body>
    <div id="container">
    <div id="circle-progress">
    <span class="progress-value">0%</span>
    </div>
<span       class="text"><strong>Matched       With:</strong>       {{       match_file
}}</span></div>
```

```
<div class="result">

<h2>Plagiarism Result</h2>

<p><strong>Matched With:</strong> {{ match_file }}</p>

<p><strong>Plagiarism Percentage:</strong> {{ result }}%</p>

<hr>

<h3>Highlighted Document</h3>

<div style="white-space: pre-wrap;">{{ highlighted|safe }}</div></div>


<script>window.onload = function () {            const result = parseInt("{{

result }}");             constcirclepro = document.getElementById("circle-

progress");            constprogressvalue =

document.querySelector(".progress-value");        let start = 0;        const

speed = 30;


   const progress = setInterval(() => {               if (start <= result) {

progressvalue.textContent = `${start}%`;                  const angle = start *

3.6;                  circlepro.style.background = `conic-gradient(rgb(210, 100,

237)

${angle}deg, #ededed 0deg)`;

                 start++;               }

else {

clearInterval(progress);
```

```
            }
        }, speed);
    };
</script>
</body>
</html>
```

app.py

```
from flask import Flask, render_template, request, redirect, session import os from
utils import extract_text_from_docx, calculate_plagiarism, highlight_matches app
= Flask(__name__) app.secret_key = 'your_secret_key'

UPLOAD_FOLDER = 'uploads'
REFERENCE_FOLDER = 'reference_docs'

@app.route('/') def
login():
    return render_template('login.html')

@app.route('/login',       methods=['POST'])
defdo_login():
    username = request.form['username']
password = request.form['password']     if username
== 'admin' and password == 'admin':
        session['logged_in'] = True         return redirect('/home')
return render_template('login.html', error="Invalid credentials")


@app.route('/home') def
home():

if not session.get('logged_in'):
        return   redirect('/')              return
render_template('index.html')


@app.route('/check', methods=['POST'])
def     check():                        if     not
session.get('logged_in'):
        return redirect('/')


    file = request.files['upload']      if file and
file.filename.endswith('.docx'):
```

```
upload_path = os.path.join(UPLOAD_FOLDER, file.filename)
file.save(upload_path)        uploaded_text =
extract_text_from_docx(upload_path)

max_score = 0
matched_file = ""
matched_text = ""

        for ref_file in os.listdir(REFERENCE_FOLDER):
            if ref_file.endswith('.docx'):
ref_path = os.path.join(REFERENCE_FOLDER, ref_file)                    ref_text =
extract_text_from_docx(ref_path)                score = calculate_plagiarism(ref_text,
uploaded_text)                  print(f"Compared with {ref_file} → Score: {score}")  #
Debug
                if score >max_score:
max_score = score
matched_file = ref_filematched_text =
ref_text

highlighted_result = highlight_matches(matched_text, uploaded_text)        return
render_template('result.html', result=max_score, match_file=matched_file,
highlighted=highlighted_result)
```

```
    return redirect('/home')


if __name__ == '__main__':
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(REFERENCE_FOLDER, exist_ok=True)     app.run(debug=True)
```

APPENDIX B

SCREENSHOTS

Sample Output

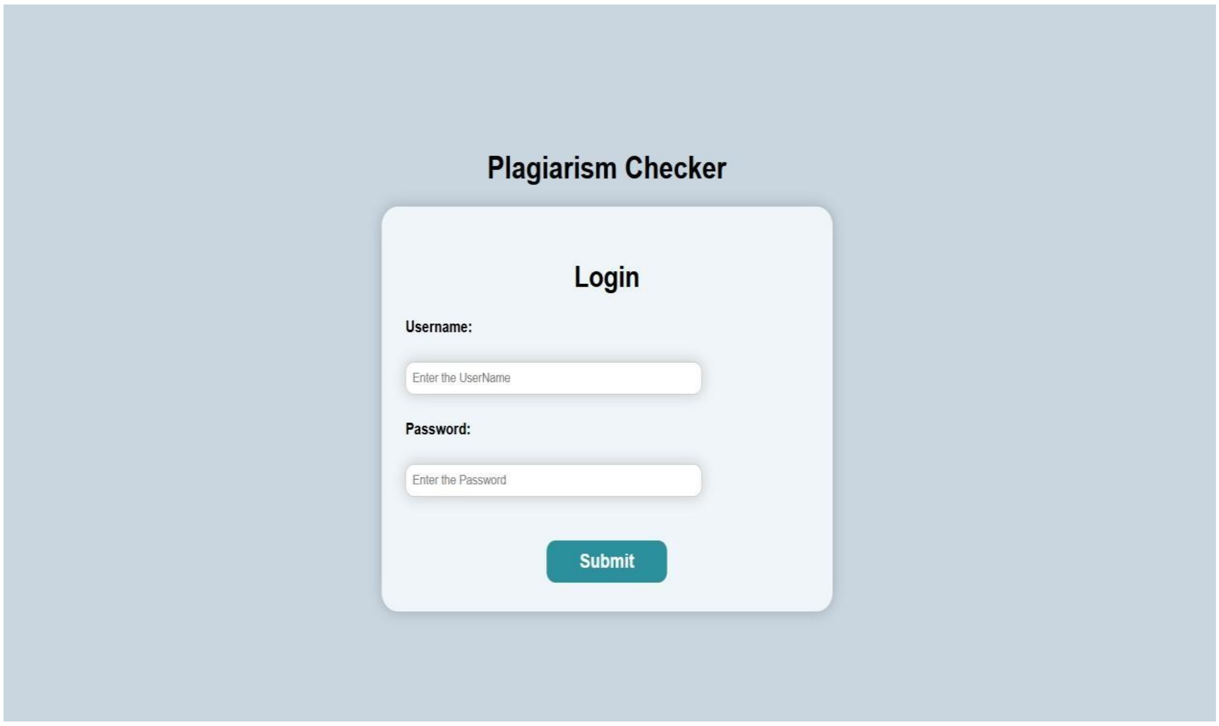


Fig 2.1: Login Page

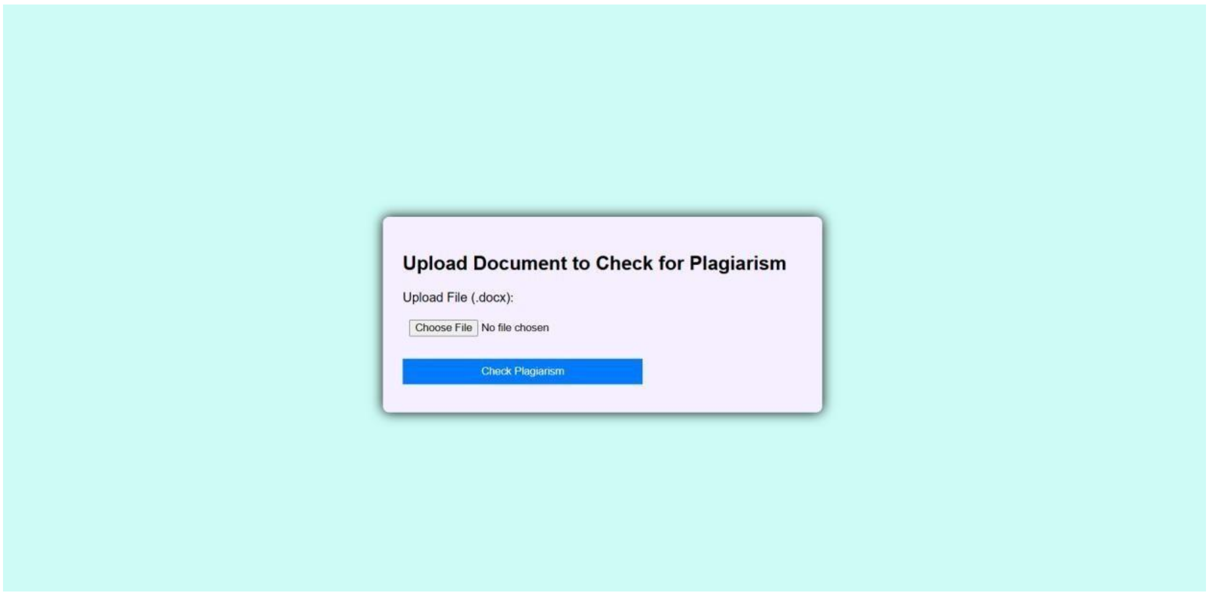


Fig 2.2: Document Upload Page

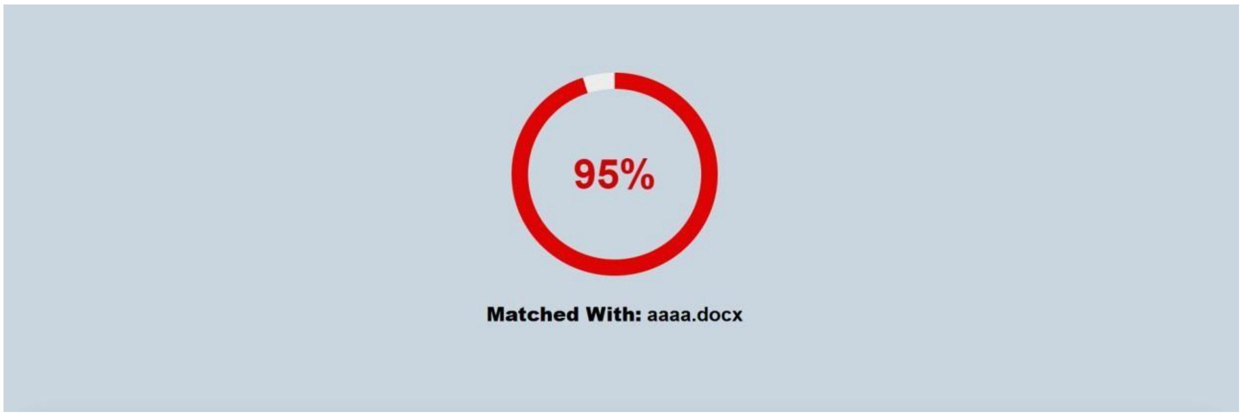
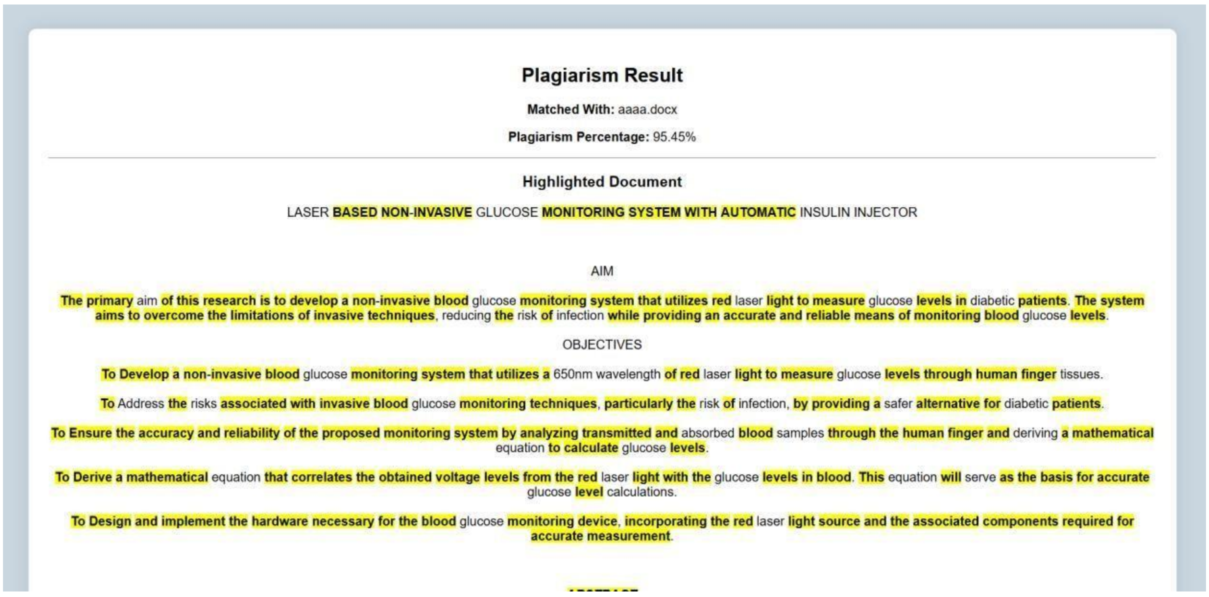


Fig 2.3 Document-wise Plagiarism Percentage



Plagiarism Result

Matched With: aaaa.docx

Plagiarism Percentage: 95.45%

Highlighted Document

LASER BASED NON-INVASIVE GLUCOSE MONITORING SYSTEM WITH AUTOMATIC INSULIN INJECTOR

AIM

The primary aim of this research is to develop a non-invasive blood glucose monitoring system that utilizes red laser light to measure glucose levels in diabetic patients. The system aims to overcome the limitations of invasive techniques, reducing the risk of infection while providing an accurate and reliable means of monitoring blood glucose levels.

OBJECTIVES

To Develop a non-invasive blood glucose monitoring system that utilizes a 650nm wavelength of red laser light to measure glucose levels through human finger tissues.

To Address the risks associated with invasive blood glucose monitoring techniques, particularly the risk of infection, by providing a safer alternative for diabetic patients.

To Ensure the accuracy and reliability of the proposed monitoring system by analyzing transmitted and absorbed blood samples through the human finger and deriving a mathematical equation to calculate glucose levels.

To Derive a mathematical equation that correlates the obtained voltage levels from the red laser light with the glucose levels in blood. This equation will serve as the basis for accurate glucose level calculations.

To Design and implement the hardware necessary for the blood glucose monitoring device, incorporating the red laser light source and the associated components required for accurate measurement.

Fig 2.4 Plagiarism highlighted text

REFERENCES

1. D. S. Morris, ‘‘Automatic grading of student’s programming assignments: An interactive process and suite of programs,’’ in Proc. 33rd Annu. Frontiers Educ. (FIE), vol. 3, 2003, pp. 1–6.
2. S. Rajesh, V. V. Rao, and M. Thushara, ‘‘Comprehensive investigation of code assessment tools in programming courses,’’ in Proc. IEEE 9th Int. Conf.

   Converg. Technol. (I2CT), Apr. 2024, pp. 1–6.
3. M. Messer, N. C. C. Brown, M. Kölling, and M. Shi, ‘‘Automated grading and feedback tools for programming education: A systematic review,’’ ACM Trans. Comput. Educ., vol. 24, no. 1, pp. 1–43, Mar. 2024.
4. M. Lan and K. F. Hew, ‘‘Examining learning engagement in MOOCs: A self- determination theoretical perspective using mixed method,’’ Int. J. Educ. Technol. Higher Educ., vol. 17, no. 1, p. 7, Dec. 2020.

5. M. Jukiewicz, “The future of grading programming assignments in education: The role of ChatGPT in automating the assessment and feedback process,” Thinking Skills Creativity, vol. 52, Jun. 2024, Art. no. 101522.
6. C. K. Shyamala, C. S. Velayutham, and L. Parameswaran, “Teaching computational thinking to entry-level undergraduate engineering students at Amrita University,” in Proc. IEEE Global Eng. Educ. Conf. (EDUCON), Apr.
2017, pp. 1731–1734.
7. R. K. Raj, B. A. Becker, M. Goldweber, and P. Jalote, “Perspectives on computer science curricula 2023 (CS2023),” in Proc. ACM Conf. Global Comput. Educ., vol. 2, Dec. 2023, pp. 187–188.
8. B. Cheang, A. Kurnia, A. Lim, and W.-C. Oon, “On automated grading of programming assignments in an academic institution,” Comput. Educ., vol.
41, no. 2, pp. 121–131, Sep. 2003.
9. M. Zakeri-Nasrabadi, S. Parsa, M. Ramezani, C. Roy, and M. Ekhtiarzadeh, “A systematic literature review on source code similarity measurement and clone detection: Techniques, applications, and challenges,” J. Syst. Softw., vol. 204, Oct. 2023, Art. no. 111796.
10. L. Van Praet, J. Hoobergs, and T. Schrijvers, “ASSIST: Automated feedback generation for syntax and logical errors in programming exercises,” in Proc.
ACM SIGPLAN Int. Symp. SPLASH-E, Oct. 2024, pp. 66–76.