TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A PROJECT REPORT ON INTRA-CLASS PLAGIARISM CHECKER

SUBMITTED BY:

AVAHAN TAMRAKAR (PUL077BCT015)

JAVED ANSARI (PUL077BCT033)

KRIPESH NIHURE (077BCT037)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

March 10, 2024

Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **”Intra-Plagiarism Checker”** submitted by **Avahan Tamrakar**, **Javed Ansari** and **Kripesh Nihure** in partial fulfillment of the requirements for the Bachelor’s degree in Electronics & Computer Engineering.

....	....
Supervisor	Internal examiner
Assistant Professor	Assistant Professor
Department of Electronics and Computer Engineering,	Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.	Pulchowk Campus, IOE, TU.

....
External examiner

Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

Acknowledgments

This project report stands as a collective effort of a dedicated team of three members, each bringing unique perspectives and expertise to the table.
First and foremost, we would like to express our sincere gratitude towards our cluster supervisors and cluster head and project supervisor, **Dr. Basanta Joshi** sir, from the Department of Electronics and Computer Engineering, Pulchowk Campus for providing us with the opportunity to work on this project. And for providing invaluable guidance, encouragement, and unwavering support throughout the process of our project.
We express our deep sense of gratitude towards our project supervisor **Asst. Prof Bibha Sthapit** for providing guidance when needed. We are also grateful to our peers and mentors who have offered their encouragement,constructive critiques, and intellectual engagement, fostering a stimulating academic environment conducive to innovation and excellence.
Finally, we would like to acknowledge the support of the Department of Electronics and Computer Engineering, our family and friends, who have been a constant source of motivation and encouragement throughout this journey.

Thank you all for your contributions and support.

Abstract

The application (plagiarism checker) allows teachers and students to create and join virtual classrooms, where students can upload assignments for assessment. Utilizing techniques such as document vectorization and cosine similarity calculation, our engine analyzes the uploaded documents to determine the percentage of similarity between them. Through a process of document normalization and algorithmic comparison, potential instances of plagiarism are identified and presented to teachers for review.

Our project focuses on developing a web-based application aimed at facilitating plagiarism detection within educational settings, inspired by the functionality of renowned platforms like Google Classroom. The primary objective is to provide teachers and students with a user-friendly tool for managing assignments and identifying similarities between submitted documents.

The project's methodology involves tokenizing documents using an N-grams technique, normalizing them, and performing similarity calculations using the cosine similarity index. The results are then visualized through a side-by-side comparison of uploaded PDFs, highlighting similar segments found by the algorithm.

In conclusion, our project offers a comprehensive solution for detecting plagiarism in educational environments, providing a valuable resource for maintaining academic integrity and fostering a culture of originality among students.

Keywords: *plagiarism, document-vectorization, N-grams, Cosine-similarity*

Contents

List of Figures

1. Introduction

1.1 Background

Plagiarism within educational institutions, particularly among students in the same class, poses a significant challenge to maintain academic integrity. It is a well known fact that students have resorted to copying from their peers' work, compromising the fairness and credibility of assessments. Traditional methods of plagiarism detection often fall short in identifying intra-class plagiarism due to their focus on external sources. Hence, there is a critical need for tailored solutions that can effectively address plagiarism within the classroom environment. So we developed an intra-class plagiarism checker specifically designed to assess students' work within the same class that will foster a culture of originality and honesty among students.

1.2 Problem statements

The prevalence of intra-class plagiarism undermines the educational process by devaluing the efforts of honest students and distorting academic outcomes. Teachers face the challenge of detecting and addressing instances of plagiarism among their students, which requires substantial time without specialized tools like this. Teachers will no longer have to go through hundreds of assignments to see if anyone has copied from peers' work and students will no longer be able to cheat from assignments knowing some sort of checker has been implemented. It will assert academic honesty and integrity among the students and also provide an effective evaluation method for teachers that would be fruitful for all stakeholders.

1.3 Objectives

- Create a web based UI for plagiarism checker
- Mathematical based implementation to check similarity
- Check for plagiarism within the class and if detected, point out the extent of similarity

1.4 Scope

- scope is limited to within the classroom, does not deal with web based plagiarism
- can be used by schools and colleges of all levels

2. Literature Review

There are many existing systems for detecting plagiarism by comparing the documents with a vast database of past documents and online resources. Plagiarism detection systems for pointing out the similarity between two documents exist as well.

2.1 Related Work

2.1.1 Existing system

Web-based plagiarism detection systems

The existing systems designed for educational settings that often employ similarity detection algorithms to compare submitted documents against a database of existing documents to identify similarities that may indicate plagiarism include:

1. Turnitin: Turnitin is one of the most widely used plagiarism detection systems in educational institutions. It compares submitted papers against its extensive database of academic content, internet sources, and previously submitted papers to identify potential instances of plagiarism.

2. Grammarly: While Grammarly is primarily known as a writing assistant tool, it also includes a plagiarism detection feature that checks submitted text against a large database of web pages and academic papers to identify potential plagiarism.

3. Plagiarism Checker X: Plagiarism Checker X is a standalone software tool that allows users to check documents for plagiarism by comparing them against online sources and academic databases.

4. Unicheck: Unicheck is a plagiarism detection system specifically designed for educational institutions. It provides similarity detection features for academic papers, assignments, and other documents.

5. MOSS (Measure Of Software Similarity): MOSS is a plagiarism detection system commonly used in computer science and programming courses. It compares source code submissions to identify similarities and potential instances of code plagiarism.

Document based Plagiarism detection systems

1. Draftable: It is an enterprise-grade comparison software tool. It compares documents side-by-side highlighting differences between documents. The tool typically highlights changes and provides a side-by-side comparison to facilitate easy identification of discrepancies.
2. RepostSeo: This comparison search tool will check duplicate content according to your input values. It will not match content all over the internet.

Recent trends

Advancements in machine learning have shown promising possibilities in plagiarism detection. Studies explore the application of machine learning models for identifying similarities in text. These approaches offer the potential for a more nuanced understanding of plagiarism, especially within the unique context of classmates collaborating on assignments.

2.1.2 Challanges and gaps in existing systems

Current plagiarism detection tools lack solutions specifically tailored to handle intra-class plagiarism scenarios. Traditional plagiarism detection systems primarily focus on comparing documents with external databases or online resources, overlooking the unique dynamics of plagiarism within the classroom environment.

2.1.3 Advancements over existing systems

The implementation of intra-class plagiarism detection fills the gap in existing systems by providing a targeted and effective solution for intra-class plagiarism detection. By analyzing similarities exclusively among classmates' submissions, this system offers educators a more relevant and actionable insight into instances of collusion or unauthorized collaboration within their classes.

2.2 Related theory

Most clustering approaches use distance measures to assess the similarities or differences between a pair of objects. Among such measures, the one which we used is Cosine distance.

2.2.1 Text extraction

The textual content can be retrieved from PDFs for analysis and processing. This extraction is essential for plagiarism detection systems to compare the textual content of different documents. Typically, PDF text extraction involves parsing the PDF file and extracting text from each page using libraries or tools designed for this purpose. Once extracted, the text can be processed and analyzed for similarities.

2.2.2 Tokenization

Tokenization is the process of breaking down a document into individual tokens, which can be words, phrases, or other meaningful units of text. This process involves splitting the document into smaller components to facilitate further analysis and processing. In our project, document tokenization is a crucial step in preparing textual assignments for plagiarism detection. By tokenizing documents, we create a structured representation of the text, allowing us to analyze and compare documents at a more granular level. Additionally, tokenization enables the application of techniques such as Bag-of-Words (BoW) or N-grams, which rely on individual tokens to represent the content of the document. Overall, document tokenization plays a fundamental role in the initial preprocessing of textual data, laying the foundation for subsequent analysis and comparison in our plagiarism detection system.

2.2.3 Vectorization

In this context, document vectorization refers to the process of converting textual documents into structured representations that can be compared and analyzed efficiently using various algorithms.
One common method of document vectorization, particularly suitable for our project, is the **Bag-of-Words (BoW)** model. In this approach, each document is represented as a vector where each dimension corresponds to a unique word in the document corpus. The value of each dimension represents the frequency of the corresponding word in the document. By converting documents into BoW vectors, we can compare them based on the occurrence of words, identifying potential similarities between assignments.

Overall, document vectorization remains a fundamental step in the process of plagiarism detection within the updated version of the project. By converting textual documents into structured representations, we enable efficient comparison and analysis, facilitating the identification of potential instances of plagiarism between student submissions.

2.2.4 N-gram

N-grams are a technique used in natural language processing (NLP) to represent sequences of N contiguous words or characters within a document. Instead of considering individual words in isolation, N-grams capture the contextual relationships between words by analyzing sequences of adjacent words. This approach provides a more detailed and contextually rich representation of the text, allowing for more nuanced analysis and comparison. In the context of our project, N-grams can be applied during the document vectorization process to enhance the representation of textual documents. For example, when using word-based N-grams, a document is segmented into sequences of N adjacent words, where N represents the number of words in each sequence. These sequences, or N-grams, capture not only individual words but also the relationships between adjacent words within the document.

By incorporating N-grams into the document vectorization process, we can capture more complex patterns and structures within the text, which may not be adequately represented by individual words alone. This enables us to better capture the semantic meaning and context of the documents, facilitating more accurate comparison and analysis. In summary, N-grams provide a powerful tool for capturing contextual relationships between words in textual data. By segmenting documents into sequences of contiguous words, N-grams enhance the representation of text, enabling more nuanced analysis and comparison, particularly in tasks such as plagiarism detection. When N = 1, it is equivalent to bag-of-words (BoW)

2.2.5 Cosine index

The cosine similarity index is a measure of similarity between two non-zero vectors in an inner product space that measures the cosine of the angle between them. It measures clustering that determines the cosine of the angle between two vectors given by the following formula:

$$Similarity(A, B) = cos(theta) = \frac{A.B}{|A||B|}$$

Here $'\theta'$ gives the angle between two vectors and A, B are n-dimensional vectors.

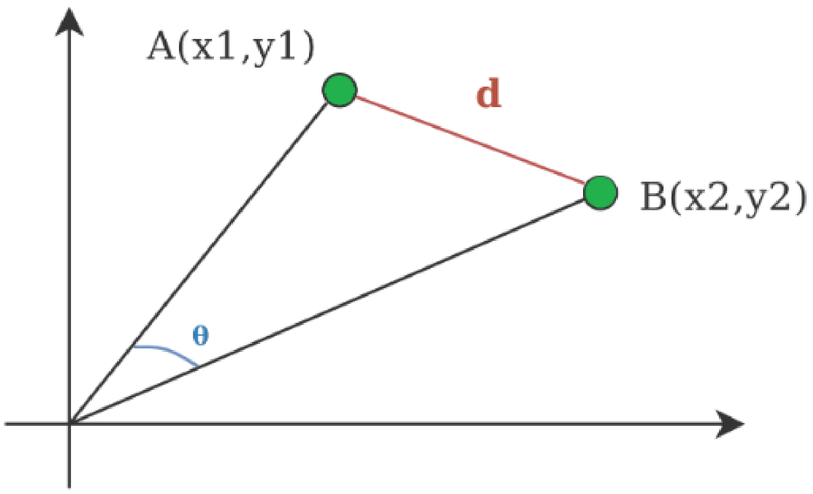


Figure 2.1: Cosine distance

In the context of text analysis and plagiarism detection, the vectors typically represent the term frequencies of words in the documents.
Compute the dot product of the normalized vectors. The dot product of two vectors A and B is given by:

DotProduct(A,B) $= \sum_{i=1}^{n} A_i.B_i$

Where Ai and Bi are the components of vectors A and B.
The magnitudes of the vectors A and B are given as:

$$Magnitude(A) = \sqrt{\sum_{i=1}^{n} A_i^2}$$

$$Magnitude(B) = \sqrt{\sum_{i=1}^{n} B_i^2}$$

Cosine similarity is obtained as:

$$CosineSimilarity(A,B) = \frac{DotProduct(A.B)}{Magnitude(A).Magnitude(B)} = \frac{\sum_{i=1}^{n} A_i.B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}.\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Cosine similarity is the best fit for this purpose because of the following reasons:

- Robustness to Document Length
  Cosine similarity is robust to variations in document length. It measures the cosine of the angle between two vectors, representing the document in a high-dimensional space. This means that even if the documents being compared are of different lengths, the cosine similarity can still provide a meaningful measure of similarity.

- Ignores the Magnitude Differences
  Cosine similarity considers the direction of the vectors rather than their magnitude. This is beneficial because it makes the similarity measure insensitive to the overall length of the documents. It focuses on the relative frequencies of terms, which is often more important in plagiarism detection.

- Effective for Text Representation
  In text analysis, documents are often represented as bags-of-words or vectors of word embeddings. Cosine similarity is well-suited for comparing these vector representations because it captures the similarity in the distribution of words, regardless of their frequency.

3. Methodology

3.1 Feasibility study

Before embarking on the development journey of our intra-class plagiarism checker, we first conducted a thorough feasibility study to determine whether the project was technically and practically feasible. This study involved assessing various factors to ensure the project's viability. We began by exploring the availability of appropriate algorithms and tools essential for key tasks such as text extraction, tokenization, and similarity analysis. These processes are vital for accurately comparing documents and identifying potential instances of plagiarism among student submissions.

Furthermore, our feasibility study delved into the computational requirements of the project, considering the resources needed to process large volumes of documents efficiently. We evaluated the scalability and performance of the proposed system to ensure it could handle the expected workload without compromising speed or reliability.

Additionally, we investigated the integration of our plagiarism detection system with existing online classroom platforms. This integration would streamline the submission and analysis of student assignments, providing a seamless user experience for both teachers and students. By examining these aspects comprehensively, our feasibility study laid the groundwork for a successful and effective implementation of the intra-class plagiarism checker.

3.2 Requirements analysis

3.2.1 Functional requirements

1. User Authentication: The system must allow teachers and students to create accounts and log in securely. Different access levels should be provided for teachers and students, enabling them to perform specific actions within the platform.

2. User friendly interface: Teachers should be able to create virtual classrooms and invite students to join. Students, in turn, should be able to join existing classrooms and view assignments posted by their teachers.

3. Assignment Upload: Students should have the capability to upload assignments to their respective classrooms. The system must support PDF file formats.

4. Plagiarism Detection: The core functionality of the system is to detect similarities between submitted assignments. It should analyze uploaded documents using document vectorization and cosine similarity calculation methods, providing teachers with a percentage of similarity between pairs of documents.

3.2.2 Non-fuctional requirements

1. Security: The system must prioritize data security, ensuring that user accounts, assignments, and other sensitive information are protected from unauthorized access or tampering.

2. Scalability: The application should be able to handle a growing number of users and documents as more teachers and students join the platform and submit assignments. It should scale efficiently to accommodate increasing demand without compromising performance.

3. Usability: The user interface should be intuitive and easy to navigate, catering to users with varying levels of technical expertise. Clear instructions and feedback mechanisms should be provided to guide users through the assignment submission and plagiarism detection process.

4. Performance: The system should be responsive and reliable, providing timely feedback to users when uploading assignments and generating plagiarism reports. It should minimize downtime and latency to ensure a seamless user experience.

3.3 System design

During the system design phase, our primary focus was on crafting the architecture, components, and their interactions for the plagiarism detection system. We aimed to ensure that the system could effectively identify similarities between student submissions while maintaining a user-friendly interface for teachers and students alike.

3.3.1 Frontend

For the frontend interface, our objective was to create an intuitive platform for teachers and students to engage with the system seamlessly. We developed features such as class creation, joining classes, assignment creation, document submission, and result viewing. These functionalities were designed with the convenience of teachers in mind, enabling them to efficiently manage their classes, while also providing students with a straightforward means of submitting assignments.

3.3.2 Backend

we meticulously designed the backend architecture to handle complex processes such as document processing, similarity analysis, and data storage. We carefully selected appropriate technologies and frameworks to support these functionalities, ensuring optimal performance and scalability. The backend components were intricately crafted to integrate seamlessly with the frontend interface, facilitating smooth communication and interaction between different layers of the system.
Through meticulous planning and strategic selection of technologies, our goal was to deliver a plagiarism detection system that not only meets technical requirements but also provides a user-friendly experience for both teachers and students.

3.3.3 Testing

Testing was conducted to validate the functionality, accuracy, and reliability of the plagiarism detection system. This included unit testing of individual components, integration testing to ensure seamless interaction between frontend and backend, and system testing to evaluate the overall performance. Test scenarios were devised to simulate different usage scenarios and assess the system's ability to detect plagiarism accurately and efficiently.

3.4 Implementation

This section provides insights into how the different aspects of the system was actually implemented, so the reader can gain understanding of the inner-workings of our system.

The system architecture comprised frontend and backend components designed to facilitate seamless interaction between teachers and students. The frontend interface was developed using web technologies to create an intuitive online classroom environment where teachers could create classes, assign assignments, and view student submissions. The backend was implemented using programming languages such as Python, incorporating libraries for PDF text extraction, tokenization, and cosine similarity calculation. Firebase Storage was utilized for secure storage of PDF submissions.

The Plagiarism Checker Engine processes uploaded documents, examining them for similarities through advanced algorithms. By comparing text content, the engine identifies potential instances of plagiarism, providing educators with insights into the originality of student submissions. This crucial tool enhances academic integrity by detecting unauthorized collaboration or content reuse, fostering a culture of honesty and originality within educational environments.

3.4.1 Algorithm

1. Text Representation:

   Implement natural language processing (NLP) techniques to preprocess text content, including tokenization, lemmatization, and removal of stop words.

2. Document Vectorization:

   Bag-of-Words (BoW): Represent each document as a vector where each dimension corresponds to a unique word, and the value represents the frequency of that word in the document.

   N-gram: N-grams are contiguous sequences of n items from a given sample of text or data, typically used in natural language processing (NLP) to capture patterns and relationships within text data. For example, in the context of text, a 2-gram (also known as a bigram) would represent two consecutive words occurring in the text. N-grams provide a way to analyze the structure and context of textual data by considering sequences of items rather than individual items alone.

   In the case where n=1, each individual item (typically words) in the text is considered as a separate unit, essentially creating a bag of words (BoW) representation. This approach treats each word independently of its surrounding context, resulting in a loss of sequential information. However, it still provides valuable insight into the frequency and distribution of individual words within the text, forming the basis of many text processing and analysis techniques, such as document classification and sentiment analysis.

   When using an n-gram approach with n=2, each unit consists of pairs of adjacent items, such as pairs of consecutive words. This allows capturing some sequential information compared to the bag of words approach (where n=1), but still maintains a certain level of simplicity compared to higher-order n-grams.

   Here's an example of how the text "The quick brown fox jumps over the lazy dog" would be represented using bigrams (2-grams):

   Bigrams:

   - The quick
   - quick brown
   - fox jumps

- jumps over
- over the
- the lazy
- lazy dog

In this representation, each bigram represents a pair of consecutive words in the text. This allows capturing some contextual information about how words are used together while still maintaining a relatively simple representation compared to higher-order n-grams. This type of representation is often used in various natural language processing tasks, including language modeling, machine translation, and part-of-speech tagging.

When using an n-gram approach with n=3, each unit consists of triplets of adjacent items, such as triplets of consecutive words. This captures even more sequential information compared to bigrams (n=2) but increases the complexity of the representation.

Here's an example of how the text "The quick brown fox jumps over the lazy dog" would be represented using trigrams (3-grams): Original Text: "The quick brown fox jumps over the lazy dog"

Trigrams:

- The quick brown
- quick brown fox
- brown fox jumps
- fox jumps over
- jumps over the
- over the lazy
- the lazy dog

In this representation, each trigram represents a triplet of consecutive words in the text. This captures more contextual information compared to bigrams by considering groups of three words at a time. Trigrams are commonly used in various natural language processing tasks where capturing more detailed sequential information is important, such as text generation, speech recognition, and named entity recognition.

3. Vector normalization:

   Normalize each document vector to have a unit length. This step is essential for cosine similarity as it focuses on the direction, not the magnitude, of the vectors.

4. Cosine Similarity Computation:

   The cosine similarity index is a measure of similarity between two non-zero vectors in an inner product space that measures the cosine of the angle between them. In the context of text analysis and plagiarism detection, the vectors typically represent the term frequencies of words in the documents. Here's how the algorithm for computing cosine similarity is implemented:

$$CosineSimilarity(A, B) = \frac{DotProduct(A.B)}{Magnitude(A).Magnitude(B)} = \frac{\sum_{i=1}^{n} A_i.B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}.\sqrt{\sum_{i=1}^{n} B_i^2}}$$

   Where Ai and Bi are the components of vectors A and B.

   The magnitudes of the vectors A and B are given as:

$$Magnitude(A) = \sqrt{\sum_{i=1}^{n} A_i^2}$$

$$Magnitude(B) = \sqrt{\sum_{i=1}^{n} B_i^2}$$

5. Convert into percentage:

   To convert the output of cosine similarity calculation, which is a real number ranging from 0 to 1, into percentage form, you can simply multiply it by 100. This will scale the similarity score to a range from 0% to 100%, making it more readable and easily displayable.

   How we can achieve this is explained by the figure below:

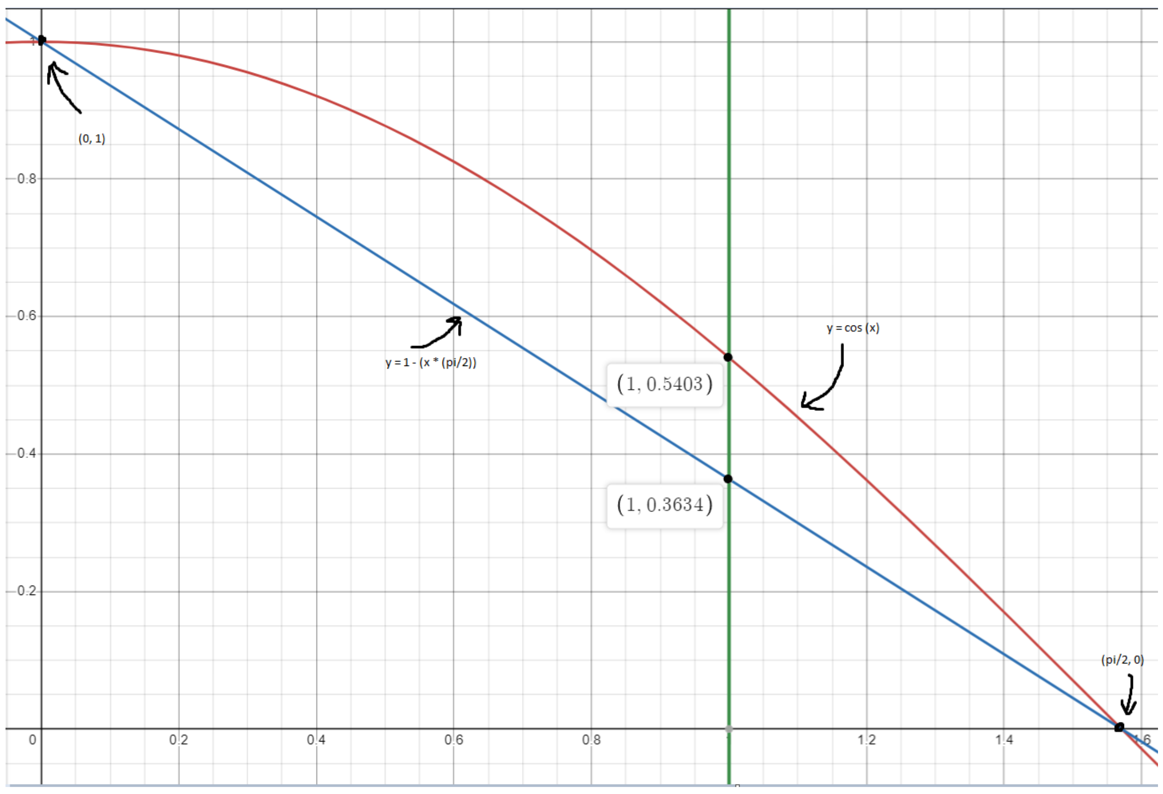


Figure 3.1: Normalization of output

Here, as we can see, if the similarity value obtained was found to be 0.5403 from cosine similarity, then the document is not actually 54.03% . Rather, it is 36.34% similar.

6. Listing groups of Plagiarized documents:
   Eg: [['flowchart_modified.drawio.pdf', 'flowchart.drawio.pdf', 'Untitled Diagram.drawio.pdf'] , ['sample5.pdf', 'tala_mathi.pdf']]
   In the provided example, two groups are identified: one comprising documents like 'flowchart_modified.drawio.pdf', 'flowchart.drawio.pdf', and 'Untitled Diagram.drawio.pdf', which share similarities suggesting they belong to a common category, possibly different versions of flowchart diagrams.

   The second group includes documents such as 'sample5.pdf' and 'tala_mathi.pdf', which exhibit similarities among themselves, indicating they belong to a separate category from the first group. This approach facilitates efficient organization and analysis of document collections, aiding tasks such as document clustering and classification in various domains, including text mining and information retrieval.

   Here, ['flowchart_modified.drawio.pdf', 'flowchart.drawio.pdf', 'Untitled Diagram.drawio.pdf'] are similar documents of one kind while ['sample5.pdf', 'tala_mathi.pdf'] are similar documents of another kind.

7. Generate PDFs with highlighted sections:
   The algorithm is implemented to create highlighted PDF files that emphasize common words or phrases found among a selected group of PDF documents. Here's a summary of its functionality:

   Input Parameters: The function takes three parameters: source_folder, destination_folder, and file_list.

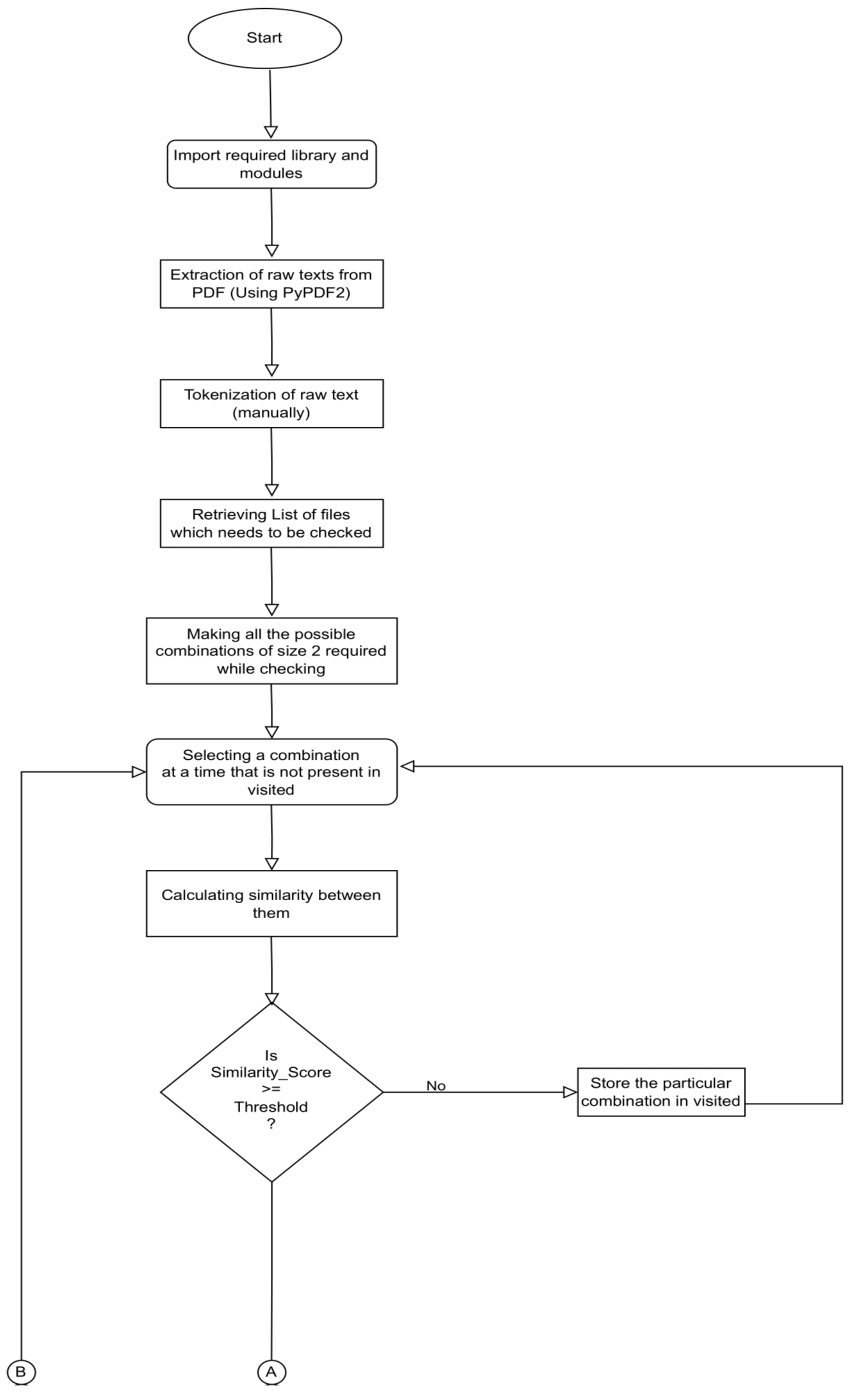
   - source_folder: The folder containing the original PDF files.
   - destination_folder: The folder where the highlighted PDF files will be saved.
   - file_list: A list of filenames of the PDF documents selected for comparison.

   Preprocessing: It extracts the text content from each PDF file in the file_list using the extract_text_from_pdf function. Then, it tokenizes the text into n-grams.
   Common Words Extraction: It identifies common words or phrases among the text content of all selected PDFs. This is achieved by finding the intersection of the n-grams from each PDF using set operations.

   Highlighting: For each PDF file in the file_list, the function iterates through the common words and highlights occurrences of these words in the PDF using the highlight_word_in_pdf function. This function utilizes the fitz library to add highlight annotations to the PDF pages.

   Saving Highlighted PDFs:The highlighted PDF files are saved in the destination_folder. Each PDF file is saved with the same filename as the original file but with "_highlighted" appended to the filename.

3.4.2 Flowchart


Start
Import required library and modules
Extraction of raw texts from PDF (Using PyPDF2)
Tokenization of raw text (manually)
Retrieving List of files which needs to be checked
Making all the possible combinations of size 2 required while checking
Selecting a combination at a time that is not present in visited
Calculating similarity between them
Is Similarity_Score >= Threshold ?
No
Store the particular combination in visited
B
A

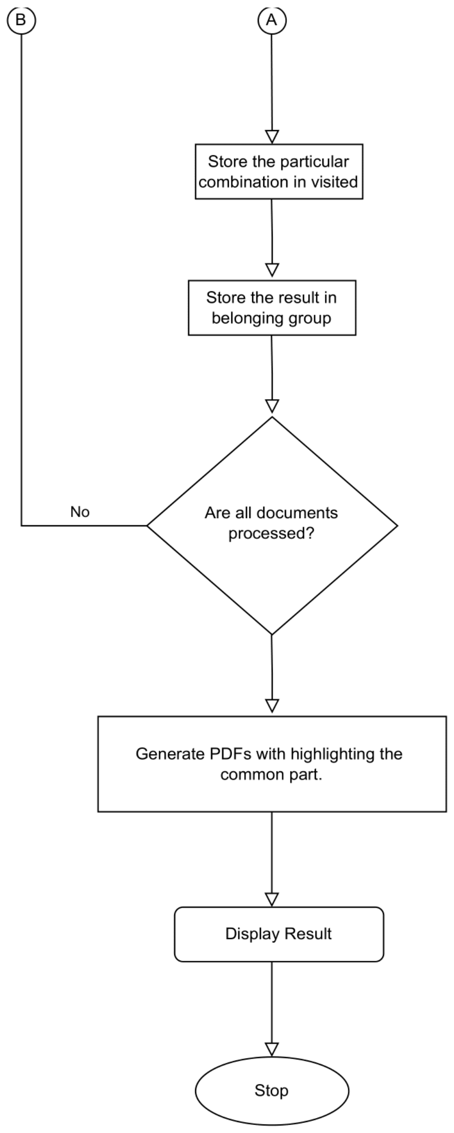

B
A
Store the particular combination in visited
Store the result in belonging group
No
Are all documents processed?
Generate PDFs with highlighting the common part.
Display Result
Stop

Figure 3.2: flowchart for engine

Dependencies

- PyPDF2: PyPDF2 is a versatile Python library designed for interacting with PDF files. It offers capabilities for extracting text and metadata from PDF documents, enabling users to analyze and manipulate their content programmatically. Additionally, PyPDF2 provides functionalities for performing various operations on PDFs, including merging, splitting, rotating, and encrypting files, making it a valuable tool for managing PDF documents efficiently.
- Collections: The 'collections' module in Python extends the functionality of built-in

data structures with specialized alternatives such as OrderedDict, defaultdict, Counter, and namedtuple. These classes offer efficient solutions for tasks like preserving order, managing default values, counting elements, and representing structured data with named fields.

- Itertools: The ‘itertools‘ module in Python provides a collection of tools for efficient iteration over iterable objects, offering functions for generating combinations, permutations, and Cartesian products. It focuses on memory-efficient techniques, allowing for the manipulation of large data sets or infinite sequences without storing them in memory.

- PyMuPDF: PyMuPDF is a Python library that provides bindings to the MuPDF library, allowing developers to work with PDF files programmatically. It offers functionalities to read, write, and manipulate PDF documents, including features such as extracting text, images, metadata, and performing various operations like merging, splitting, and encrypting PDFs. PyMuPDF is well-suited for tasks such as text extraction, document analysis, and PDF manipulation within Python applications.

3.4.3 web-application

Front end: HTML,CSS,Js

The user interface (UI) of the plagiarism checker is developed using HTML, CSS, and JavaScript, with a significant emphasis on the React JavaScript library. HTML provides the structure of the UI, CSS styles the UI elements, and JavaScript adds interactivity and dynamic behavior to the UI components. React is used heavily to create reusable UI components and manage the application's state efficiently. The frontend allows users, both teachers, and students, to interact with the system, submit assignments, view plagiarism detection results, and navigate the application seamlessly.
The frontend of the plagiarism checker is structured around screens and components, leveraging modularity and reusability to create a cohesive user interface. Screens encapsulate distinct functionalities and user interactions, while components represent reusable UI elements that enhance consistency and maintainability across the application.
Screens

- Class Screen:

  The Class screen provides a comprehensive view of the class details, assignments, and student submissions. It utilizes the Navbar component for navigation and the Announcement component to display important announcements or updates.

- Teacher Screen

  The Teacher screen enables teachers to manage classes, create assignments, and evaluate student submissions. It incorporates the Navbar component for navigation and the ClassCard component to display class information and assignments.

- Dashboard Screen

  The Dashboard screen serves as the main hub for users, displaying relevant information and facilitating navigation to other screens. It features the Navbar component for navigation and utilizes the ClassCard component to provide an overview of enrolled classes.

- Login Screen

  The Login screen allows users to authenticate and access their accounts securely. It includes the Navbar component for navigation and the Login component for user authentication.

- Submission Screen

  The Submission screen enables students to submit assignments and view plagiarism detection results. It utilizes the Navbar component for navigation and includes the Submission component for assignment submission and result display.

Components

- Navbar Component

  The Navbar component provides a consistent navigation interface across screens, facilitating seamless navigation between different sections of the application.

- Announcement Component

  The Announcement component displays important announcements or updates to users, ensuring effective communication within the class or platform.

- ClassCard Component

  The ClassCard component represents a reusable UI card that displays class information, including class name, instructor, assignments, and submission status.

Back end: Flask ,firebase

The backend of the plagiarism checker is powered by Flask, a lightweight and flexible web framework for Python. Flask handles server-side logic, request handling, and response generation. Additionally, Firebase is utilized for backend functionalities such as authentication and storage. Firebase provides a scalable and secure authentication mechanism, allowing users to securely log in and access the system's features. Firebase storage is used for storing PDF submissions uploaded by students, ensuring reliable and efficient file storage and retrieval.

API: Axios

Axios is employed as the HTTP client library for making asynchronous HTTP requests from the frontend to the backend server. Axios simplifies the process of making API calls and handling responses, enabling seamless communication between the frontend and backend components of the plagiarism checker. For example, Axios can be used to fetch assignment details from the backend server, submit student submissions, and retrieve plagiarism detection results for display on the UI.

Database: Firestore for authentication, firebase storage

Firestore, a NoSQL cloud database provided by Firebase, serves as the database solution for the plagiarism checker. Firestore is used for storing authentication credentials, user profiles, assignment details, and other relevant data. Additionally, Firebase storage complements Firestore by providing scalable storage for PDF documents uploaded by students. Firestore's real-time database capabilities ensure data consistency and synchronization across multiple users and devices, enhancing the overall reliability and performance of the plagiarism checker.

4. Experimental Setup

To evaluate the effectiveness of the plagiarism detection system, experimental testing was conducted using a dataset of sample documents. The testing involved uploading PDF submissions from students to the system and analyzing the similarity between pairs of documents using the cosine similarity algorithm. Different threshold values were applied to determine the sensitivity of the system in detecting instances of plagiarism within the class submissions. The experimental setup aimed to validate the accuracy and reliability of the plagiarism detection system in identifying potential cases of plagiarism.

4.1 Data selection for testing

A diverse dataset of sample documents representing student submissions from various classes and courses were selected for testing. The dataset includes a mix of assignments, essays, reports, and other academic documents. A set of known plagiarized and non-plagiarized document pairs within the dataset were established. Variations in document length and intentional modifications were introduced to assess system robustness.

4.2 Performance Metrics

The performance metrics to evaluate the plagiarism detection system, such as precision, recall, F1 score, and accuracy can be used. These metrics quantify the system's ability to correctly identify plagiarized and non-plagiarized document pairs. The results were compared against the predefined threshold to determine instances of plagiarism.

- True Positive (TP): The number of plagiarized document pairs correctly identified as plagiarized by the system.
- False Positive (FP): The number of non-plagiarized document pairs incorrectly identified as plagiarized by the system.
- True Negative (TN): The number of non-plagiarized document pairs correctly identified as non-plagiarized by the system.
- False Negative (FN): The number of plagiarized document pairs incorrectly identified as non-plagiarized by the system.

1. Precision: Precision measures the proportion of true positives among all document pairs identified as plagiarized by the system. It is calculated as TP / (TP + FP).

2. Recall (Sensitivity): Recall measures the proportion of true positives among all actual plagiarized document pairs. It is calculated as TP / (TP + FN).

3. Accuracy: Accuracy measures the proportion of correctly identified document pairs (both plagiarized and non-plagiarized) among all document pairs. It is calculated as (TP + TN) / (TP + TN + FP + FN).

4.3 Result Validation

Analyze the experimental results to assess the system's performance in detecting plagiarism. Evaluate the system's accuracy, sensitivity, specificity, and efficiency under different experimental conditions and plagiarism scenarios.

Validate the experimental results through cross-validation and verification with ground truth data. Compare the system's findings with the established baseline to validate its effectiveness in detecting plagiarism accurately.

4.4 Iterative refinement

Iterate on the experimental setup and system parameters based on the observed results. Fine-tune the similarity threshold, tokenization parameters, and other system settings to optimize the system's performance and minimize false positives and false negatives. The similarity threshold determines the minimum level of similarity between documents that will trigger a plagiarism detection alert. Experimentation with different threshold values, such as 60By analyzing the system's performance metrics, such as precision, recall, and F1 score, under different threshold values, you can determine the threshold that minimizes false positives (incorrectly identifying non-plagiarized documents as plagiarized) and false negatives.

5. System design

This chapter presents a comprehensive collection of diagrams aimed at visually representing the system, its components, and operations in our project.

5.1 Block diagram

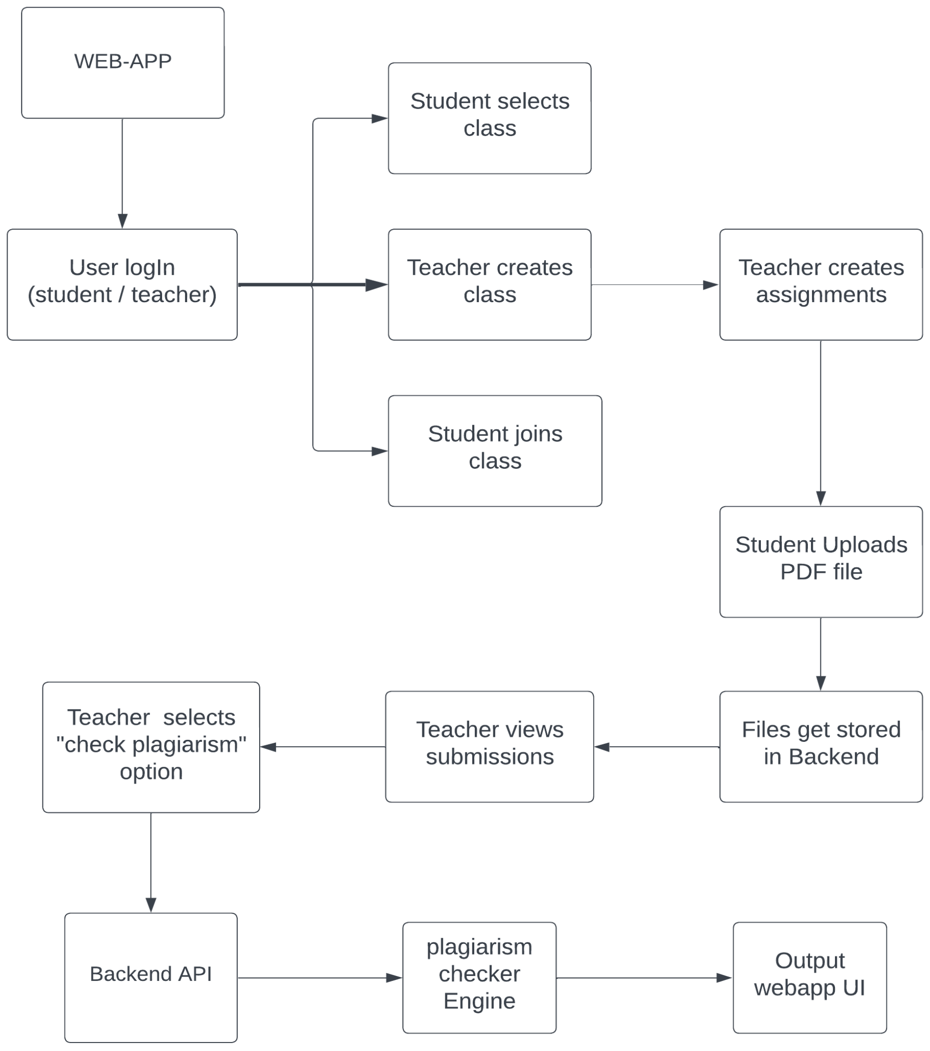

WEB-APP
User logIn (student / teacher)
Student selects class
Teacher creates class
Student joins class
Teacher creates assignments
Student Uploads PDF file
Files get stored in Backend
Teacher views submissions
Teacher selects "check plagiarism" option
Backend API
plagiarism checker Engine
Output webapp UI

Figure 5.1: System Block Diagram

5.2 Sequence diagram

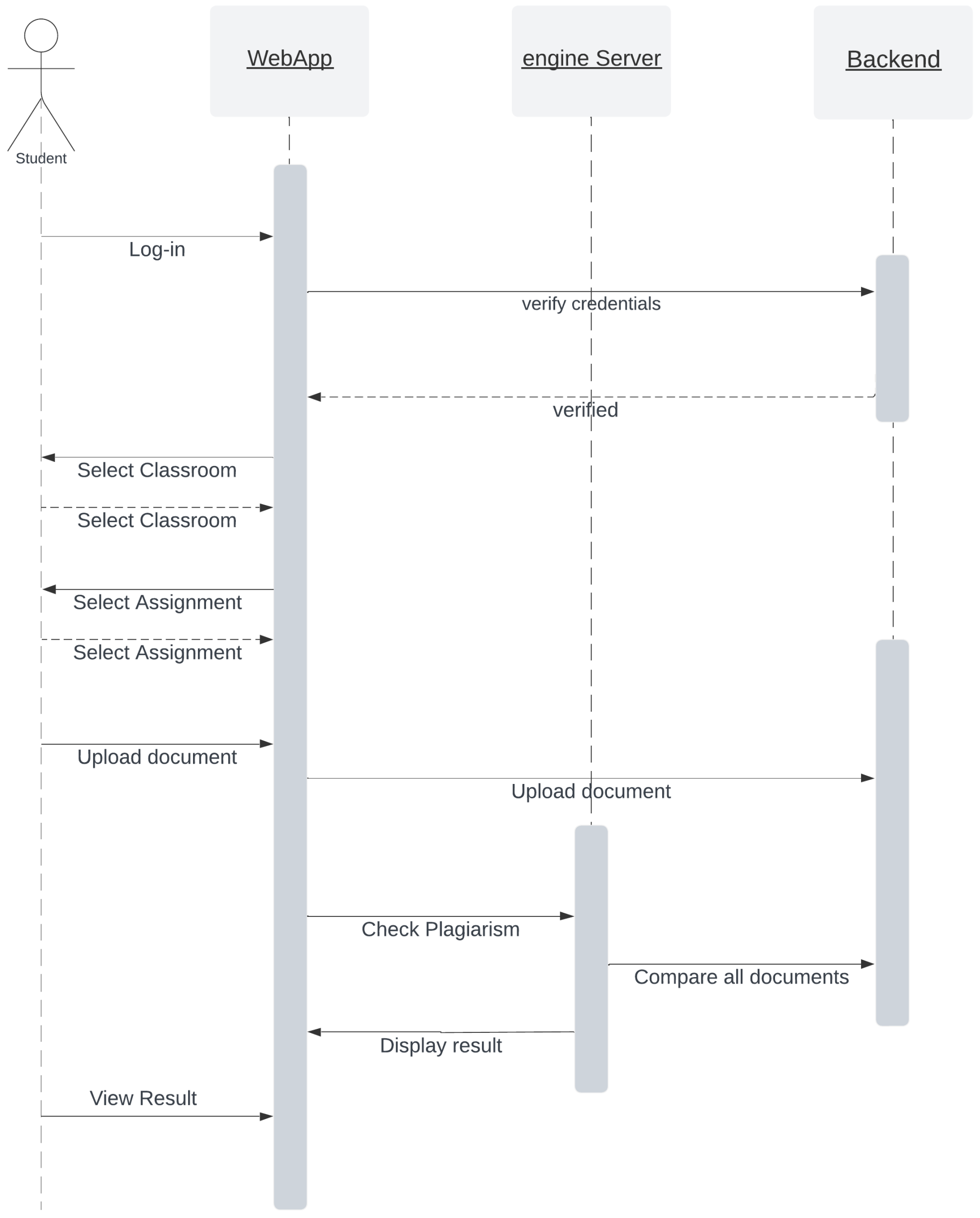


Figure 5.2: Sequence diagram

5.3 use case diagram

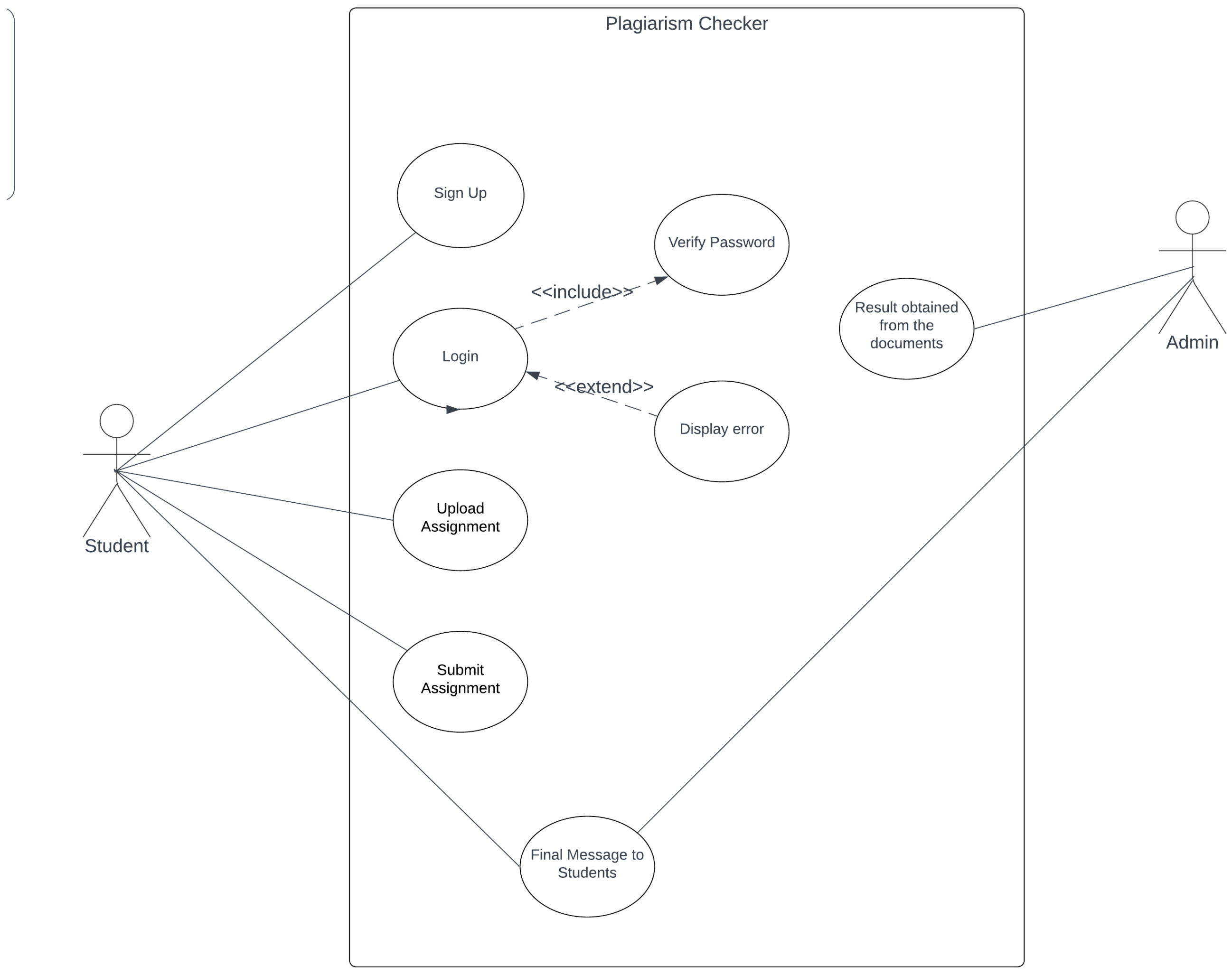


Figure 5.3: Use case diagram

6. Results and Discussion

From this project, we achieved significant milestones in developing an intra-class plagiarism checker tailored for educational environments. The system demonstrated the ability to accurately identify instances of plagiarism among students' submissions, providing valuable insights for educators to address academic dishonesty effectively. Key achievements include the seamless integration with existing online classroom platforms, user-friendly interface, and robust backend processing capabilities utilizing cosine similarity algorithms.
Moreover, the project's success in seamlessly integrating with popular online classroom platforms, such as Google Classroom or Moodle, highlights its adaptability and scalability across diverse educational environments. This integration streamlines the workflow for educators and students, ensuring that plagiarism detection becomes an integral part of the assignment submission and assessment process. Additionally, the user-friendly interface enhances accessibility and usability, enabling educators to navigate the system effortlessly and interpret plagiarism detection results efficiently. With robust backend processing capabilities leveraging cosine similarity algorithms, the system can handle large volumes of student submissions efficiently while maintaining high accuracy in detecting instances of plagiarism. Overall, the intra-class plagiarism checker represents a significant milestone in promoting academic integrity and fostering a culture of originality and honesty within educational institutions.

6.1 Output

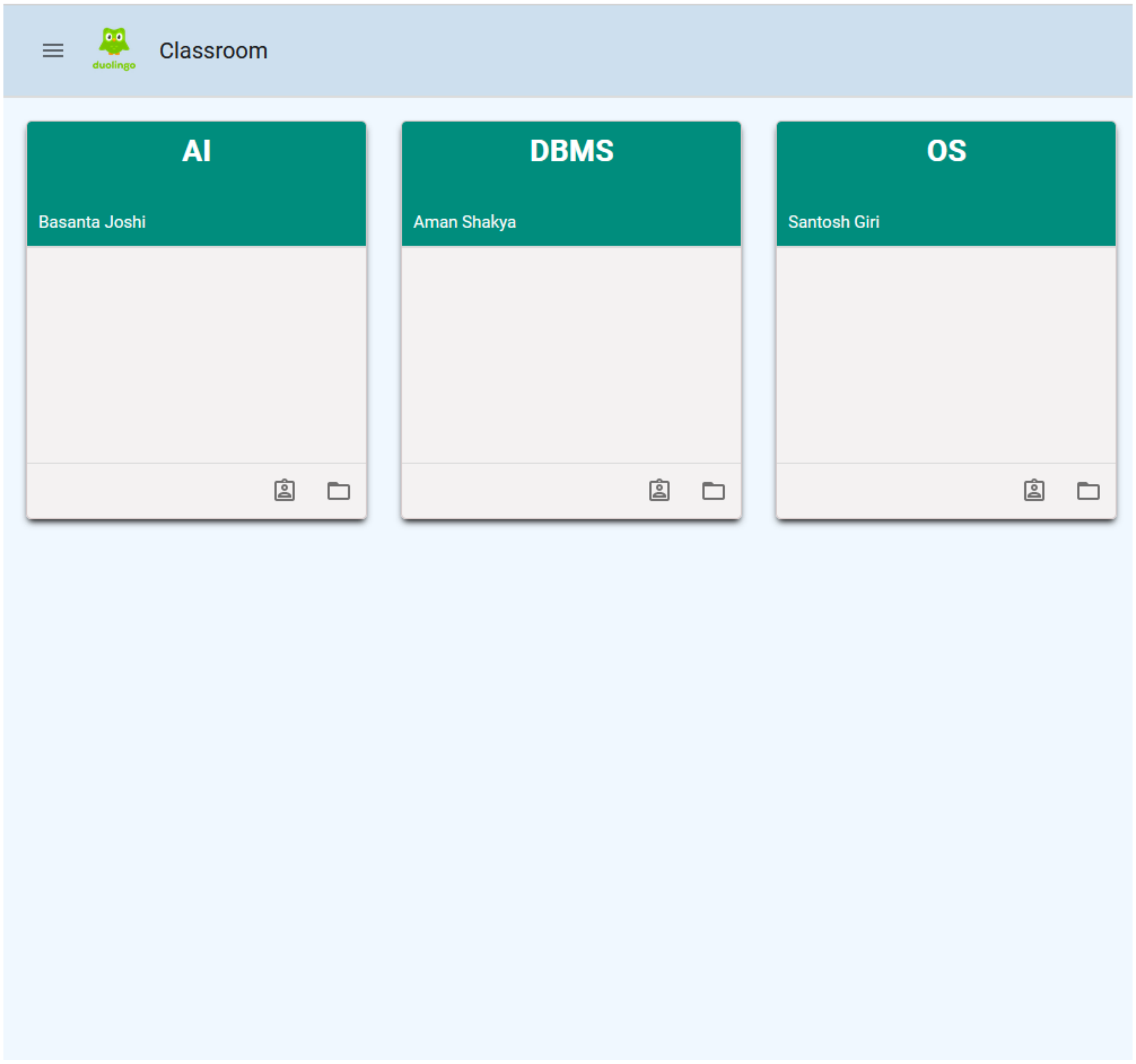
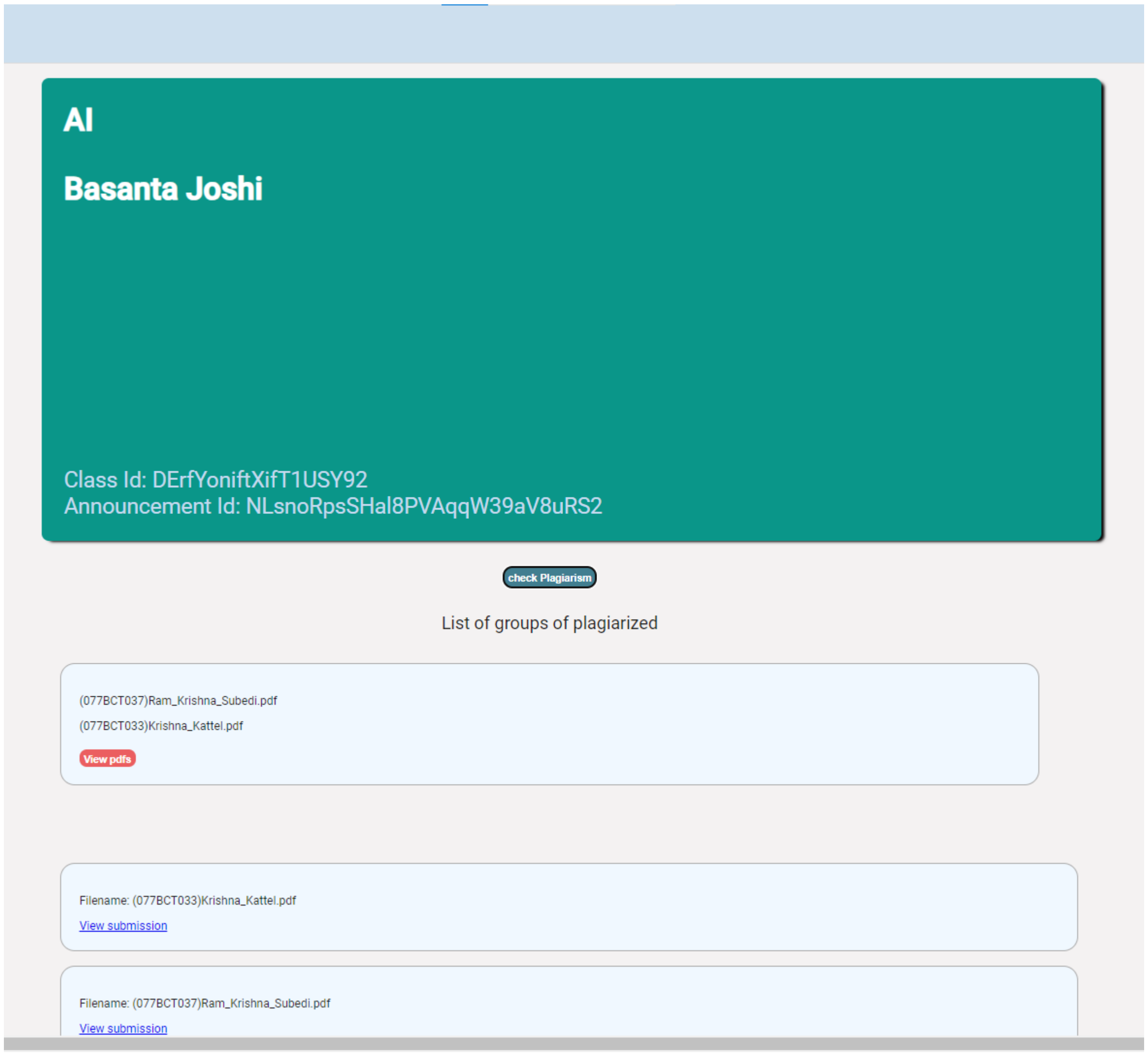


Figure 6.1: Output screenshot-1



AI

Basanta Joshi

Class Id: DErfYoniftXifT1USY92
Announcement Id: NLsnoRpsSHal8PVAqqW39aV8uRS2

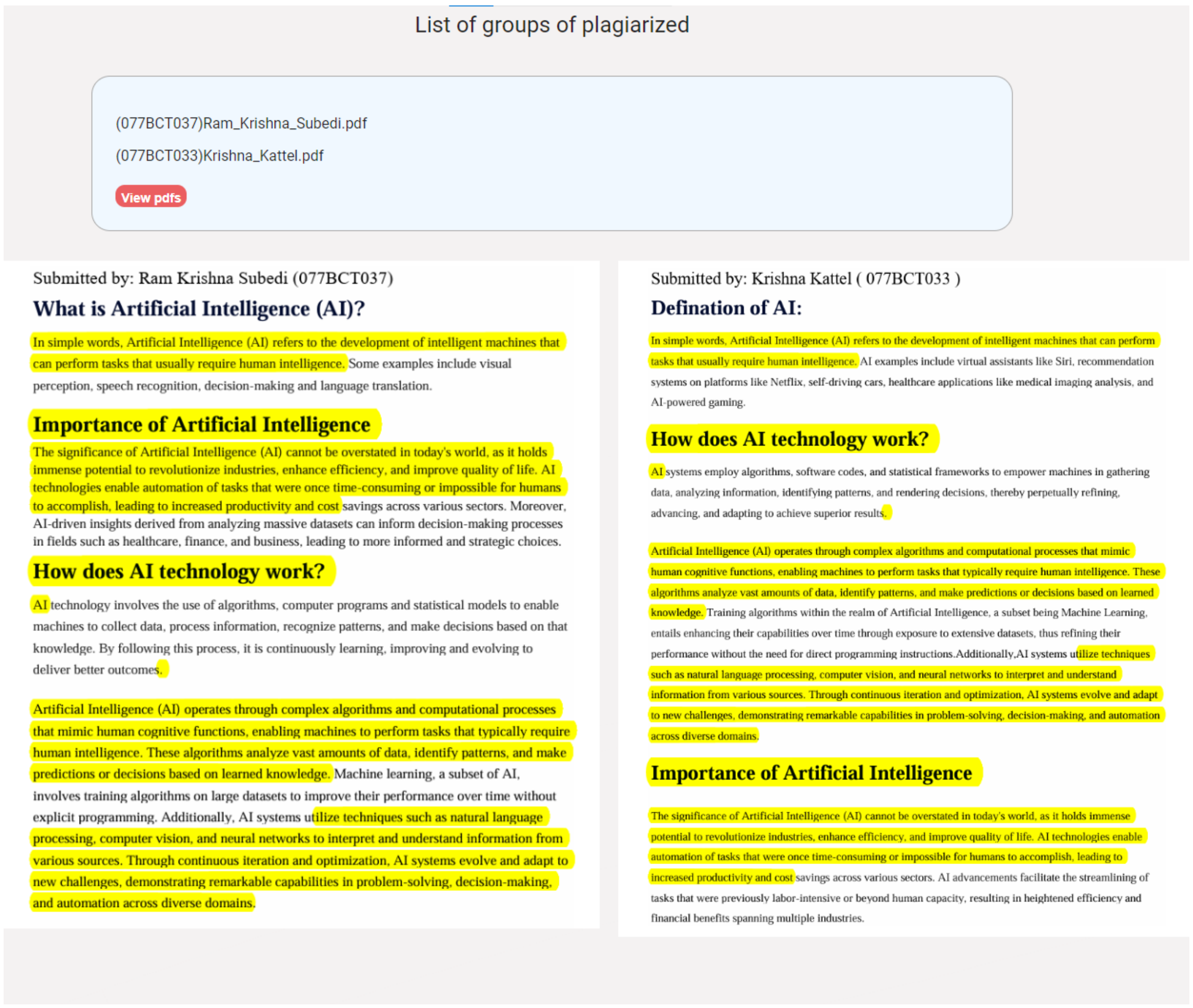
List of groups of plagiarized

(077BCT037)Ram_Krishna_Subedi.pdf
(077BCT033)Krishna_Kattel.pdf

Filename: (077BCT033)Krishna_Kattel.pdf
View submission

Filename: (077BCT037)Ram_Krishna_Subedi.pdf
View submission

Figure 6.2: Output screenshot-2



List of groups of plagiarized

(077BCT037)Ram_Krishna_Subedi.pdf

(077BCT033)Krishna_Kattel.pdf

View pdfs

Submitted by: Ram Krishna Subedi (077BCT037)

What is Artificial Intelligence (AI)?

In simple words, Artificial Intelligence (AI) refers to the development of intelligent machines that can perform tasks that usually require human intelligence. Some examples include visual perception, speech recognition, decision-making and language translation.

Importance of Artificial Intelligence

The significance of Artificial Intelligence (AI) cannot be overstated in today's world, as it holds immense potential to revolutionize industries, enhance efficiency, and improve quality of life. AI technologies enable automation of tasks that were once time-consuming or impossible for humans to accomplish, leading to increased productivity and cost savings across various sectors. Moreover, AI-driven insights derived from analyzing massive datasets can inform decision-making processes in fields such as healthcare, finance, and business, leading to more informed and strategic choices.

How does AI technology work?

AI technology involves the use of algorithms, computer programs and statistical models to enable machines to collect data, process information, recognize patterns, and make decisions based on that knowledge. By following this process, it is continuously learning, improving and evolving to deliver better outcomes.

Artificial Intelligence (AI) operates through complex algorithms and computational processes that mimic human cognitive functions, enabling machines to perform tasks that typically require human intelligence. These algorithms analyze vast amounts of data, identify patterns, and make predictions or decisions based on learned knowledge. Machine learning, a subset of AI, involves training algorithms on large datasets to improve their performance over time without explicit programming. Additionally, AI systems utilize techniques such as natural language processing, computer vision, and neural networks to interpret and understand information from various sources. Through continuous iteration and optimization, AI systems evolve and adapt to new challenges, demonstrating remarkable capabilities in problem-solving, decision-making, and automation across diverse domains.

Submitted by: Krishna Kattel (077BCT033)

Defination of AI:

In simple words, Artificial Intelligence (AI) refers to the development of intelligent machines that can perform tasks that usually require human intelligence. AI examples include virtual assistants like Siri, recommendation systems on platforms like Netflix, self-driving cars, healthcare applications like medical imaging analysis, and AI-powered gaming.

How does AI technology work?

AI systems employ algorithms, software codes, and statistical frameworks to empower machines in gathering data, analyzing information, identifying patterns, and rendering decisions, thereby perpetually refining, advancing, and adapting to achieve superior results.

Artificial Intelligence (AI) operates through complex algorithms and computational processes that mimic human cognitive functions, enabling machines to perform tasks that typically require human intelligence. These algorithms analyze vast amounts of data, identify patterns, and make predictions or decisions based on learned knowledge. Training algorithms within the realm of Artificial Intelligence, a subset being Machine Learning, entails enhancing their capabilities over time through exposure to extensive datasets, thus refining their performance without the need for direct programming instructions.Additionally,AI systems utilize techniques such as natural language processing, computer vision, and neural networks to interpret and understand information from various sources. Through continuous iteration and optimization, AI systems evolve and adapt to new challenges, demonstrating remarkable capabilities in problem-solving, decision-making, and automation across diverse domains.

Importance of Artificial Intelligence

The significance of Artificial Intelligence (AI) cannot be overstated in today's world, as it holds immense potential to revolutionize industries, enhance efficiency, and improve quality of life. AI technologies enable automation of tasks that were once time-consuming or impossible for humans to accomplish, leading to increased productivity and cost savings across various sectors. AI advancements facilitate the streamlining of tasks that were previously labor-intensive or beyond human capacity, resulting in heightened efficiency and financial benefits spanning multiple industries.

Figure 6.3: Output screenshot-3

7. Conclusions

The development of an intra-class plagiarism checker addresses a critical need in educational settings by providing educators with a tool to combat academic dishonesty effectively. The project's success in implementing a scalable and accurate plagiarism detection system underscores its potential for widespread adoption in educational institutions. Future enhancements may include refining the similarity algorithms, incorporating machine learning techniques for improved accuracy, and expanding the system's capabilities to detect different forms of plagiarism across various file formats.

Furthermore, the intra-class plagiarism checker serves as a valuable resource for fostering a culture of academic integrity and promoting ethical conduct among students. By proactively detecting and addressing instances of plagiarism, educators can educate students on the importance of originality, proper citation practices, and intellectual honesty. Additionally, the project's open-source nature allows for collaboration and contributions from the educational community, facilitating continuous improvement and innovation in plagiarism detection methodologies. As the project evolves, potential enhancements may include the integration of real-time feedback mechanisms, automated citation checking, and adaptive learning features to personalize the educational experience and empower students to cultivate their research and writing skills responsibly.

8. Limitations and Future enhancement

This chapter should contain major limitations of the project and the further enhancement of the project soon with a different but related approach.

8.1 Limitations

- Scalability: The current implementation may face challenges in scaling to handle large volumes of documents and classes with a high number of students. Scalability issues could arise in terms of processing speed, storage capacity, and system performance, particularly when analyzing complex or lengthy documents.

- Algorithm Sensitivity: The effectiveness of the plagiarism detection algorithms employed in the project may vary depending on factors such as document length, language complexity, and writing style diversity. The algorithms may struggle to accurately detect subtle instances of plagiarism or may produce false positives due to similarities in common phrases or terminology.

- User Interface Complexity: The user interface of the plagiarism detection system may be complex or unintuitive for some users, particularly educators who are not familiar with technical tools or algorithms. Improvements in user experience and interface design could enhance usability and adoption among a wider audience of educators.

- Plagiarism detection speed: While our plagiarism detection system effectively identifies similarities between documents, it operates at a relatively slow pace. There is room for improvement in optimizing the algorithms and techniques used to enhance the efficiency of the plagiarism detection process.

- Limitation with large files: The system may encounter difficulties when comparing large files, as it can significantly slow down the processing speed and compromise the accuracy of the results. Additionally, the cosine similarity technique may not be sufficient for accurately comparing large files, necessitating the exploration of alternative approaches.

- File format restrictions: Currently, the system only accepts PDF files for plagiarism detection, limiting its compatibility with other document formats. Expanding the system's capabilities to support a broader range of file formats would enhance its usability and applicability in diverse educational settings.

8.2 Future enhancements

- Machine Learning Integration: One potential approach for enhancing the project involves integrating machine learning techniques to improve the accuracy and robustness of plagiarism detection algorithms. By training models on annotated datasets of plagiarized and non-plagiarized documents, machine learning algorithms can learn to identify patterns and features indicative of plagiarism, leading to more precise detection results.

- Semantic Analysis: Incorporating semantic analysis techniques into the plagiarism detection process can enable a deeper understanding of document content and context. By analyzing the meaning and semantics of text, rather than relying solely on textual similarity measures, the system can better distinguish between genuine similarities and instances of paraphrasing or rephrasing.

- Extension for google classroom: Developing an extension for Google Classroom integration represents a significant opportunity for enhancing the accessibility and usability of the plagiarism detection system. The extension would enable seamless integration with Google Classroom, a widely used platform for online education, allowing educators to incorporate plagiarism detection seamlessly into their existing workflows.

- Implementation of local hash maps: To enhance efficiency and optimize resource utilization, the implementation of local hash maps could be explored. By utilizing local hash maps, the system can improve data retrieval and storage operations, leading to better performance and reduced processing time.

- Accept file formats other than .pdf: The system should accept all file formats like .docx, .odt and detect plagiarism if any.

References

- GeeksforGeeks. (n.d.). Measures of Distance in Data Mining. Retrieved from https://www.geeksforgeeks.org/measures-of-distance-in-data-mining/

- Potthast, M., Eiselt, A., Barrón-Cedeño, A., Stein, B., & Rosso, P. (2014). A Survey of Plagiarism Detection Methods. ACM Computing Surveys (CSUR), 47(2), 1-50. DOI: 10.1145/2596952

- Mili, A., Bouras, A., Mezghani, M., & Ben Yahia, S. (2019). Plagiarism Detection in Research Articles. IEEE Transactions on Information Forensics and Security, 14(5), 1133-1146. DOI: 10.1109/TIFS.2018.2878548

- Alzahrani, S., Salim, N., & Sulaiman, R. (2017). A Novel Approach for Plagiarism Detection Based on Semantic Similarity. Expert Systems with Applications, 79, 80-94. DOI: 10.1016/j.eswa.2017.04.014

- S. K. Pal, O. J. Raffik, R. Roy, V. B. Lalman, S. Srivastava and B. Sharma, "Automatic Plagiarism Detection Using Natural Language Processing," 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2023, pp. 218-222.

- Pallets Projects. (n.d.). Flask Documentation (Version 3.0.x). Retrieved from https://flask.palletsprojects.com/en/3.0.x/

- Google. (n.d.). Firebase Documentation. Retrieved from https://firebase.google.com/docs

- React. (n.d.). React Documentation. Retrieved from https://react.dev/