PLAGIARISM DETECTION USING MACHINE LEARNING

MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Science and Engineering)

Submitted By:

Sahil Guleria (2104173)

Harnoor Singh (2203582)

Kaushal Kundan (2203586)

Submitted To:

Dr. Hardeep Singh Kang

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE

LUDHIANA, 141006

June, 2025

PLAGIARISM DETECTION USING MACHINE LEARNING

MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Science and Engineering)

Submitted By:

Sahil Guleria (2104173)

Harnoor Singh (2203582)

Kaushal Kundan (2203586)

Submitted To:

Dr. Hardeep Singh Kang

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE

LUDHIANA, 141006

June, 2025

ABSTRACT

Plagiarism is a serious problem in academic and professional settings, where people copy the work of others and present it as their own. The extensive use of the internet and online media has increased occurrences of plagiarism. Machine learning provides solutions to this problem by examining textual data and comparing it with a repository of known sources for similarity identification. This project proposes a machine learning based plagiarism detection approach. Utilizing Natural Language Processing (NLP) techniques, the system assesses text similarity and generates a similarity score.

Prior to analysis, the data goes through some preprocessing steps for quality and consistency. The findings show that the model is efficient in detecting plagiarism and can be implemented as a tool to aid academic integrity. The study mainly deals with plagiarism in institutes, where students tend to copy others' work for assignments. An educators' system would be able to simplify the process of checking students' work for originality, which is an innovative solution in place of manual checks. It lessens the possibility of idea theft and encourages honest academic practices. Plagiarism has been considered a breach of moral rights in many nations, and its prevalence has grown with the increasing availability of digital content.

In light of this, the demand for effective plagiarism detection tools has grown. These systems are essential in sustaining scholarly standards and academic credibility. The objective of this project is to develop a free-of-charge plagiarism detection tool appropriate for research and educational purposes. Using sophisticated algorithms and language analysis, the software can detect a variety of plagiarism types—ranging from direct copying to more sophisticated types such as paraphrasing and mosaic plagiarism. Though useful, plagiarism detection software has some drawbacks. Most are not readily accessible because they are very expensive, limiting their use among wider groups. Additionally, computer systems can occasionally mislabel content, necessitating human intervention to properly interpret whether information is plagiarized or properly referenced.

Acknowledgement

We are highly grateful to the Dr. Sehijpal Singh, Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the major project work at Plagiarism Detection Using Machine Learning.

The constant guidance and encouragement received from Dr. Kiran Jyoti H.O.D. CSE Department, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to Dr. Hardeep Singh Kang, without his wise counsel and able guidance, it would have been impossible to complete the project in this manner.

We express gratitude to other faculty members of computer science and engineering department of GNDEC for their intellectual support throughout the course of this work.

Finally, we are indebted to all whosoever have contributed in this report work.

Sahil Guleria (2104173)

Harnoor Singh (2203582)

Kaushal Kundan (2203586)

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Introduction to Project

Plagiarism is the copying of another's intellectual work—in the form of written content, creative concepts, or research—presenting someone else's output as your own without due crediting or permissions. This academically irregular act goes contrary to the morals of honesty, originality, and intellectual integrity. Sadly enough, this uncouth practice is increasingly prevalent within different academic as well as innovative disciplines such as essay writing, research papers, and artistic creativity, which depicts a problem people tend to remain unaware of.

The word "plagiarism" derives from Latin, where it denotes "kidnapper" or "to steal." In schools, plagiarism detection is important because students, scholars, and educators are supposed to come up with original work in their studies.

It's worth mentioning that although the utilization of machine learning technologies to produce content might seem to be a gray area, its utilization without valid licensing or permission for academic assessment is considered academic dishonesty. Moreover, self-plagiarism, or the reuse of one's own already published material without appropriate citation, is also unethical. In addition, academic tests have rigorous anti-plagiarism policies, and offenses usually lead to severe disciplinary measures.

The best way to prevent plagiarism is to learn academic integrity standards early in your university career. Not preventing plagiarism means not only getting citations right but also rewriting so that plagiarism can't be detected; it means applying your academic skills to create high-quality original work.

To prevent and deter plagiarism, numerous educational institutions and professional organizations have strict policies and penalties in place. Proper source attribution—either through direct quotations, paraphrasing, or summarizing—is a must, as is adhering to the guidelines of citation style offered by your institution or publisher. This serves to ensure your work is still ethically secure.

To meet the increasing demand for detecting plagiarism, the objective of this project is to develop a free of cost plagiarism detection tool for educational and research purposes. Such tools apply sophisticated algorithms and linguistic analysis methods to detect different types of plagiarism from direct copying to the more insidious forms such as paraphrasing or mosaic plagiarism.

While plagiarism checkers are helpful, there are some limitations. First, they are not always widely available, and the ones that are available are usually very expensive, which limits accessibility to the public at large. Second, sometimes these tools will give a false positive or a false negative reading, which means human intuition is still needed to correctly identify true plagiarism versus correctly cited material.

To ensure originality and adhere to ethical principles in academic and professional work, there is a need to utilize plagiarism detection software along with good research skills and writing culture.

Down below, are enumerated 8 prominent types of Plagiarism methods employed by students and researchers routinely:

1. Verbatim quotation: Such statements need to be indicated as such with proper citations of the original works and either quotation marks or indentation. Always make it clear to the reader what ideas and words you have directly borrowed from some other source and what you have devised yourself.

2. Cut Paste from internet: Anything gathered from the Internet needs to be referenced properly and included in the bibliography. It is crucial to critically review anything you come across on the Internet since what you find there is less likely to have gone through the same thorough scrutiny as conventional sources.

3. Paraphrasing: Paraphrasing is plagiarism when the writer whose work you are borrowing is not adequately credited. This involves rearranging a couple of words and their order in addition to rigorously adhering to the structure of the argument.

4. Collusion: Students can collude without authorization, forget to acknowledge help given, or not strictly adhere to the group work project guidelines. It is your job to make sure you know how much you are allowed to collaborate and what of the work needs to be completely original.

5. Wrong Citation: As per the regulations of your profession, it's crucial to cite sources correctly. Not only should you cite your sources (i.e., have a bibliography), but you also

have to name the sources in a footnote or in-text citation. You also shouldn't have in your bibliography anything or anyone that you didn't actually use.

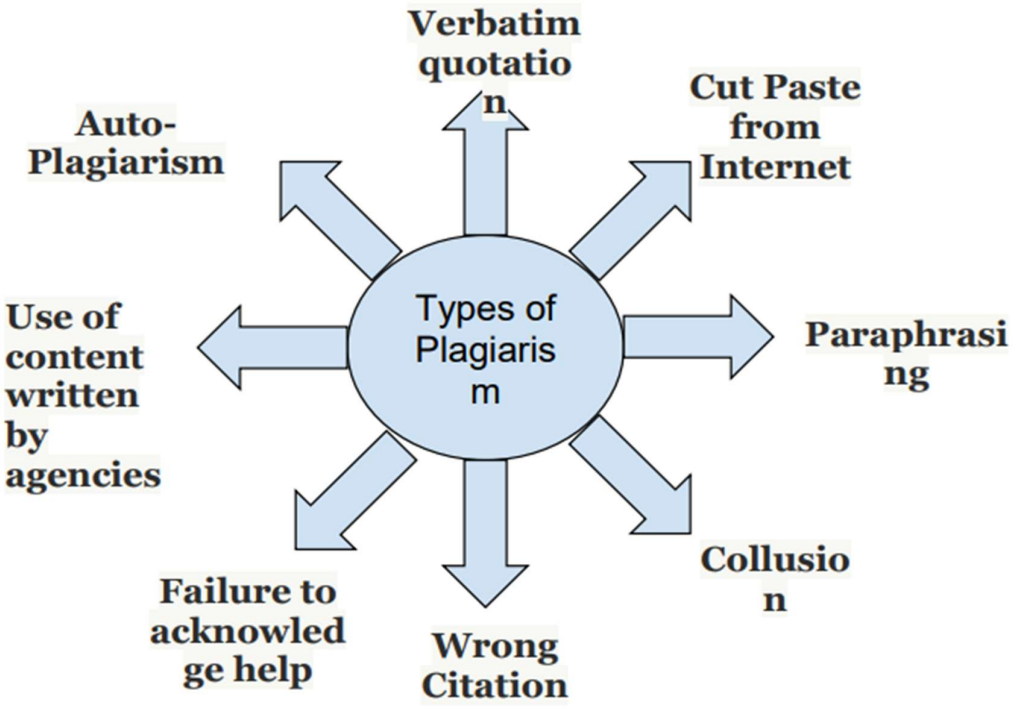


Figure 1.1 - Types of Plagiarisms

6. Failure to acknowledge help: All input provided to the development of your work, such as ideas from other students, laboratory staff, and external sources, should be recognized. It is important to note any additional advice that results in material alterations in the content or method, even though it is not related to your supervision or tutoring or to mere amendment.

7. Use of content written by agencies: Never turn in work completed for you by a professional firm or utilize their services in the development of your work, not even with the permission of the author. This research process has to be done independently because it is imperative to your intellectual development and progression.

8. Auto-Plagiarism: You may not submit for assessment which you have previously submitted (either partially or entirely) for your current course or for another award of this or any other institution, except where it is explicitly mentioned in the particular regulations of your course. Any earlier work by you that is citable or publishable must be

specially cited. Same works submitted in the same way will also amount to auto-plagiarism.

In summary, plagiarism checkers are vital to maintaining ethical standards and upholding academic integrity. Such approaches apply technology to assist in prevention and detection of plagiarism, promoting a culture of originality and intellectual integrity. To keep plagiarism checkers useful amidst the evolving context of scholarly communication and information exchange, further research and innovation in this field are essential.

1.2 Project Category

The project titled Plagiarism Detection Using Machine Learning is classified as an institute-based initiative, developed entirely within the academic framework of our institution. Its primary objective is to assist in preserving academic honesty by identifying copied or unoriginal content in student assignments and documents. By applying machine learning techniques such as natural language processing and text similarity analysis, the system can detect subtle and direct forms of plagiarism. This project not only reinforces the institute’s dedication to academic ethics but also provides a practical solution that can be integrated into evaluation processes to support fair and authentic academic work.

1.3 Problem Formulation

Plagiarism is an academic integrity violation that portrays dishonesty and a lack of proper acknowledgement of other people's ideas, research, and words. It not only depicts a deficiency in academic rigor but also a poor understanding of the learning process. Plagiarism does not only lower the integrity of educational institutions but also the value of qualifications offered by those institutions. Apart from this, it also has great ethical consequences and may have lasting effects on future professional opportunities.

A passion to create excellent, original work is an inherent driving force to prevent plagiarism. Having a firm understanding of citation procedures and source use is responsibly achieved, and preventing plagiarism is easy. Furthermore, developing writing skills in this way results in cleaner, more refined work overall. It's important to appreciate that learning these academic

writing strategies is not so much a skill but an integral part of displaying intellectual integrity. This commitment strengthens the credibility and authority of one's scholarly work.

Addressing the problem of plagiarism is a collective responsibility that can result in improved standards in academic work and professional work. By upholding academic honesty, we are part of a culture of integrity and excellence.

1.4 Identification/Recognition of Need

The acknowledgment of this project is significant because it provides a solution to a common problem in schools—verifying the originality and authenticity of students' work. With the spread of digital material and sophisticated AI software, old ways of detecting plagiarism are no longer effective. This project provides a novel solution by using machine learning methods to detect both paraphrased and direct cases of plagiarism, providing a faster and more precise solution. Recognizing this effort not only brings to light its potential effectiveness in upholding academic integrity but also instigates the creation of such initiatives that can further promote the validity of educational tests.

In addition, this project is an example of the translation of theoretical knowledge gained under academic study into practical reality, showcasing the capacity to address real-world issues through technological innovation. The acknowledgment of such initiatives creates a culture of problem-solving and creativity within academia, encouraging others to undertake projects that benefit institutional objectives and societal requirements.

1.5 Existing System

The current plagiarism detection systems are based mainly on keyword searching and rule-based comparison with huge databases of web-based contents and earlier submitted work. Although these systems can identify literal copy, they do not detect more sophisticated types of plagiarism, including paraphrased and structurally modified texts. Being based on exact wording and not being context-aware, their effectiveness in academic settings is restricted. Moreover, most of these solutions are commercial and will not be available to every institution because they are either too expensive or subject to usage limitations. In spite of their usefulness, existing plagiarism detection software has serious limitations. They usually fail to comprehend meaning

and detect content that has been changed subtly, translated, or reproduced conceptually. This reduced depth of analysis makes them less reliable, particularly when assessing originality in innovative or challenging academic work. These challenges highlight the necessity for smarter, adaptive systems—such as the one being suggested in this project—that are able to learn language and context more intelligently using machine learning.

1.6 Objectives

1. To develop a web-based tool to detect plagiarism and provide report to the user.
2. To compare with available open-source plagiarism tools.
3. To study different algorithm in machine learning such as cosine similarity.

1.7 Proposed System

The proposed system is a web application developed to detect plagiarism using machine learning techniques, specifically leveraging the cosine similarity algorithm. This approach allows the system to analyze the textual similarity between user-submitted content and data retrieved via the Google Search API, enabling the detection of both direct and contextually similar or paraphrased content. The frontend of the application is built using React, ensuring a responsive and user-friendly interface that allows users to easily submit text and receive detailed plagiarism analysis results. For secure user authentication and access management, the system integrates Google Firebase, utilizing OAuth for a safe and seamless login experience. By combining machine learning-driven similarity detection with a modern web stack, this system offers an innovative solution for plagiarism detection that is both accurate and easily accessible for users.

1.8 Unique features of the proposed system

1. Machine Learning-Based Detection: The system uses the **cosine similarity algorithm**, a powerful machine learning technique, to analyze the semantic similarity between documents. This enables it to detect plagiarism that goes beyond simple keyword matching and identifies rephrased or paraphrased content.

2. **Real-Time Content Search:** By leveraging the **Google Search API**, the system can perform real-time searches across the web, comparing the submitted text against a vast pool of online resources, ensuring accurate and up-to-date plagiarism detection.

3. **React Frontend for Enhanced User Experience:** The **frontend** of the application is built with **React**, providing a smooth, interactive, and responsive interface. This allows users to easily submit documents and instantly view plagiarism analysis results in a user-friendly format.

4. **Secure User Authentication with Firebase:** The system integrates **Google Firebase** for secure **OAuth**-based user authentication, ensuring that only authorized users can access the platform and their plagiarism reports, enhancing data security and privacy.

5. **Real-Time Feedback and Reports:** Users receive **instant feedback** after submitting content, with detailed reports that include the percentage of similarity, sources found, and suggestions for improving originality, helping users understand and address potential plagiarism issues.

Chapter 2

Requirement Analysis and System Specification

2.1. Feasibility Study

The preliminary study evaluates whether a project is practical and beneficial for the organization. Its primary goal is to assess the feasibility of integrating new modules or resolving issues within an existing system. This phase verifies the technical, operational, and economic feasibility of the proposed solution. While anything may seem achievable with infinite time and resources, real-world constraints require thorough analysis in the following areas:

- Technical Feasibility
- Operation Feasibility
- Economical Feasibility

2.1.1. Technical Feasibility

This aspect focuses on whether the proposed solution is technically achievable.

The key considerations include:

- Is the required technology available to implement the proposed system?
- Do the hardware and software components have the capacity to handle the expected data and operations?
- Can the system scale to accommodate changes in the number of users or their locations while maintaining acceptable performance?
- Is the system capable of being upgraded in the future?
- Does the system ensure technical standards such as accuracy, dependability, user accessibility, and data protection?

There was no previous system available to meet the requirements of the Secure Infrastructure Implementation System. The newly designed system is technologically sound, featuring a

web-based user interface designed to facilitate audit workflows for NIC-CSD. It enables role-based access control, ensuring users have appropriate permissions based on their defined roles. The system provides high reliability, security, and accuracy. Additionally, the required hardware and software resources are already available within NIC or can be sourced through open-source solutions, making use of existing infrastructure. Adequate internet bandwidth ensures fast and efficient responses for users, regardless of system load.

2.1.2. Operational Feasibility

Recommended projects are of use only if they can be produced into information system. That will meet the operating needs of the organization. Operational feasibility aspects of the project are to be taken into account as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following:

- Do users and management support the implementation?
- Will the system function effectively after deployment?
- Could user resistance hinder system adoption?

The proposed system was designed after considering user feedback and management expectations. There are no anticipated obstacles or user opposition that would hinder its implementation. The system is tailored to organizational needs and is structured to optimize the use of computing resources, thereby improving performance and productivity.

2.1.3. Economical Feasibility

This dimension examines whether the system represents a sound financial investment. It involves a cost-benefit analysis to ensure that the expected returns justify the investment.

The proposed system is cost-effective. It requires no additional expenditure on hardware or commercial software, as it leverages existing NIC infrastructure and free open-source tools.

Given its minimal cost of implementation and reliance on in-house resources, the system demonstrates strong economic viability, ensuring that benefits outweigh associated costs.

2.2 Software Requirement Specification Document

A software requirements specification (SRS document) outlines how a software system is to be created. In simple terms, an SRS gives everyone involved a blueprint for that project. It provides top-shelf definitions for the functional and non-functional specifications of the software, and may also contain use cases that show how a user would use the system when it is finished. An SRS must contain sufficient information for the developers to finish the software defined. It not only defines the description of the software being developed but also its purpose: what the software will do and how it should work.

2.2.1 Introduction

With the advent of the digital era, the convenience of access to large pools of information has led to the widespread problem of plagiarism, a practice that goes against the canons of academic and professional integrity. Plagiarism detection tools are essential in maintaining these canons by detecting cases of content misappropriation. Conventional approaches, like rule-based systems, are limited in dealing with the dynamic nature of plagiarism. To address this challenge, this paper presents a new solution to detecting plagiarism through leveraging the capabilities of Machine Learning (ML) methodologies. The prevalence of digital information requires adaptive and automatic systems to identify subtle patterns and similarities across massive datasets. ML, whose strength lies in learning from data patterns and generating predictions, provides a promising front for improving plagiarism detection accuracy and efficiency. This work investigates the application of ML algorithms to the difficult task of separating authentic and plagiarized work in an attempt to offer a stronger and scalable solution.

2.2.2 Purpose

During the age of Internet revolution, the sudden act of plagiarism has been highest ever. Therefore, the world needs up plagiarism detectors badly. Most of the systems available in the market request personal information of the user. The objective of our team through this project is to give the user, particularly teachers and educational institutions, a freely available, easy to use plagiarism detector.

We suggest an approach to develop a plagiarism checker using in-built machine learning libraries. We concentrate on sci-kit library that has some useful and efficient machine learning algorithms. Simple techniques such as Vectorization and cosine similarity combined could be utilized in developing an efficient plagiarism checker.

2.2.3 Overview

The purpose of this document is to identify unambiguously the user requirements and clearly define both functional and non-functional requirements of car rental system. In addition, this document is intended to cover technical goals as well as objectives of the proposed System.

2.2.4. Functional Requirement:

2.2.4.1. Scope of the work

This study encompasses the exploration of various ML algorithms, including but not limited to logistic regression, support vector machines, and neural networks, to identify the most effective approach for plagiarism detection. The research also considers different feature extraction methods, such as TF- IDF and word embeddings, to capture the semantic nuances of text.

2.2.4.2. Functional Requirements

- **User Authentication**
  - Users must be able to sign up and log in using Firebase Authentication.
  - OAuth-based authentication using Google Sign-In.
  - Password reset functionality.
- **Plagiarism Detection**
  - Users can submit **text or document files (TXT, PDF, DOCX)**.
  - The system will preprocess text (tokenization, stopword removal, stemming).
  - The **cosine similarity** algorithm will compare the input text with existing datasets.

- Integration with **Google Search API** to compare input text against online sources.

➢ **Report Generation**

- A plagiarism percentage score will be displayed.
- The system will highlight **similar text portions** in the document.
- Users can download a **detailed plagiarism report (PDF format)**.

2.2.4.3. Non-Functional Requirements

➢ **Performance Requirements**

- The system should process documents and generate reports in **under 1 minute**.
- Must support **simultaneous user requests** efficiently.

➢ **Security Requirements**

- Secure authentication via Firebase OAuth.
- **Data encryption** for user submissions.
- API keys and sensitive data should be **stored securely**.

➢ **Usability Requirements**

- Simple **dashboard for text submission and results display**.
- Support for both **desktop and mobile browsers**.

➢ **Scalability**

- The system should be **scalable** to handle a growing number of users.
- Cloud-based storage for text submissions and reports.

➢ **Look and Feel Requirement**

- Guidelines for the visual design of the web-application, focusing on the user experience principles.

- Ensuring a consistent look and feel across different parts of the web application.

2.3 SDLC model

After reviewing the task requirements to be executed, the second step is problem analysis and familiarization with its context. Once the task objectives are clearly understood, the next crucial phase is the analysis of the problem and gaining familiarity with its surrounding context. This stage begins with two primary activities: examining the existing system and understanding the requirements and domain of the proposed system. Both of these are essential steps, but learning the current system forms the base for crafting accurate functional specifications and for guiding a successful system design.

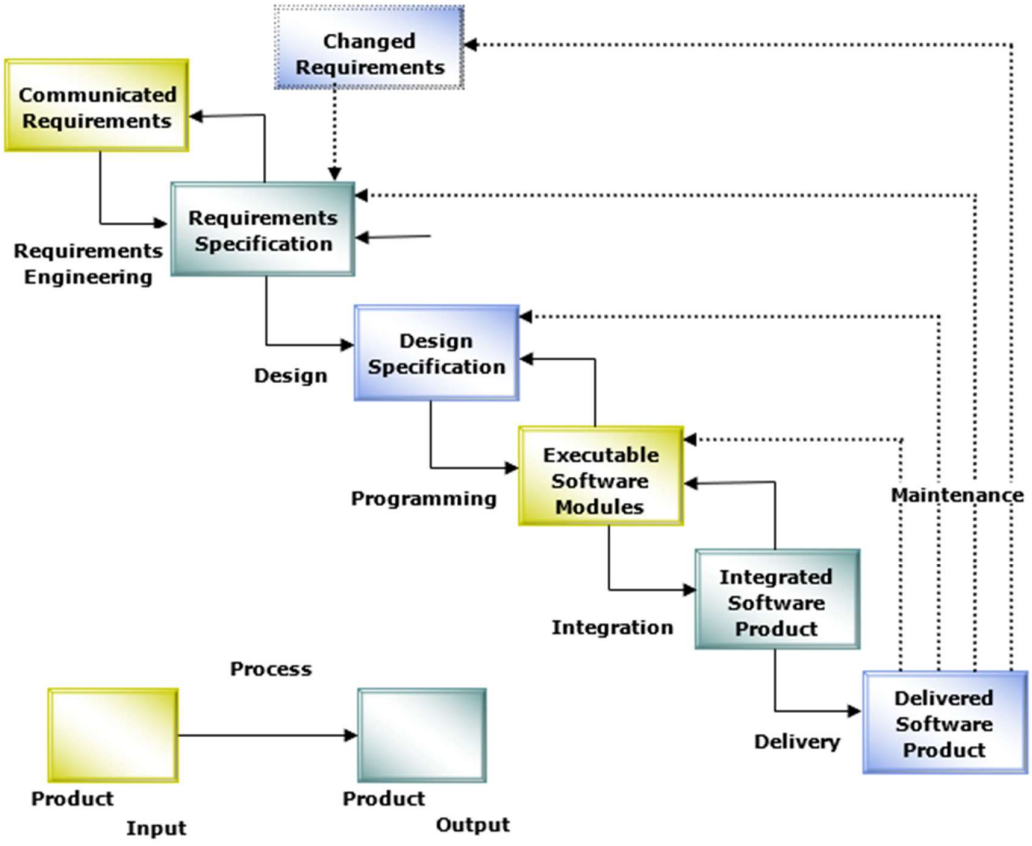


Figure 2.1 Water Fall Model

The design begins once the requirement analysis is done and the coding is done once the design is finished. The testing is carried out once the programming is done. In the model the order of activities carried out in a software development project are: -

- Requirement Analysis
- Project Planning
- System design
- Detail design
- Coding
- Unit testing
- System integration & testing

Identifying the needs and characteristics of a new system can be more complex—it requires creative problem-solving and a deep understanding of user requirements. At the same time, analyzing the current system also comes with its challenges, as any misinterpretation could lead to flaws in the final solution.

The sequential nature of these activities plays a critical role. The completion and output of each phase act as the input for the next, forming a linear development cycle. It is important that the outcome of each stage aligns with the overall system objectives. Elements from the spiral model are also reflected here, such as conducting phase reviews by project stakeholders to ensure everything is on track before moving forward.

The Waterfall Model was chosen for this project because all software requirements were clearly defined in advance, and the main goal was to digitize and automate an already established manual process.

Chapter 3

Software Design

3.1 Object-Oriented Design (OOD) Approach

Here we have a slightly different perspective on systems development than we have looked at previously. In the last one we were inclined to look at our system from a process-oriented or functional perspective, and mapped this onto the data structure. Although this method does deliver well-designed, functional systems, the prevailing view among most practitioners at present is that the resulting systems are inflexible and cause it to be hard to adapt quickly to changing user requirements.

In contrast to its two predecessors, the object-oriented approach unites data and processes (referred to as methods) into one entity known as objects. Objects typically relate to the actual things a system handles, e.g., customers, suppliers, contracts, and invoices.

Object-oriented models can represent detailed complex relationships and represent data and data processing using a sound notation, enabling a simpler combination of analysis and design in an evolving process. The Object-Oriented approach's goal is to make system components more modular and hence enhance system quality and the effectiveness of systems analysis and design.

In the Object-Oriented approach we tend to focus more on the behavior of the system. The main feature we document is the Object or Class.

What is an Object?

An Object, as we have already stated, is something we hold data about. This is usually in the form of a noun, e.g. an apple is an object. It has attributes such as size and color and taste.

What is a Class?

A Class is the description of a set of common objects, e.g. the apple would belong to the class Fruits, which all have similar group characteristics but also wide differences between them: i.e. another object, Orange, would also be part of the Fruit class but has different attributes of taste, color and size from an apple.

3.2 Detailed Design

3.2.1 Flowchart

Sci-kit-learn is a library incorporated into the system for machine learning utilities. It comprises machine learning and statistical modeling utilities. It has been implemented in the system under consideration to extract features from text. Tf-idf vectorizer is implemented to perform word embedding, i.e., textual data conversion to an array of numbers.

This transformed version of text data to the vector form is now used to identify similarity between two files of text. Cosine similarity calculates the cosine of the angle between the two vector representations of text files. This calculation produces a score from 0-1, thereby giving us information about the level of similarity between the two given input files.

The implementation approach involves four crucial steps that includes,

1) Input File

The input to the plagiarism detection system must be a text file, typically with a .txt, .pdf, .docx extension, to ensure compatibility.

2) Vectorization of text

Using Sci-kit's built-in features, the words from the text input are transformed into a vector format, enabling further analysis.

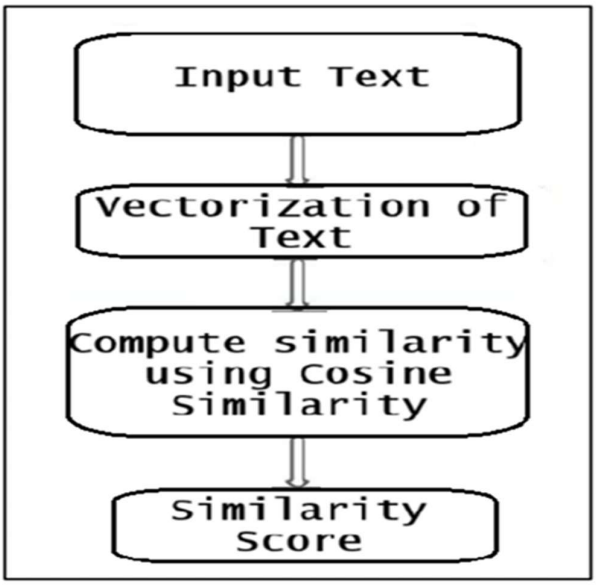


Figure 3.1 Flowchart of the Plagiarism Detection

3) Compute similarity

The similarity between two text files is computed using Cosine Similarity. This method determines how similar two vectorized text files are by calculating the dot product of their corresponding vectors, represented by cos(θ), where θ is the angle between the vectors.

4) Similarity Score

A similarity score is then derived, quantifying the degree of similarity between the two text files. The score ranges from 0 to 1, with positive values of cos(θ) indicating varying levels of similarity.

3.2.2 Use Case Diagram

A Use Case Diagram within the Unified Modeling Language (UML) is a behavioral diagram that is described by and constructed from a Use-case analysis. The intention is to provide a graphical representation of the functionality of a system in the form of actors, goals of the actors (expressed as use cases), and the relationships between the use cases.

The primary aim of a use case diagram is to indicate what system functions are executed for which actor. System actor roles may be represented. Interaction between actors is not indicated on the use case diagram. When interacting with each other is fundamental to a consistent description of the desired behavior, possibly the system or use case boundaries need reconsideration. Alternatively, interaction between actors can be included in the assumptions applied in the use case.

Use cases: A use case captures a series of steps that deliver something of quantifiable utility to an actor and is represented as a horizontal ellipse.

Actors: An Actor represents a kind of role enacted by an object that communicates with the subject (e.g., by sending signals and information), but which is outside the subject (i.e., in the sense that an object of an actor is not a component of the object of its corresponding subject). Actors can represent roles enacted by human users, peripheral devices, or other. subjects. Observe that an actor is not necessarily a particular physical object but rather a particular aspect (i.e., "role") of some object that is of interest to the specification of its corresponding use cases. A

physical instance may, therefore, fill the role of multiple different actors and, alternatively, a particular actor may be filled by multiple different instances.

Association: An association defines a semantic relationship that can exist between typed instances. An association has at least two ends represented by properties, each of which is related to the end type. Multiple ends of the association can be of the same type.

System boundary boxes (optional): The use cases are enclosed in a rectangle, termed the system boundary box, to mark the boundaries of system. Everything inside the box is considered in-scope functionality and anything outside the box is not. Four types of relationships between use cases are used frequently in practice.

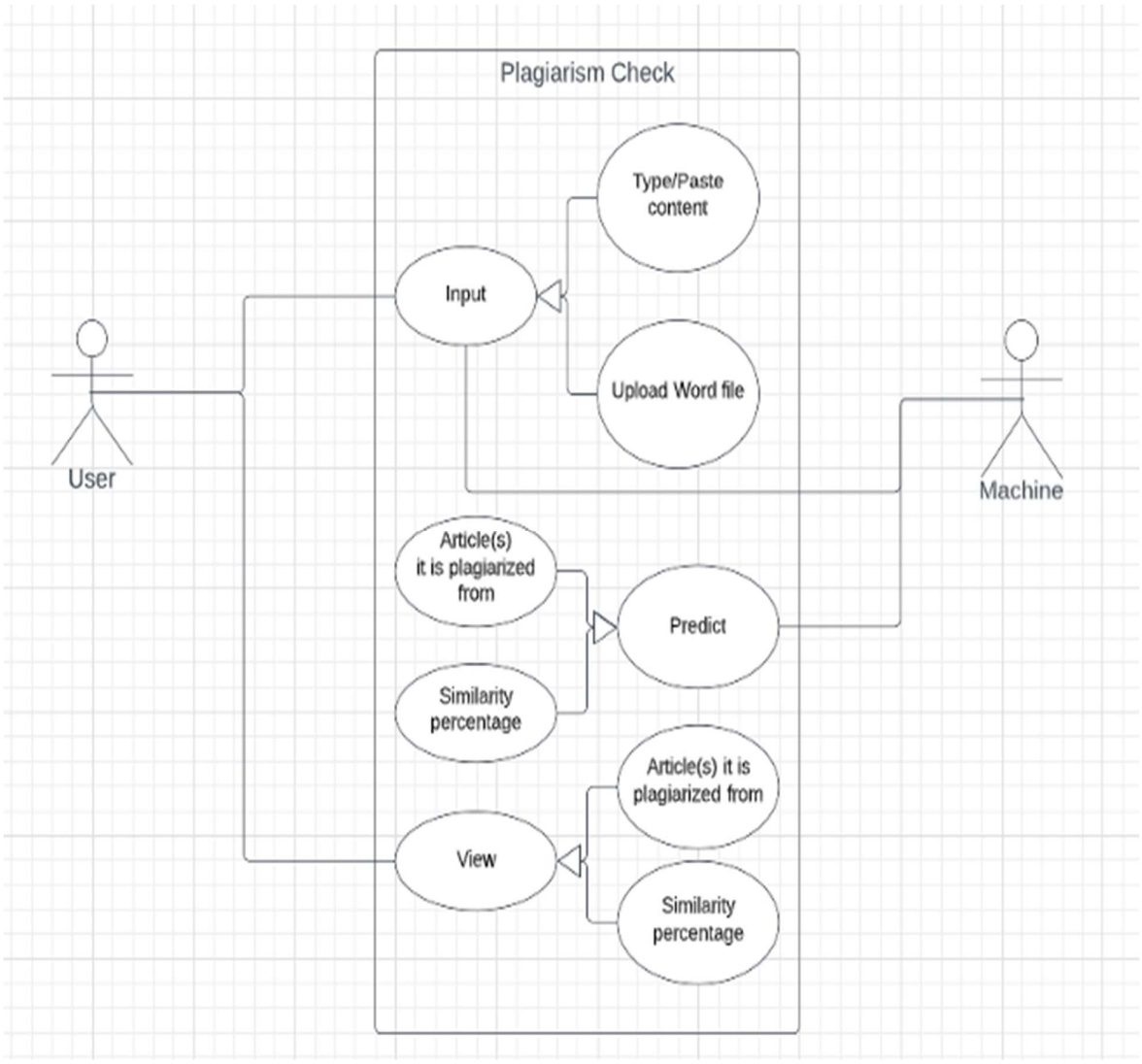


Figure: 3.2 Use case diagram

a. **Include:** One type of relationship between use cases is the include relationship. This occurs when one use case incorporates the behavior of another. An include relationship is a directed link between two use cases, signifying that the behavior of the included use case is inserted into the behavior of the including use case. Typically, the execution of the primary use case depends on the outcome of the included use case. This mechanism is useful for extracting and reusing shared behaviors across multiple use cases. In UML notation, this is represented by a dashed arrow pointing from the including use case to the included one, labeled with «include». No parameters or return values are passed. To indicate the inclusion, point in an event sequence, simply mention include followed by the included use case's name—for example, in a process like tracking an order.
b. **Extend:** Another form of use case interaction is the *extend* relationship. In this case, an extending use case may introduce additional behavior to an existing base use case under specific conditions. This is symbolized by a dashed arrow from the extending use case to the extended one, marked with «extend». This relationship is often used to represent optional or conditional functionality that enhances the base use case when certain criteria are met.
c. **Generalization:** The third kind of relationship is generalization, which also applies to use cases. Here, a specific use case inherits the characteristics—such as behavior, requirements, or constraints—of a more general one. These shared elements are defined once in the general use case, and any differences are addressed within the specialized cases. In UML, this is shown with a solid line ending in an open triangle pointing from the specialized use case to the general one, following standard generalization notation.
d. **Associations:** Associations represent connections between actors and use cases in use case diagrams and are depicted as solid lines. An association exists whenever an actor is involved in a use case's interaction. These lines can include an optional arrowhead, which typically indicates the direction of the first interaction or identifies the primary actor in the use case.

Identified Use Cases: The user model view captures both the problem and its resolution from the perspective of the individuals experiencing the issue. It reflects the objectives and needs of the problem stakeholders, emphasizing what they aim to achieve through the solution. This viewpoint is illustrated using use case diagrams, which detail the services a system offers to external entities. These diagrams include actors, the use cases themselves, and the relationships between them.

3.2.3 Class Diagram

A Class Diagram is a type of Structure Diagram commonly used to identify the potential classes within a system. It defines the attributes and operations each class should have and illustrates how these classes interact with one another—essentially mapping out the relationships between different classes in the system.

In modeling, objects represent entities that encapsulate three main elements: state (data), behavior (methods or procedures), and identity (a unique presence among other objects). The behavior and structure of these objects are governed by classes, which act as templates or blueprints for creating specific objects. When an object is instantiated from a class, it becomes a concrete example of that class. Unlike basic structures, objects include additional features such as method references, controlled access to members, and an implicit mechanism that enables them to locate instances of the same class within the hierarchy—this is essential for supporting features like runtime inheritance.In the realm of software development, a class diagram is a static structural diagram that conveys the architecture of a system. It does so by representing the system's classes, their attributes, and the associations among them.

The class diagram serves as a core element in object-oriented modeling. It supports both high-level conceptual modeling—helping visualize the business logic—and detailed design modeling, assisting in the translation of designs into source code. Each class in a class diagram symbolizes either a key entity or a significant interaction within the system, including those that will be implemented in the codebase. These classes are visually represented using boxes divided into three distinct sections:

- The top section displays the class name.
- The middle section lists the class's attributes.
- The bottom section shows the operations or methods the class can perform.

Figure 3.3: Class Diagram

In our proposed system, we have two classes: User and Machine. The User has only one attribute, i.e., Input, and the operations that they can perform are to give input, view the plagiarized percentage and content after evaluation. Coming to Machine class, its attributes are plagiarized percentage and content. The operations that our machine can perform are creating metadata (Vector Search Index, Aka Vector Database), performing similarity checks, return plagiarized percentage and the plagiarized content.

3.2.4. Data Flow Diagram

It is a method of showing a flow of a data of a system or process (most often an information system). The DFD also shows information about the output and input of every entity and the process itself. There is no control flow in a data-flow diagram, there are no decision rules and no loops. Certain operations based on the data can be shown by a flowchart.

Data flow diagram is the beginning of the design stage that functionally breaks down the requirements specification. A DFD is a series of bubbles that are connected with lines. Bubbles symbolize data transformation while lines symbolize data flows within the system. A DFD explains what data flow and not how they process, therefore does not involve hardware, software and data structure.There are a few notations to represent data-flow diagrams. The above notation was introduced in 1979 by Tom DeMarco as part of Structured Analysis.

For each data flow, there must be at least one of the ends (source and/or destination) within a process. The more detailed representation of a process may be achieved in another data-flow diagram, dividing this process into sub-processes.

A data-flow diagram (DFD) is a diagram representing the "flow" of information throughout an information system. Data flow diagrams (DFDs) may be used also to visualize the data processing (structured design). A data flow diagram (DFD) is a notable modelling tool to use to analyze and build information processes. DFD means literally a picture describing the direction or progression of information within a process. DFD shows such an information flow in a process dependent on inputs and outputs. A DFD can also be known as a Process Model.

The data-flow diagram is one of the structured-analysis modelling tools. While working with UML, the activity diagram usually replaces the role of the data-flow diagram. One special type of data- flow plan is a site-oriented data-flow plan. There are seven rules for construct a data flow diagram:

i. Data flow arrows should never cross each other.
ii. All elements such as processes (circles), external entities (squares), and data stores (files) must be labeled. iii. When breaking down data flows into detailed levels, ensure that the overall input and output remain consistent—this is known as balancing.
iii. No two data flows, external entities, or processes should share the same name.
iv. All data flows should be positioned along the outer edge of the diagram.
v. Assign clear, descriptive names to data flows, processes, and data stores.
vi. Elements like data units, passwords, and validation procedures are considered outside the scope of a DFD and should be handled separately.

A Data Flow Diagram can serve to visually represent how data is processed or organized within a system. A high-level or fundamental DFD can be further broken down into more detailed, lower-level diagrams that illustrate finer steps and more specific aspects of the system's operation.

In a DFD, data travels from an external source or internal data store to another internal data store or external destination, passing through a process that manipulates or transforms the data. The design process typically begins with a context-level DFD, which portrays the interactions between the system and external elements—these external elements act as sources and sinks of data.

DFDs help software engineers develop models for both the information domain and the functional domain simultaneously. As more detailed levels of the DFD are created, the system's internal functionality is gradually broken down and clarified.

The Data Flow Diagram has 4 components:

- **Process:** Represents the transformation of incoming data into outgoing data. Symbols used can vary (e.g., circles, rounded rectangles, or ovals). Each process is labeled with a short phrase or word summarizing its function.
- **Data flow:** Illustrates how data moves between different parts of the system. It is symbolized using arrows. Each data flow should be clearly named to specify the kind of data being transferred. This flow can also represent physical materials in addition to information.
- **Warehouse:** Represents the storage of data for future use. Depicted by two parallel horizontal lines, the data store can refer to various physical or digital storage formats, such as files, folders, optical media, or filing cabinets.

- **Terminator:** These are sources or recipients of data located outside the system. Although previously described similarly to warehouses, terminators serve a distinct role—they are represented as squares and define how the system interacts with external elements.
- **Context Diagram:** Also known as a Level 0 DFD, this diagram focuses entirely on data exchange between the system and external entities. The internal structure is not shown. The whole system is summarized as a single process, with arrows showing input from and output to the external world.

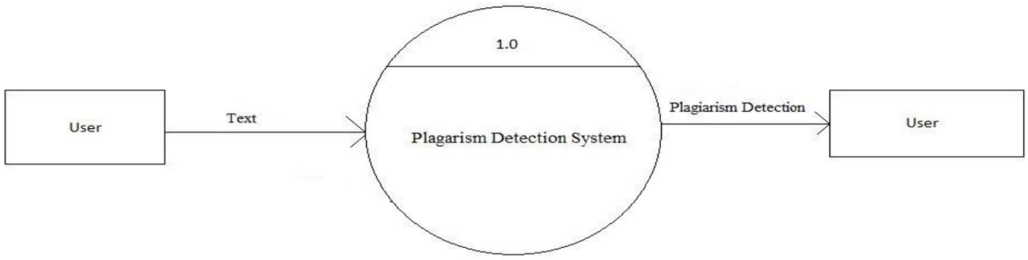


Figure 3.4: Context diagram of Plagiarism Detection System

3.2.5 Activity Diagram

Activity diagrams are graphical models of step-by-step activities and actions with support for concurrency, iteration and choice. In Unified Modeling Language, activity diagrams can be employed to define the business and operational step-by-step processes of elements in a system. An activity diagram illustrates the entire flow of control. Activity diagrams are built from a restricted vocabulary of shapes, linked by arrows. The most significant shape types:

- rounded rectangles denote activities;
- diamonds denote decisions;
- bars indicate the beginning (split) or termination (join) of parallel activities;
- a black circle signifies the beginning (initial state) of the workflow;
- an encircled black circle represents the end (final state).

Arrows travel from beginning to end and illustrate the sequence in which activities occur. But the split and join symbols on activity diagrams only work this out for trivial cases; the semantics of the model are not unambiguous when they are combined with the decisions or loops arbitrarily.

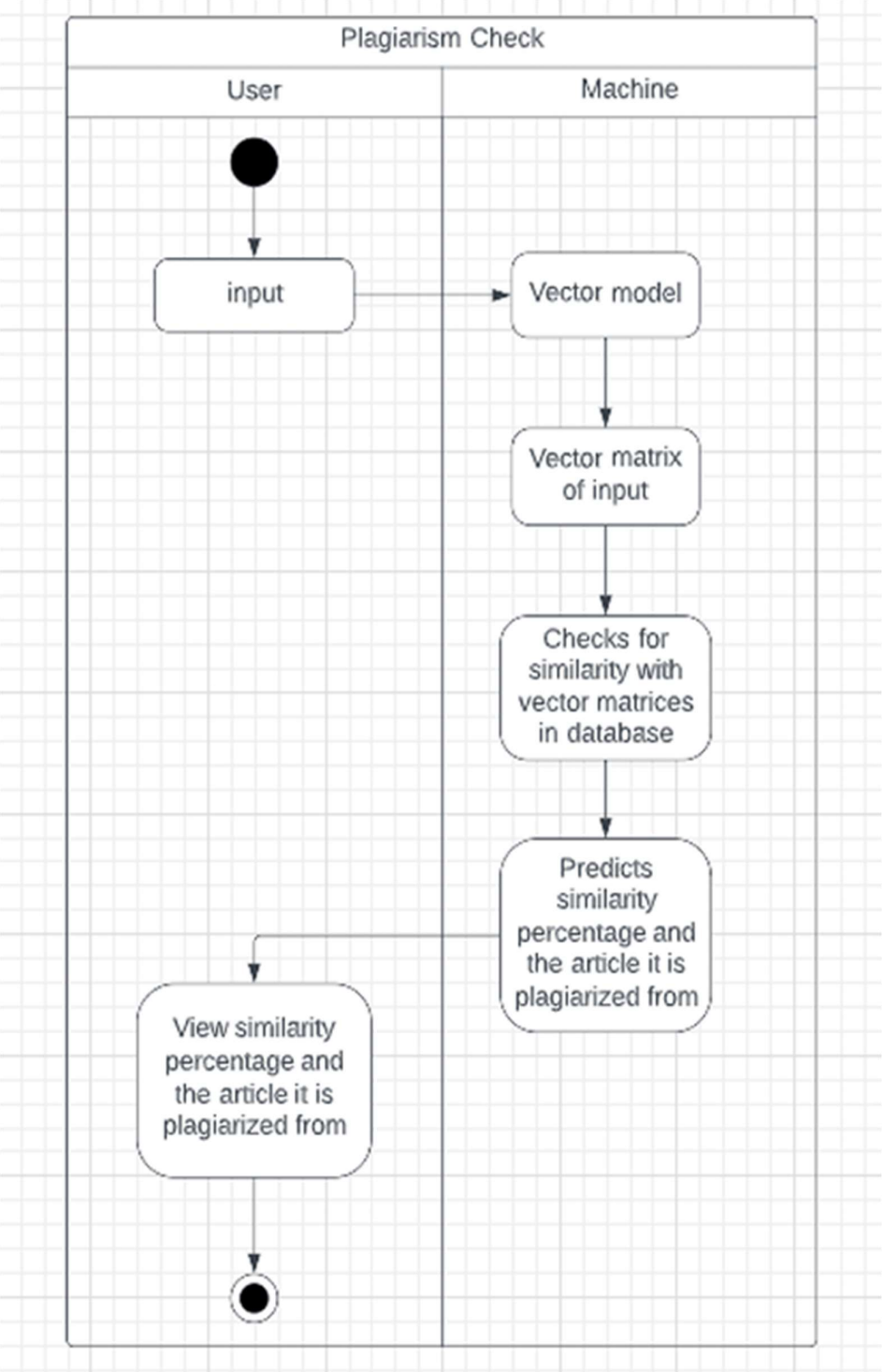


Figure 3.5: Activity Diagram

The website begins with text input from the user, which goes into the model which converts the input into vectors. Based on this vector matrix, the model will query the database for similarity. Once the querying is done, the result is returned to the user for view.

3.2.6 Sequence Diagram

A Sequence Diagram is yet another Behavioral Diagram whose main role is to model how the various objects within the system communicate with each other. A sequence diagram and an activity diagram in most instances are largely similar to others.

Sequence diagrams, also known as event diagrams, event scenarios, or timing diagrams, visually depict interactions between various processes or objects over time. In these diagrams, vertical lines (representing lifelines) are drawn parallel to one another to show entities existing concurrently. Horizontal arrows indicate messages exchanged between these entities, following their sequence of occurrence. This approach provides a clear graphical representation of simple runtime scenarios. When the lifeline represents an object, it is shown as fulfilling a particular role. Not specifying an instance name can indicate anonymous or unnamed instances. The diagram uses messages to depict interactions, with horizontal arrows labeled with the message name. Synchronous calls are represented by dashed arrows with filled heads, return messages by dashed arrows with stick heads, and asynchronous calls by solid arrows with stick heads.

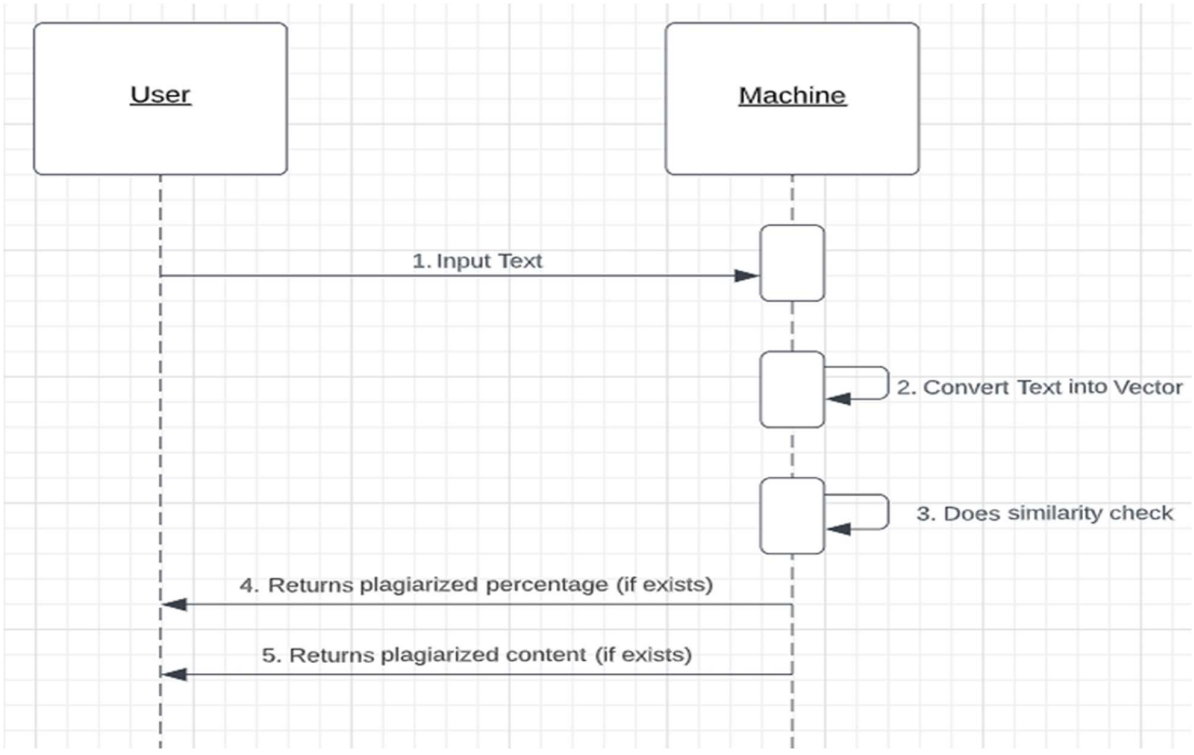


Figure 3.6: Sequence Diagram

Activation boxes, also known as call-method boxes, are rectangular, vertical boxes drawn on top of lifelines in sequence diagrams. They indicate periods when an object is actively performing a

procedure in response to receiving a message. In UML, these are referred to as Execution Specifications.

When an object sends a message to itself, it places a new activation box above the existing one, illustrating a nested or recursive level of execution. If an object is destroyed (i.e., removed from memory), an X is marked at the top of its lifeline, and the lifeline ends—meaning no further interactions are recorded below that point. This termination must occur as the result of a message, either sent from the object itself or from another object. (Note: In some examples, this termination convention may not be shown.)

A message received from outside the diagram—originating externally—can be represented in UML as a "found message", which uses a filled-in circle, or it can appear as entering from the border of the sequence diagram, referred to as a gate.

In the illustrated model, two main entities are involved: the User and the Machine. The interaction begins when the user provides text input on the website. This input is processed by the system, which converts the text into a vector matrix. Based on the computed vectors, the model performs a similarity search within the database. Once the relevant data is retrieved, the system returns the result, which is then displayed to the user.

3.2.7 Deployment Diagram

Deployment diagrams are used to illustrate the physical arrangement of a system's hardware and how software components are distributed across it. These diagrams provide a static view of how the system is physically deployed, showing the nodes and the connections between them.

Objective:

As the name suggests, deployment diagrams are focused on illustrating the physical deployment of software on hardware devices. While component diagrams define the individual software units, deployment diagrams show how these components are installed on the system's hardware.

Although UML (Unified Modeling Language) mainly emphasizes the design of software components, deployment diagrams are among the few that incorporate both hardware and software aspects. While most UML diagrams focus on logical structures, deployment diagrams specifically highlight the system's physical infrastructure. These diagrams are especially useful to system engineers.

The primary goals of deployment diagrams include:

- Visualizing the physical architecture of the system.
- Detailing the hardware used for deploying software components.
- Illustrating the runtime environment and processing nodes.

Creating a Deployment Diagram:

Deployment diagrams represent the deployment aspect of a system and are closely related to component diagrams. Components shown in component diagrams are physically deployed using deployment diagrams. The core elements in these diagrams are *nodes*, which represent physical devices used in the deployment of software applications.

Deployment diagrams are valuable tools for system engineers, as they help address several critical factors such as:

- **System performance**
- **Scalability**
- **Ease of maintenance**
- **Portability**

So before drawing a deployment diagram the following artifacts should be identified:

- Nodes
- Relationships among nodes

1. **Components**:
    a. **Web Server**: Represents the server hosting the web application.
    b. **Database Server**: Stores historical house price data and model parameters.
    c. **Machine Learning Algorithm**: The plagiarism detection system (e.g. consine similarity) deployed on the server.
    d. **User's Browser**: Represents the client-side interface where users interact with the system.

2. **Connections**:
    a. Users access the system through their **Browsers**.

b. The **Web Server** communicates with the **Database Server** to retrieve historical data.
c. The **Web Server** interacts with the **Machine Learning Model** to make predictions.

3. **Deployment Nodes**:
    a. **Web Server Node**: Represents the physical or virtual server hosting the web application.
    b. **Database Server Node**: Represents the server where the database is hosted.

4. **Explanation**:
    a. The **Web Application** interacts with the **House Price Data** stored in the **Database** to retrieve historical information.
    b. The **Machine Learning Algorithm** resides within the **Web Application** and makes predictions based on input data.
    c. Users (represented by the **User** actor) access the system via their browsers.

So the following deployment diagram has been drawn considering all the points mentioned above:

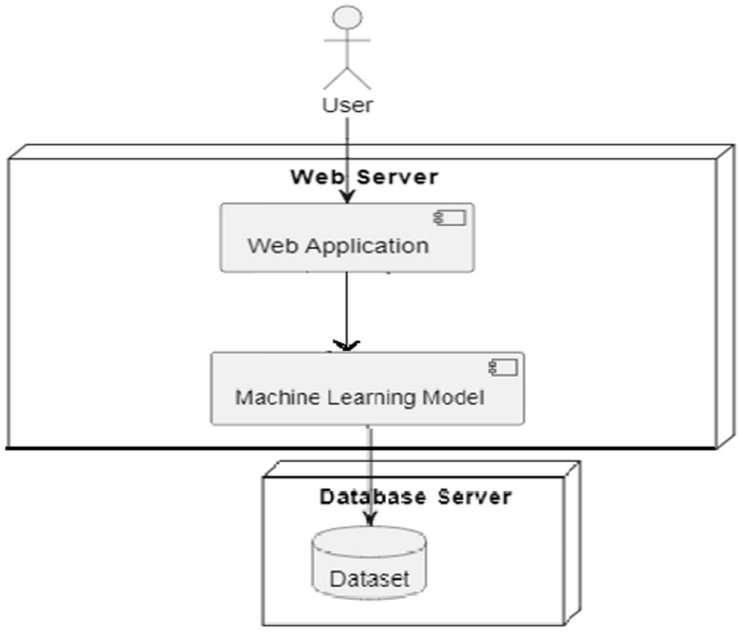


Figure 3.7: Deployment diagram

3.3 User Interface Design

The user interface (UI) of a plagiarism detection web application is an essential element that connects advanced machine learning algorithms to end-users. It ought to be intuitive, interactive, and informative, enabling users to easily input pertinent data and obtain their plagiarism valuation in a precise and comprehensible format. The UI ought to guide users visually through every step of the detection process, from data input to result presentation, for a smooth experience.

Key points to consider for such a UI design include:

- **Simplicity:** A clean and uncluttered interface that focuses on user tasks.
- **Consistency:** Uniform elements and layout throughout the application to avoid confusion.
- **Intuitiveness:** An interface that is easy to understand and navigate, even for first-time users.
- **Feedback:** Immediate and clear feedback on user actions, such as successful data submission.
- **Accessibility:** Design that is accessible to all users, including those with disabilities.
- **Visual Elements:** Use of charts and graphs to visually represent data and detection.
- **Mobile Responsiveness:** Ensuring the application is fully functional on various devices and screen sizes.

With time and ingenuity, innovations have been a part of technology. With ingenuity there is progress and with progress there is complexity but we still want things to be simple but advanced. This is when we have to think about how to make things simple. We require simplicity in product development, and we require simplicity in using the product. We need something to satisfy this requirement. User interface design just fills the bill. It makes it easier for developer/designer to design end product in step-by-step manner.

User Interface (UI) design has four main elements:

- **Usability:** Usability is primary component of interface design. Formulating simple questions will make it possible to make application usable. The questions can be like can user navigate from one page to other, can user recall last visited page and so on. It is

associated with navigation and process of visual and functional features. It is the attribute for quality measure which assesses ease of use of user interface of an application or a web application

- **Visualization:** There are many numbers of websites and web applications that are being used nowadays. A person himself employs a lot and various of them in a day. Creating a workable application is not sufficient if one desires its application's usability to grow there where visualization need arises to function. Visualization is utilized to render data and application/system/site content presentable and understandable; Visualization is not merely turning an application or she fancy into existence but turning its content readable and data easily understandable. Visualization assists in the communication of statistical data, users, and designers. It should be utilized whenever possible and necessary. It possesses extremely high importance but that doing is fine. The reason for using visualization is so that the content remains easy to comprehend and nest and presentable and simple. Simple is the secret to good user interface design.

- **Functionality:** A system is made up of visual and functional components. Visual components enhance look and feel of the component while functional component brings the system into operation. Visual component can be said as a body and functional component a brain. Every part of the body is capable of doing its job because brain is providing instructions about what to do and how to do Functionality brings life into a system. When we create layout, it doesn't do anything by itself until it is combined with programming logic. When are creating an interface. not just creating visual stuff, but we are creating logic behind it too. Adding functionality is like informing an object what job it has to perform. It is attribute of a system's visible things.

- **Accessibility:** Designs are not created for the largest user group or high-tech resource users alone. Not everyone can use the system due to various reasons. While designing, all types of users have to be thought about and catered to as much as possible. All those who lack as much computer knowledge as some other people should also be taken into account. The slow connection users must also be kept in mind while designing. The disabled too should be kept in mind. This has to be done so that an alternative is made available to them in order to facilitate the particular users to appreciate the content as much as others.

3.4 Methodology

The method that we suggest analyzes text to obtain a similarity score through machine learning and NLP. Through tokenization, stemming, and removal of stop words, the model preprocesses text in a simple way. The model then transforms text into a vector representation through the term frequency-inverse document frequency (TF-IDF) method. In the model, cosine similarity is employed to match the vector representations of the input text against a database of known sources. The cosine similarity score, which is the measure of similarity of two vectors, ranges between 0 and 1, where 0 represents no similarity and 1 represents perfect similarity.

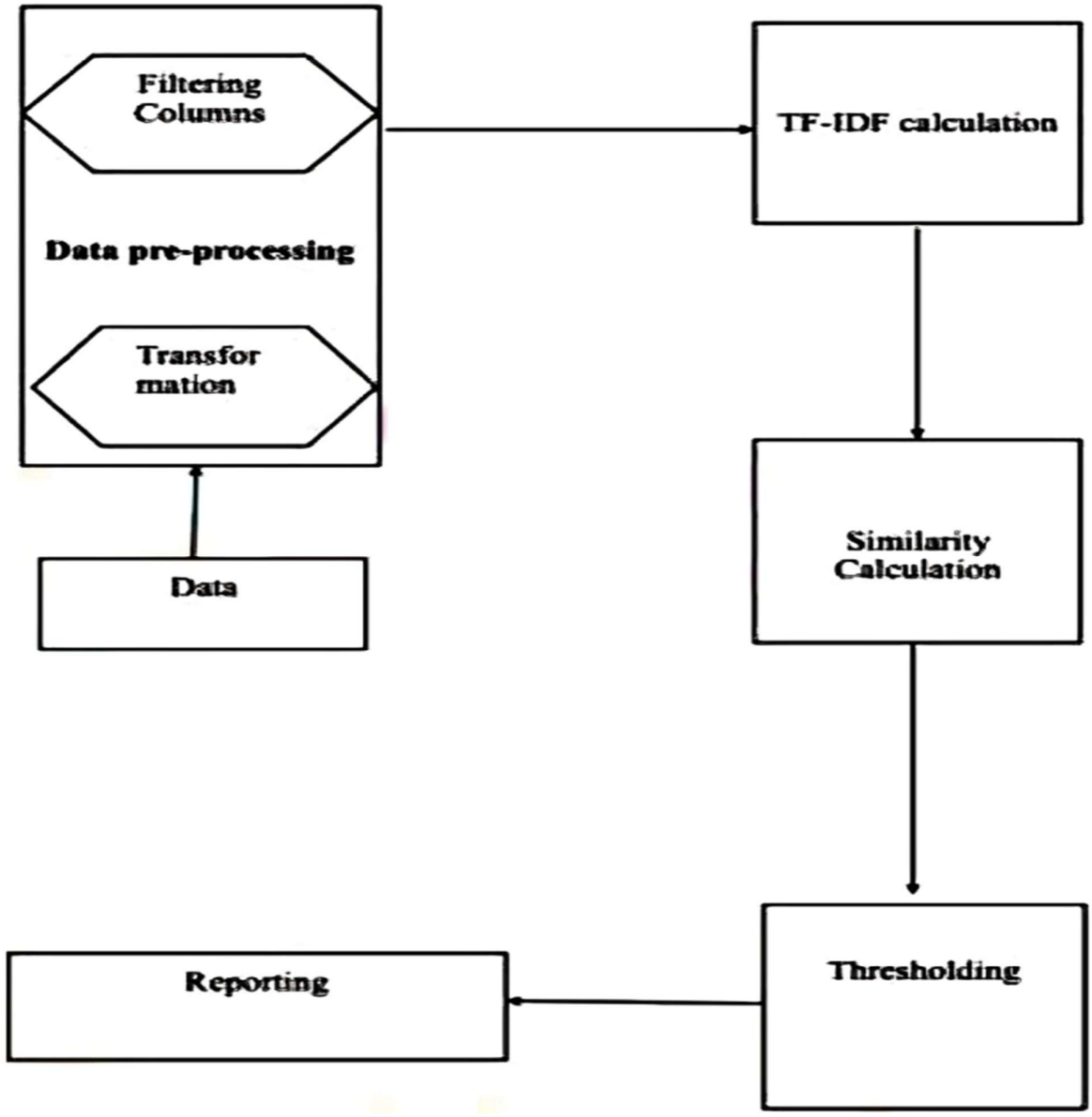


Figure 3.8: Methodology

1. **Flowchart:** The architecture of the proposed plagiarism detection system is divided into multiple stages. The initial stage, known as pre-processing, focuses on cleaning the input text by eliminating stop words, punctuation, and irrelevant symbols. After transforming the text to lowercase, it is tokenized. Each token is then analyzed using the Term Frequency-Inverse Document Frequency (TF-IDF) technique, which assigns importance to a term based on its frequency in the document and its rarity across a collection of documents. This value is computed by multiplying the term frequency (TF) with its inverse document frequency (IDF). Following this, the system performs a similarity analysis. Both cosine similarity and Jaccard similarity are calculated to compare the input document against a reference database. Jaccard similarity evaluates the overlap between two sets of tokens, while cosine similarity compares the TF-IDF vectors to measure textual closeness. Before presenting the results, a similarity threshold is applied.

2. **Data Collection & Pre-processing:** This stage involves gathering both original and plagiarized documents from a variety of sources such as academic articles, essays, and research publications. Pre-processing is critical to filter out irrelevant data and background noise. Techniques like lemmatization, stemming, and stop word removal are employed. Additionally, normalization ensures uniformity by converting text to lowercase and removing punctuation and special symbols.

3. **Feature Extraction:** After pre-processing, the next step is to derive features from the cleaned data. In this system, feature extraction is accomplished using TF-IDF and Jaccard similarity methods. The TF-IDF technique assigns numerical values to terms based on their frequency and distribution, helping highlight the relevance of each word. On the other hand, Jaccard similarity assesses the proportion of shared words between two documents.

4. **Similarity Calculation:** Once features are extracted, the similarity between the input document and existing texts in the dataset is computed. Cosine similarity evaluates the alignment between two TF-IDF vectors using the ratio of their dot product to the product of their magnitudes. Jaccard similarity, by contrast, is determined by the size of the intersection divided by the size of the union of the word sets.

5. **Thresholding:** After computing similarity values, a predefined threshold is used to interpret the results. Cosine similarity helps gauge the angular difference between two text vectors, while Jaccard similarity measures the commonality between sets of words. Documents exceeding the set similarity threshold are flagged for potential plagiarism.

Chapter 4

Implementation and Testing

4.1 Introduction to Languages, IDE's, Tools and Technologies

4.1.1. Python Language

- Python is a high level, dynamic, interpreted and general-purpose programming language.
- It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.
- Python is both easy to learn and highly powerful, making it a versatile language for scripting and application development.
- Owing to the syntax and dynamic typing, not to mention being an interpreted programming language, Python is ideal for scripting and instant application development.
- Python features several programming methodologies, such as object-oriented, imperative, as well as functional programming paradigms.
- Though Python isn't developed to address the specialty domain spaces such as web programming, yet it has generalizability for it and is employed across many other applications.
- Since Python provides dynamic typing, one won't be required to state specifically which type data for the variable, such as using just `a = 10` when an integer data would be given for `a`.
- Python development and debugging are quick because there is no compilation step. This enables rapid edit-test-debug, which makes it extremely efficient for development.

4.1.1.1. Python features:

- **Beginner-Friendly**: Python is user-friendly and ideal for developers, thanks to its high-level syntax.
- **Readable and Expressive:** The readable and concise syntax of Python makes it easily understandable.
- **Interpreted:** Python runs code line by line, making it simple to debug, which is highly beneficial for newcomers.

- **Object-Oriented Language:** Python supports object-oriented programming with classes and objects.
- **Cross-platform Language:** Python is portable as it runs on a variety of platforms such as Windows, Linux, and macOS.
- **Open Source and Free Technology:** Python is free to use, and its source code is available for modification.
- **Wide Built-in Library:** Python has an extensive library covering many modules and functions for quick development.
- **GUI Programming Support:** Python offers support for development of graphical user interfaces for software.
- **Compatibility with Other Languages:** Python is smoothly integrated with languages like C, C++, and Java.
- **Extensible:** Python provides extension interface to use other languages such as C and C++ to build on top.

4.1.1.2. Python applications:

Python is also renowned for being highly versatile and can be used in a multitude of software development areas. Following are some application domains where Python is widely employed:

- **Web Applications:** Python is ideal for building web applications, offering libraries for handling internet protocols such as HTML, XML, JSON, and email. Popular frameworks like Django, Pyramid, and Flask help structure and develop web-based applications. Notable projects include PythonWikiEngines, Pocoo, and PythonBlogSoftware.
- **Desktop GUI Applications:** The Tk GUI library is available in Python for developing desktop application graphical user interfaces. Cross-platform toolkits such as wxWidgets, Kivy, and PyQt are also present, with Kivy gaining special popularity for multi-touch applications.
- **Software Development:** Python supports the software development process, including build management, testing, and automation tasks.
- **Scientific and Numeric:** Python is widely used for scientific and numeric computing, with libraries including SciPy, Pandas, and IPython. SciPy itself is a collection of packages aimed at engineering, science, and mathematics.

- **Business Applications:** Python is used to create business applications, such as ERP and e-commerce systems. Tryton, for instance, is a high-level platform for business applications.
- **Console Based Application:** Python may be utilized in creating console-type applications. An example of which is the well-known interactive shell IPython for Python.
- **Audio or Video based Applications:** Python works well when used to create audio and video applications. Some examples include TimPlayer and cplay.
- **3D CAD Applications:** Python may be utilized in designing CAD (Computer-Aided Design) applications, an example being a fully-featured CAD application like Fandango.
- **Enterprise Applications:** Python is utilized in developing enterprise-level applications, like OpenERP, Tryton, and Picalo, that organizations utilize for a number of different internal purposes.
- **Applications for Images:** Python enables the development of applications for image manipulation. Some examples include VPython, Gogh, and imgSeek

4.1.1.3. Python Libraries

A Python library is a collection of functions and methods which allows us to perform a lot of actions without writing the code. Some commonly used Python libraries are: -

4.1.1.3.1. Matplotlib:

- Matplotlib is a 2D plotting library for Python that enables the creation of high-quality figures suitable for publication. It supports a variety of output formats and works across different platforms and interactive environments.
- Matplotlib can be utilized in Python scripts, within the Python and IPython shells, Jupyter notebooks, web application servers, and with four different graphical user interface toolkits.
- Matplotlib aims to simplify basic tasks while making complex ones achievable.
- You can easily generate a wide range of visualizations, including plots, histograms, power spectra, bar charts, error charts, and scatter plots, all with minimal code.
- The pyplot module offers a MATLAB-like interface for simple plotting, especially when used with IPython.

- For advanced users, Matplotlib provides extensive control over elements like line styles, font settings, and axis properties through an object-oriented interface or a function set similar to MATLAB's.

4.1.1.3.2. Requests:

- Requests is a Python HTTP library, released under the Apache2 License.
- The goal of the project is to make HTTP requests simpler and more human-friendly.
- The latest version is **2.32.3**.
- The requests library has become the standard for making HTTP requests in Python.
- It abstracts the complexities of making requests behind a beautiful, simple API so that you can focus on interacting with services and consuming data in your application.

4.1.1.3.3. Scikit-learn:

- **Scikit-learn** (formerly known as **scikits.learn** and commonly referred to as **sklearn**) is an open-source machine learning library for Python.
- It includes a wide range of algorithms for classification, regression, and clustering, such as support vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to work seamlessly with Python's numerical and scientific libraries like NumPy and SciPy.
- Scikit-learn is primarily written in Python, utilizing NumPy for efficient linear algebra and array operations.
- Some key algorithms are written in Cython to enhance performance.
- Support vector machines are implemented via a Cython wrapper around **LIBSVM**, and logistic regression and linear support vector machines use a similar wrapper around **LIBLINEAR**.
- In some cases, extending these methods with Python might not be feasible.
- Scikit-learn integrates well with other Python libraries, such as **matplotlib** and **plotly** for visualization, **NumPy** for vectorized array operations, **pandas** for dataframes, and **SciPy**, among others.

- Scikit-learn is one of the most popular machine learning libraries on GitHub.

4.1.1.3.4. SciPy:

- SciPy is an open-source, free Python library designed for scientific and technical computing.
- It includes modules for various tasks such as optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, and other common scientific and engineering operations.
- SciPy is built on the NumPy array object and is a core part of the NumPy stack, which also includes libraries like Matplotlib, pandas, SymPy, and other scientific computing tools.
- The NumPy stack is widely used, comparable to software like MATLAB, GNU Octave, and Scilab.
- This collection of tools is sometimes referred to as the SciPy stack.
- SciPy is also associated with several conferences, including SciPy (US), EuroSciPy (Europe), and SciPy.in (India), which bring together users and developers of these tools.
- Enthought initiated the SciPy conference in the US and continues to sponsor these events and manage the official SciPy website.
- The library is distributed under the BSD license and is developed with contributions from an open community of developers.
- SciPy is also supported by NumFOCUS, an organization dedicated to promoting reproducible and open science.
- The core data structure in SciPy is the multidimensional array, which is provided by the NumPy module, ensuring seamless integration with various databases.
- While NumPy provides basic functions for linear algebra, Fourier transforms, and random number generation, SciPy offers more advanced and general-purpose versions of these functions.
- NumPy also serves as an efficient container for multidimensional data with arbitrary data types.
- Earlier versions of SciPy used Numeric as the array type, which has now been replaced by the more modern NumPy array structure.

4.1.2 Jupyter Notebook

- The Jupyter Notebook is an interactive, web-based environment that allows you to write and execute human-readable documents while describing data analysis processes.
- Jupyter Notebook is an open-source web application that lets you create and share documents containing live code, equations, visualizations, and text. It is maintained by the Project Jupyter team.
- Originally part of the IPython project, Jupyter Notebooks evolved from the IPython Notebook. The name "Jupyter" reflects the core languages it supports: **Julia**, **Python**, and **R**. Jupyter includes the IPython kernel for Python programming but currently supports over 100 other kernels.
- Jupyter Notebook is not bundled with Python, so you'll need to install it separately to get started.
- Python has many distributions, but this guide focuses on two popular options for installing Jupyter Notebook. The most widely used is **CPython**, the reference implementation of Python available on the official website. It's assumed that Python is already installed.

4.1.3 PyCharm

PyCharm is the most widely used Integrated Development Environment (IDE) for Python, offering robust features like advanced code completion, inspection, debugging, and support for web programming frameworks. It is developed by **JetBrains**, a Czech company known for creating IDEs for various web development languages like **JavaScript** and **PHP**. PyCharm delivers a comprehensive set of features for users and developers, including:

- Code Completion and Inspection
- Advanced Debugging
- Support for Web Frameworks such as Django and Flask

Key Features of PyCharm:

- **Code Completion:** PyCharm enables smoother code completion whether it is for built in or for an external package.

- **SQLAlchemy as Debugger**: You can set breakpoints, pause the debugger, and view the SQL representation of expressions for SQL code.
- **Git Visualization in the Editor**: PyCharm offers an intuitive way to visualize commits. You can easily see the difference between the latest commit and the current state of your code via blue highlights.
- **Code Coverage in the Editor**: PyCharm allows you to run .py files outside the editor and still view code coverage details within the project tree and summary sections.
- **Package Management**: PyCharm visually displays all installed packages, with an option to search for and add new ones.
- **Local History**: This feature tracks changes over time and works similarly to Git. It gives detailed information on what can be rolled back or added.
- **Refactoring**: PyCharm makes refactoring easy by offering shortcuts for renaming files and making other adjustments smoothly.
- **WAMP Server**: WAMP is a software package for Windows systems that includes Apache (a web server), MySQL (an open-source database), and PHP (a scripting language for dynamic web pages). The package may also include tools like **phpMyAdmin** for database management or other scripting languages like **Python** or **Perl**.

4.1.4. Flask

- Flask is a Python-based micro web framework designed for building web applications. It was created by Armin Ronacher. Compared to Django, Flask is more straightforward and beginner-friendly due to its minimalistic design and smaller amount of boilerplate code needed to develop basic web applications.
- A Web Application Framework, or simply a Web Framework, is a collection of tools, modules, and libraries that simplifies the process of developing web applications by handling common tasks like protocol handling, threading, and more. Flask operates on the WSGI (Web Server Gateway Interface) protocol and uses the Jinja2 templating engine to render dynamic content.

METHOD DESCRIPTION

- **POST** – Sends form data to the server. The data sent is not stored in the browser cache.

- **PUT** – Updates or replaces the current state of the resource identified by the given URL.
- **GET** - Sends data to the server without encrypting it; typically used to request data from a specified resource.
- **DELETE** – Removes the specified resource associated with the given URL.
- **HEAD** – Similar to GET, but it retrieves only the headers and not the body of the response.

4.1.5. HTML

Hypertext Markup Language (HTML) is an essential language for developing websites. Our project is built using HTML, which includes key codes commonly used in web development.

HTML also supports dynamic HTML (DHTML) and scripting languages like VBScript and JavaScript. For this project, we have utilized JavaScript.

HTML is a straightforward and user-friendly language, making it easy to learn. Its simplicity allows developers to use any text editor for coding, making web page creation a hassle-free process. HTML is the language that browsers interpret, and web pages are essentially HTML documents.

HTML consists of special codes, known as tags, that are embedded in text to provide formatting and linking information. These tags define the structure and content of web pages, making HTML an essential part of web development.

4.1.5.1. HTML Tags

Paired Tags: Tags are instructions that are emended directly into the text of Pair tags called closed tags because it begins <>and close</>.

Singular Tags: A singular tags not have a companion tag e.g.

Some tags that we used in our project describe in brief given below: -

- <HTML>it is used to start.
- <HEAD> it is used to place the information about the program.
- <TITLE>it is used to give the title of the information.
-
it is used to break a line
- <H1> to <H6>it is used to give the size of the specific heading

4.1.6 CSS

CSS (Cascading Style Sheets) defines the visual appearance and layout of HTML elements.

- In this plagiarism detection frontend:
    - CSS is used to style input fields, buttons, containers, headers, and result sections.
    - It ensures a clean, consistent, and responsive user interface.
- Key benefits of CSS:
    - Allows centralized styling for multiple web pages using external stylesheets.
    - Saves development time by separating style from content.
    - Enhances accessibility and user experience through layout control, color schemes, and font settings.
- CSS makes the frontend visually appealing and easy to navigate.

4.1.7 JavaScript

JavaScript (often abbreviated as **JS**) is a programming language that follows the **ECMAScript** specification. It is a high-level, multi-paradigm language, typically compiled just-in-time. JavaScript features curly-bracket syntax, dynamic typing, prototype-based object orientation, and first-class functions.

Alongside **HTML** and **CSS**, JavaScript is one of the core technologies that power the **World Wide Web**. JavaScript code can be embedded directly into HTML pages to enhance them with dynamic features.

With JavaScript, you can add a wide range of interactive elements to your web pages, making it an essential tool for creating web applications. It is widely used for client-side behavior in web development, and all major web browsers come with a built-in JavaScript engine to execute code.

JavaScript allows developers to easily respond to user-triggered events, enabling the creation of interactive experiences. Many effects now achievable with JavaScript were once only possible through server-side technologies like **CGI**. As a result, JavaScript allows for the creation of highly sophisticated web pages on the internet.

4.1.8 ReactJS

ReactJS is an open-source JavaScript library used to build user interfaces from reusable components.

- Maintained by Meta (formerly Facebook), React is ideal for developing fast, responsive web apps like plagiarism detection systems.
- In the context of this frontend:
    - React is used to design an interactive interface for users to upload documents or text for plagiarism checking.
    - The system can support user authentication, allowing registered users to log in and view their analysis history.
    - Unregistered users can create an account and log in before using the system.
    - After login, users can input or upload content, which will be sent to the backend for processing.
    - Users can log out after completing the check, redirecting them back to the login page.
- Additional libraries may be used for routing (e.g., React Router) and managing state (e.g., Redux or Context API).

4.1.9 Firebase

- Firebase, a Google platform, simplifies backend services for frontend apps like this one.
- It provides:
    - **Authentication:** To manage user login, registration, and secure access.
    - **Firestore/Realtime Database:** To store and retrieve user data, past plagiarism checks, or uploaded files.
    - **Hosting (optional):** To deploy the frontend application online.
- Firebase helps create a seamless and secure experience for users checking documents for plagiarism.

4.2 Algorithm/Pseudocode used

Function GoogleSearch(query, api_key, cse_id):

    Connect to Google Custom Search using the API key and Search Engine ID

    Search for the query

    Return the top results (e.g., 2 links)

Function ExtractTextFromURL(url):

    Try to:

        Send a GET request to the webpage

        Parse the HTML and remove script/style content

        Extract readable text from the page

        Clean the text (remove extra spaces)

        Return the clean text

    If there's an error:

        Print error message

        Return None

Function CalculateCosineSimilarity(text1, text2):

    Use TF-IDF to convert both texts into numerical vectors

    Calculate cosine similarity between the two vectors

    Return similarity score (between 0 and 1)

Function DetectPlagiarism(content, api_key, cse_id):

    Initialize list to store plagiarized sources

    Set total_similarity_score = 0

```
Set total_sentences = 0

Split content into sentences (using '. ')

For each sentence in the content:
    If sentence has more than 15 characters:
        Print part of the sentence being searched
        Call GoogleSearch(sentence, api_key, cse_id)

        For each result from search:
            Get the URL
            Call ExtractTextFromURL(url) to get page text

            If text is extracted:
                similarity = CalculateCosineSimilarity(sentence, page_text)
                Add similarity to total_similarity_score
                Increase total_sentences by 1

                If similarity > 0.5:
                    Save sentence, URL, title, and similarity in plagiarized_sources list

If total_sentences > 0:
    average_score = (total_similarity_score / total_sentences) * 100
Else:
    average_score = 0
```

Return plagiarized_sources and average_score

4.3 Testing Techniques

Software testing is the activity that is applied to quantify the quality of produced software. Test plan is documentation that outlines the plan that is to be followed in order to confirm and certify that a system or product has its design specification and other conditions fulfilled. For one to provide an entirely functioning, a high acceptance or error-free system, the software is tested thoroughly prior to introducing into the market. In general, the purpose of software testing is to verify that all the functionality of the module in the system will function and the system performance will get the best result that satisfied the user expectation and also referred to as the process of verification and validation of the system requirements.

That is, software testing is a process of verification and validation. Verification is the activity to ensure the product meets the requirements stipulated during the beginning of the development life cycle. In other words, to ensure the product works in the desired fashion. Validation is the activity to ensure the product meets the requirement as stated when we complete the development life cycle. In other words, ensuring the product gets built according to customer requirements.

Software testing is an examination done to inform stakeholders about the quality of the software product or service being tested. Software testing can also give an objective, independent perspective of the software to enable the business to understand and value the risks of software deployment. Test methods encompass, but are not restricted to, the practice of running a program or application with the goal of discovering software bugs (faults or other defects). A minor mistake can potentially investigate into numerous big issues. The optimal program is worth less if it does not signify user requirements. The quantity in nature of mistakes in a fresh design relies on a number of factors like:

1. Communication between user and designer.
2. The programmer's ability to generate specifications. code that reflects exactly the system
3. The time frame for the design.

Subject to the product and the duty of the organization in question that a test plan would be pertinent, a test plan can cover a single test method or several methods.

4.3.1 The testing strategies used in this project consists of the followings:

- Unit Testing (Module Testing)
- Integration Testing
- System Testing

4.3.1.1 Unit Testing (Module Testing):

The unit is the least testable entity of an application. Under procedural programming, the unit would either be a program, function, or procedure alone. In object-oriented programming, the unit in the least measurement would generally be a method and can belong to a base class, an abstract class, or a derived class.

During the unit testing phase, each module is tested individually to assess its efficiency and reliability. The primary goal of unit testing is to isolate and verify the functionality of each part of the module, ensuring that each component works correctly on its own.Feedback and issues identified during this testing phase are recorded for future improvements, and modifications to the modules are often made based on these insights.

4.3.1.2 Integration Testing:

Integration testing (often referred to as I&T) is a software testing phase in which separate software modules are integrated and tested together. It is done after unit testing and before system testing. In integration testing, modules that are successfully tested by unit testing are combined into units of larger sizes, and integration test plan tests are executed. The deliverable of this stage is a totally integrated system, which is ready to be system tested. Once thoroughly tested individually, I integrated all the separate modules to form a whole system and checked its reliability afterwards. Unit testing outcomes proved very helpful since no faults were faced during integration.

Types of Integration Testing:

- **Top-Integration:** This approach follows an incremental method for building the program structure. Modules are integrated starting from the main program module and moving downward through the control hierarchy. Subordinate modules are added in a depth-first or breadth-first manner. During testing, stubs (stand-ins for modules) are used for modules that have not yet been integrated, and they are replaced as the testing progresses downwards through the structure.

- **Bottom-Up Integration:** In this method, the integration and testing process begins with the lowest-level modules in the program structure. Since modules at the lower levels are integrated first, the necessary processing for higher-level modules is already in place, eliminating the need for stubs. This ensures that all required components are available from the start.

4.3.1.3 System Testing:

System testing is conducted on a fully integrated software or hardware system to evaluate whether it meets the specified requirements. Typically, system testing involves taking all software components that have successfully passed integration testing and integrating them with any relevant hardware systems.

The main goal of system testing is to identify any inconsistencies or issues between the integrated software units (referred to as **assemblages**) or between these assemblages and the hardware components. Unlike other types of testing, system testing is more comprehensive, aiming to detect defects both within the interconnections between assemblages and within the entire system.

4.4 Test Cases

4.4.1. Test Scenario:

Creating Account and authenticate User by verifying login from the database.

Test Pre-Conditions:

1) Enter valid details for creating account.
2) Enter valid Active Account credentials to login.

Test Steps:

1) Enter User Details
2) Enter Username and Password
3) Click Register/ Login button

Sr.no.	Data Input	Excepted Output	Actual Output	Pass/Fail
1.	All fields are empty	Error message: *indicate compulsory fields	Error message: *indicate compulsory fields	Pass
2.	Email	Error message: Invalid Email-address	Error message: Invalid Email-address	Pass
3.	Password and Confirm Password	Error message: Both password does not match	Error message: Both password does not match	Pass
4.	Login	The System give an error and denied from the login	Login should fail with an error 'Invalid Login'	Pass
5.	User	Login should be allowed and User Visitor side	Login successfully and user its User Page	Pass
6.	Validation Test Case	Pre-define format must be required in control	System give error to enter valid input	Pass
		Enter data in a compulsory field with required field validations.	Data must be field in a compulsory field otherwise error message is displayed	Pass

Table 4.1: Test case for Creating Account and Authenticate User

By executing these testing techniques and relevant test cases, the Plagiarism Detection System can be thoroughly evaluated for the accuracy, reliability, and user satisfaction and by conducting thorough testing using these techniques and test cases, we aim to develop a robust and user-friendly plagiarism detection system that includes reliable algorithm selection.

Chapter 5

Results and Discussions

5.1 User Interface Representation

The user interface (UI) of a machine learning model that is used to identify plagiarism is an essential element that fills the gap between the sophisticated algorithms and the end-user. It is the interactive and visual layer through which users can enter text or document files and get a generated report for the given document. An appropriately designed user interface for this should be easy to use, lead the user through the process with unambiguous instructions and feedback. It normally comprises form fields to enter text or upload document files, submit data buttons, and download generated buttons. The purpose of the UI is to provide access to the complex machine learning processes and make them understandable, so users can take advantage of advanced predictive technology without requiring knowledge of data science. By prioritizing usability and simplicity, the UI enables users to use the plagiarism detection system without requiring in-depth technical knowledge. This makes it suitable for numerous users, including researchers, teachers, and creators of content, who can readily determine the authenticity of their content.

The design of the UI is guided by the following principles:

- **Simplicity**: The platform has a streamlined interface, making it easy for users to access and upload their content for plagiarism detection without having to deal with extra features or information.
- **Intuitiveness**: The layout follows a logical flow, with clear instructions and easily identifiable actions. From submission of text or upload of document to viewing results, the path is as streamlined as possible.
- **Responsiveness**: The UI is designed to be functional across a number of devices, providing seamless performance on anything from desktops to mobiles.
- **Accessibility:** It must be usable by people with disabilities, keeping in mind the principles of inclusive design.
- **Performance**: The UI must be optimized for performance with fast loading and immediate response to user input.

These primary considerations are designed to make sure that the UI not only fulfills its functional role but also creates a good user experience, building confidence and repeat usage.

Finally, the integration of a simple, user-friendly, and responsive UI, and sophisticated plagiarism detection algorithms, with secure authentication through Firebase, creates a strong solution for encouraging academic integrity and guaranteeing written work originality.

5.1.1 Brief Description of Various Modules of the System

The plagiarism detection system is composed of several modules that work together to ensure smooth functionality and accurate results.

Below is a high-level overview of the different components:

1. **Text/Document Input Module**: This module allows users to input text directly or upload document files for plagiarism analysis. Users can paste their content into a text box or use a file upload option to provide a document (such as a .docx or .txt) for checking.
2. **Plagiarism Checking and Similarity Calculation Module**: Once the text or document is uploaded, this module processes the input content using the **cosine similarity algorithm**. It compares the submitted content against a database to identify any matching or similar content, calculating a similarity score that reflects the degree of plagiarism.
3. **Plagiarism Report Display**: After the plagiarism check is complete, this module generates a detailed report for the user. The report highlights the sections of the content that match other sources, providing a clear indication of the percentage of similarity. It also offers feedback to help users understand the results and take appropriate action.
4. **User Authentication Module**: This module handles user authentication using **Firebase OAuth**. Users can securely log in with their existing accounts (e.g., Google), allowing them to save their reports, track plagiarism checks, and access the system with ease.
5. **Accessibility and Responsiveness**: The UI of the plagiarism detection system is designed to be accessible on various devices, including desktops, tablets, and smartphones. It automatically adjusts to different screen sizes, ensuring a seamless experience for all users.

Each module is designed to work together to ensure the plagiarism detection process is efficient, user-friendly, and secure, providing an effective solution for users needing to maintain the originality of their written content.

5.2 Snapshots of System

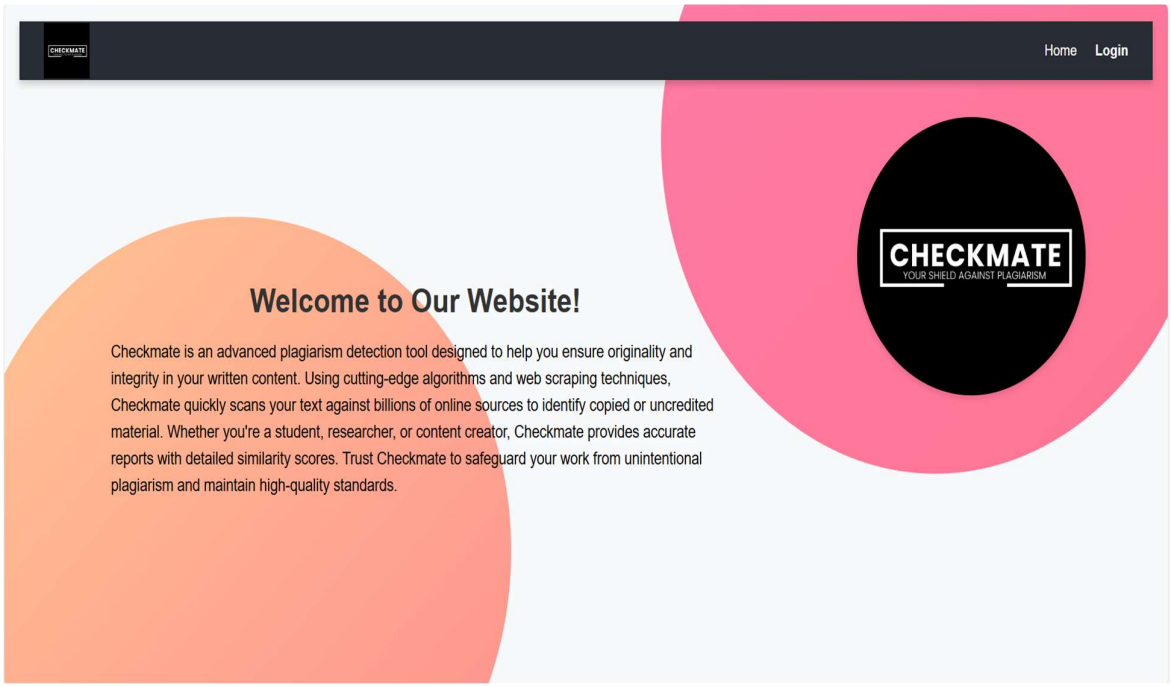


Figure 5.1 Home Page for Plagiarism Detection System

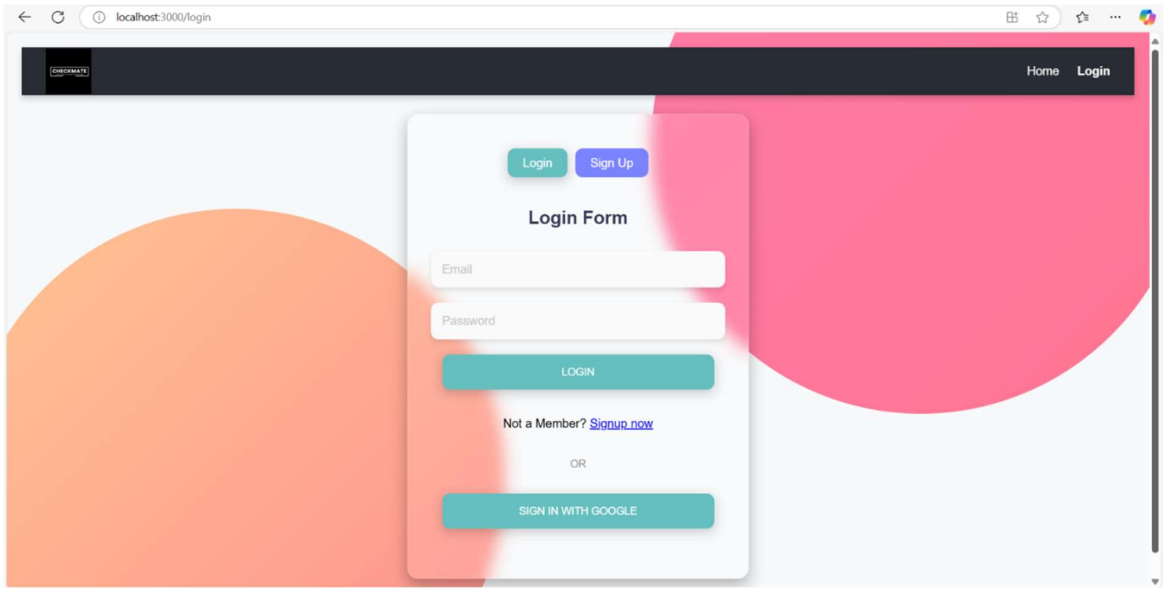


Figure 5.2 Login Page

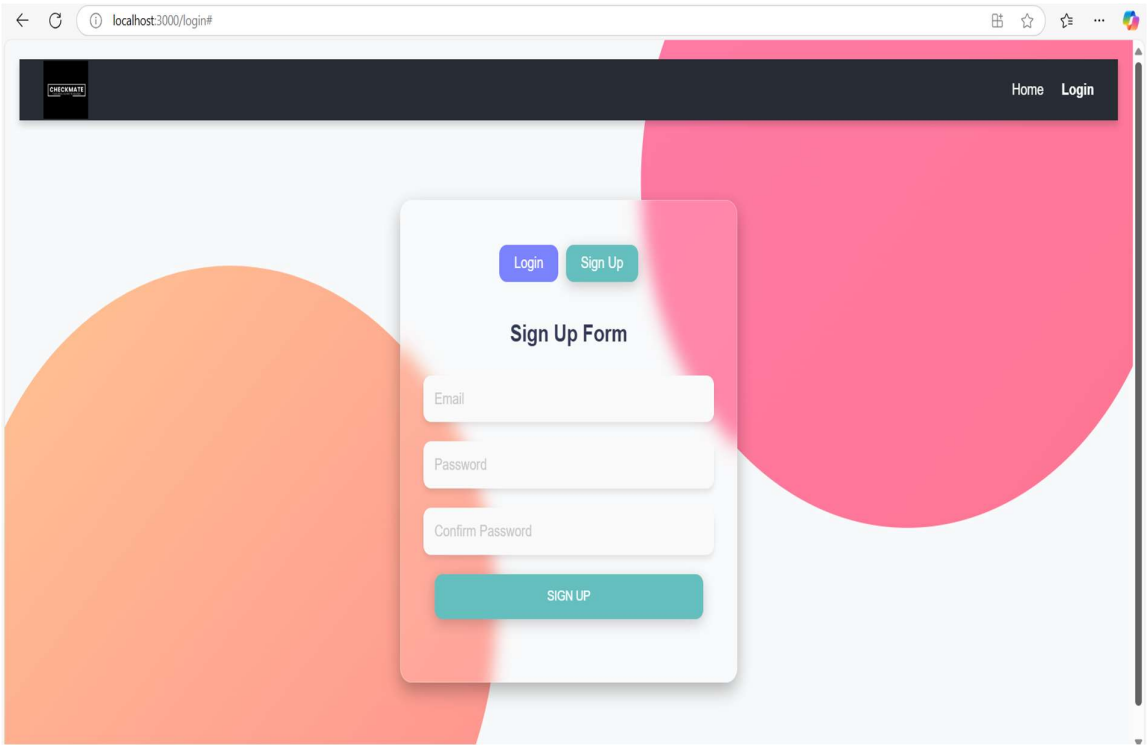


Figure 5.3 Sign Up Page

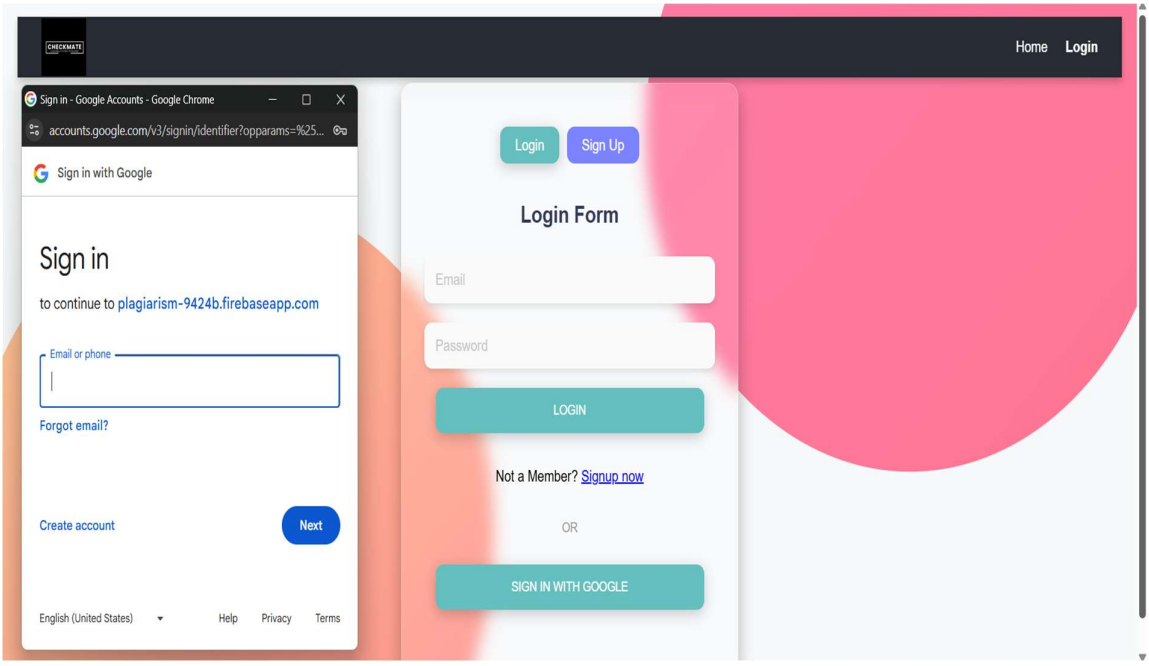


Figure 5.4 Login with Firebase Authentication

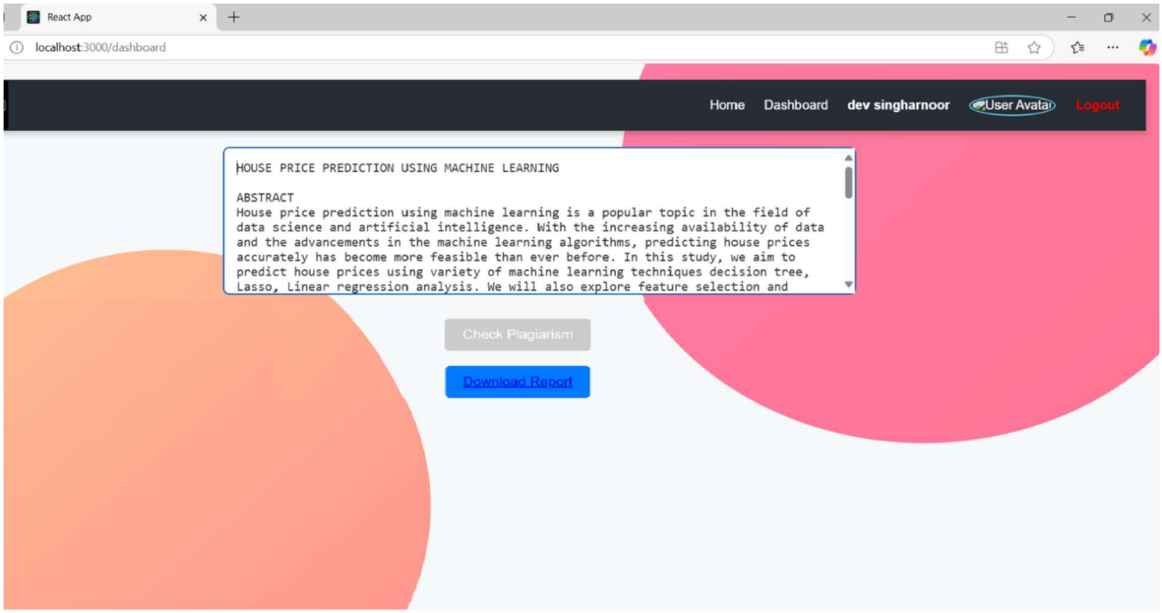


Figure 5.5 Dashboard Page for Plagiarism Detection System

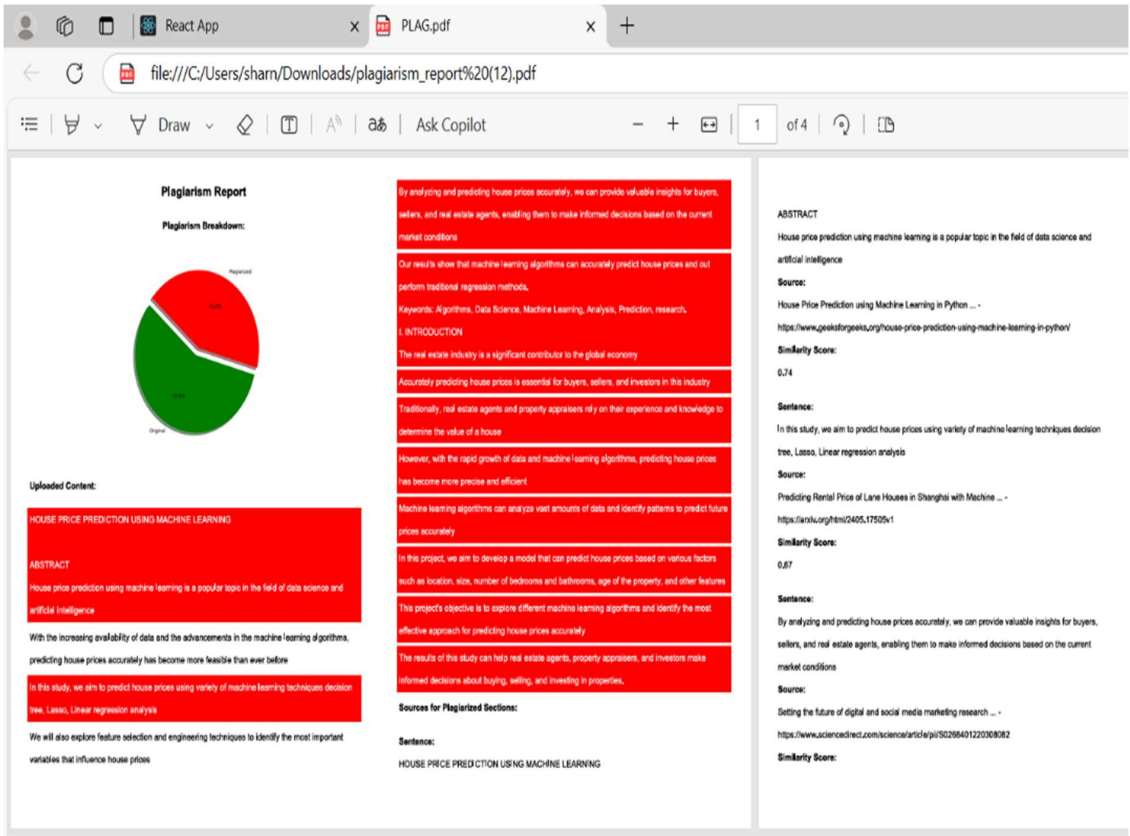


Figure 5.6 Generated Report in PDF form

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

Plagiarism has become a pressing problem in academic and professional environments due to the ever-growing use of online platforms and the internet. Plagiarism can be solved by algorithms that make use of machine learning to detect replicated content. In this research paper, we suggested a plagiarism detector that uses NLP and machine learning.

Since there is extensive utilization of digital media and the internet, plagiarism is now a threat in academic as well as business settings. It can be prevented using machine learning algorithms that identify copied or near-copied content. We propose a plagiarism detection system that incorporates Natural Language Processing (NLP) tools along with machine learning models in this research. Our framework measures textual similarity in terms of cosine similarity and produces a score indicating the degree of duplication among documents.

Experimental findings confirm that the proposed model is more effective and precise than traditional methods of plagiarism detection. The system facilitates the advancement of academic honesty by detecting instances of plagiarized content, hence discouraging academic dishonesty. Although the model works well, there are possibilities for improvement — especially in enhancing the text preprocessing techniques. Improvements in the future can include the integration of machine learning techniques and the creation of a friendlier and more interactive frontend.

The web interface of the system is implemented using the Flask framework, providing users with a convenient means to enter text to be checked for plagiarism. Through the application of sophisticated machine learning algorithms, such as cosine similarity embeddings, the checker is capable of generating accurate and reliable results.

In short, this plagiarism detection tool is a useful and effective solution for educators, students, researchers, and content producers. By assisting with maintaining originality and keeping up ethical standards in writing, the tool makes a positive contribution towards the broader aim of developing a culture of authenticity in written communication.

6.2 Future Scope

The future development of plagiarism detection tools presents numerous opportunities for growth and innovation.

Below are several key areas that could be explored:

- **Enhanced Accuracy:** Research can aim to refine the algorithms and methodologies used to identify plagiarism. This may involve incorporating advanced natural language processing (NLP), machine learning, or deep learning technologies to improve precision and minimize false positives and false negatives.
- **AI Content Recognition:** With the rise of AI-generated content, new algorithms can be designed to recognize distinctive AI writing patterns. This would make it possible to verify instances of plagiarism involving AI-generated text.
- **Support for Multiple Languages:** Given the global nature of content creation, expanding language support would be beneficial. Implementing language-specific tools and models can allow for more accurate plagiarism detection across diverse linguistic contexts.
- **Improved Text Preprocessing:** To better manage variations and irregularities in user input, the system can adopt more advanced text preprocessing techniques. Addressing issues like inconsistent punctuation, spelling, capitalization, and other textual nuances can significantly improve detection accuracy.
- **Web Platform Integration:** Seamless integration with widely used platforms such as learning management systems (LMS), content management systems (CMS), or collaborative writing tools can offer users a streamlined experience. This could enable real-time plagiarism detection and content verification within the platform itself.
- **API and External Database Integration:** Linking the detection tool with third-party APIs or academic databases and digital libraries can expand its comparison base. This enhances the system's ability to identify plagiarized content across a wider range of sources.

By advancing these areas, plagiarism detection tools can become more effective, accurate, and adaptable to the evolving challenges in content integrity and originality.

References

[1] Y. P. Ma et al., "Plagiarism Detection Based on Lexical and Syntactic Features Using Convolutional Neural Network," in Proceedings of the IEEE International Conference on Computational Science and Computational Intelligence, 2019, pp. 698-703.

[2] S. V. Moravvej, S. J. Mousavirad, D. Oliva, G. Schaefer and Z. Sobhaninia, "An Improved DE Algorithm to Optimise the Learning Process of a BERT-based Plagiarism Detection Model," 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 2022, pp. 1-7

[3] El-Rashidy, Mohamed A., et al. "Reliable plagiarism detection system based on deep learning approaches." Neural Computing and Applications 34.21 (2022): 18837-18858.

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

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

[6]. Chavan, Hiten, et al. "Plagiarism detector using machine learning." International Journal of Research in Engineering, Science and Management 4.4 (2021): 152-154.

[7] Siddharth Tata, Suguri Charan Kumar and Varampati Reddy Kumar, “Extrinsic Plagiarism Detection Using Fingerprinting”, International Journal of Computer Science and Technology (IJCST) Vol. 10, Issue 4, Oct - Dec 2019.

[8] 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.

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

Plagiarism Report

DrillBit

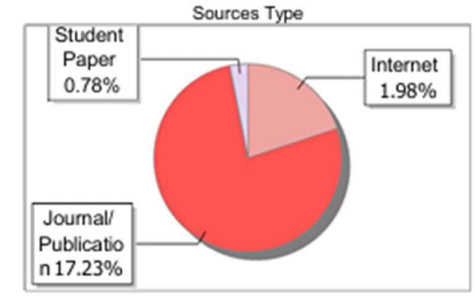
The Report is Generated by DrillBit Plagiarism Detection Software

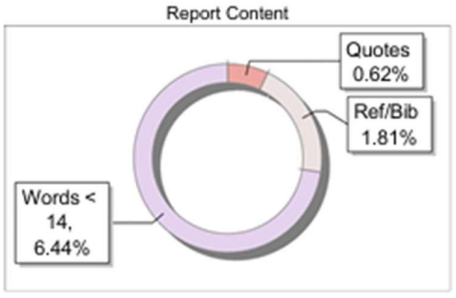
Submission Information

Author Name	G20
Title	ABC
Paper/Submission ID	3593807
Submitted by	amanpreet_brar@gndec.ac.in
Submission Date	2025-05-08 12:47:11
Total Pages, Total Words	66, 14791
Document type	Project Work

Result Information

Similarity **18 %**





Exclude Information

Quotes	Excluded
References/Bibliography	Excluded
Source: Excluded < 14 Words	Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

Github Repository QR Code