iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VI **Month of publication:** June 2023

DOI: https://doi.org/10.22214/ijraset.2023.54240

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

Similarity Detection in Two or More Submitted Documents

Piyush Nagpal[1], Mayank Gupta[2], Esha Singh[3], Ms. Sapna Gupta[4]
[1, 2, 3] *Student, ECE Department, Maharaja Agrasen Institute of Technology, New Delhi, India*
[4]*Asst. Professor, Maharaja Agrasen Institute of Technology, New Delhi, India*

Abstract: ***Educational Institutions & Organizations of the modern era are rapidly adopting the hybrid work culture, in the aftermath of the worldwide pandemic. Institutions & Organizations frequently require authentic, original content for a wide variety of purposes. Though students & employees strive towards that objective, plagiarized content often slips through scrutinization more often than the acceptable count. This paper aims to analyze and record the process & outcomes of building a web application that enables similarity detection between two or more files by uploading them on the application, or in other words, determine the similarity between the files. The web application will accept two or more files as input, and return similarity percentage which will be calculated using the Levenshtein distance Algorithm, as an automated report that will be sent to the user as an email. In our case, the web application will be majorly utilized to curb similarity in assignments. Along with this, the application will have a feature that allows any teacher to compare any two or more assignments submitted in a particular class, with just one click.***

Keywords: Similarity, Web Application, Algorithm, Levenshtein distance, Node.js, Longest Common Subsequence (LCS)

I. INTRODUCTION

In the context of textual content, similarity refers to the degree of resemblance or likeness between two or more pieces of text. It measures how closely related or comparable the content of the texts is, often based on various factors such as word choice, sentence structure, context, and overall meaning. Similarity can be assessed using different techniques and algorithms, depending on the specific task or application Thus, similar content cannot be used for ideal assessment of a person's expertise. As a direct result of increased similarity, problems are created not only for the organization, but also the original creators. Similarity in work becomes a hindrance for deserving candidates, along with hampering the overall productivity of a company & the authenticity of an organization's results. If not stopped, it can passively grow to degrade a country's human resources & corrode its core, thus jeopardizing the future. Educational institutions across the globe consider similarity in work as academic dishonesty. A perennial problem, similarity is a growing concern for not only school systems and teachers but also societies. As students we've heard our teachers complaining about how most of the students of our class have similar (and sometimes exactly same) matter in the assignments that they've submitted. To tackle this issue, we decided to build a similarity detector.

This brings us to our next question- Why is it wrong to have similarity in academic works?

A. Similarity is Unethical for Four Reasons

1) Firstly, it is unethical because it is a form of theft. By taking the ideas and words of others and pretending they are your own, you are stealing someone else's intellectual property.

2) Secondly, it is unethical because the one who plagiarizes subsequently benefits from this theft.

3) Thirdly, a degree is evidence of its holder's abilities and knowledge. If a student gains employment on the basis of a qualification they have not earned, they may be a risk to others.

No doubt some students do cheat. They deliberately take the results of other people's hard work, use it to gain credit for themselves, and learn little or nothing in the process. Students who plagiarize hurt themselves because they do not learn the process of academic achievement. It undermines academic and moral values. This is exactly why a tool like a similarity detector is the need of the hour, especially in todays 'internet dominated' world.

II. LITERATURE SURVEY

Similarity in work is taken very seriously in any academic institution, as well as a lot of organizations. Presence of similarity in an academic work makes the merit of the author questionable. Thus, strenuous efforts are put in to ensure that similarity in academics is kept to a minimum. Several online platforms already exist that are capable of detecting if a student has plagiarized content from a portal on the Internet or paraphrased the content to make it seem original.

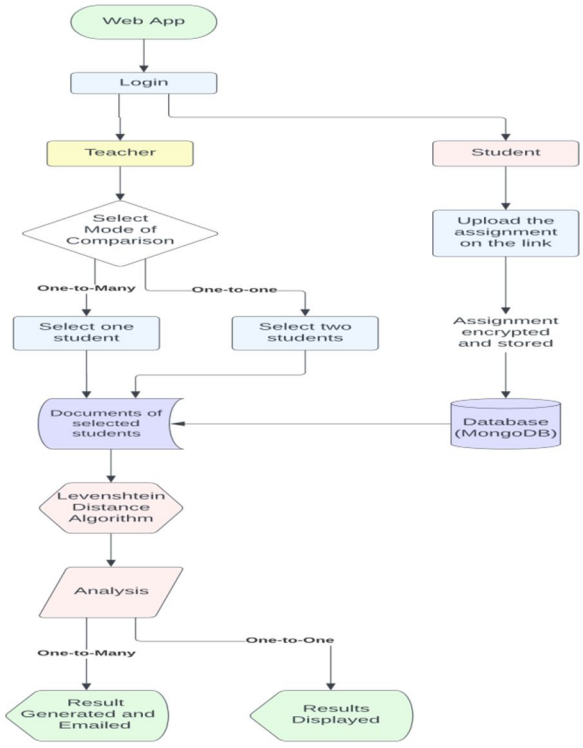
In fact, there have been attempts of research on textual similarity & similarity detection software, the performance of the software and their effects on students. In our context, similarity is effectively the same as textual plagiarism.

A.M.E.T Ali and his co-authors talk about the various types of plagiarism and the currently existing tools & techniques for detecting textual plagiarism, along with their limitations. [1]. Norman Meuschke and Bela Gipp surveyed the state of the art in detecting academic plagiarism, presented plagiarism detection systems, and summarized evaluations of their detection effectiveness. They outlined the limitations of text-based plagiarism detection methods and suggested that future research should focus on semantic analysis approaches that also include non-textual document features, such as academic citations [2]. Ahmed Hamza Osman, Naomie Salim, and Albaraa Abuobieda highlight the four-step process of textual plagiarism detection, along with some commonly-used techniques [3]. Due to the advancements in Artificial Intelligence, tools like ChatGPT also contribute to the rise in similarity, which requires a different angle to be solved [4]. Meanwhile, A. Parker and J. O. Hamblen explore the scope of computer algorithms in textual plagiarism detection for computer programs submitted by students as assignments, explaining the pre-processing of the programs in detail [5]. Shengnan Zhang, Yan Hu & Guangrong Bian worked on determining string similarity through an algorithm based on Levenshtein Distance, explored the possibilities of combining it with Longest Common Subsequence (LCS) and studied the results in terms of applicability and accuracy [6]. L. Bergroth, H. Hakonen and T. Raita compared various LCS Algorithms comprehensively, and studied their behaviour in various application environments, in terms of space demand, properties of the problem instances, and supporting data structures utilized [7]. D. Gunawan, C.A. Sembiring & M.A. Budiman conducted preliminary research to utilize cosine similarity and calculate the text relevance between an input document and the results generated on the internet, to select a topic-relevant document after thorough text preprocessing [8], which is crucial for any algorithm, as highlighted by Z. Ceska and C. Fox [9].

III. METHODOLOGY

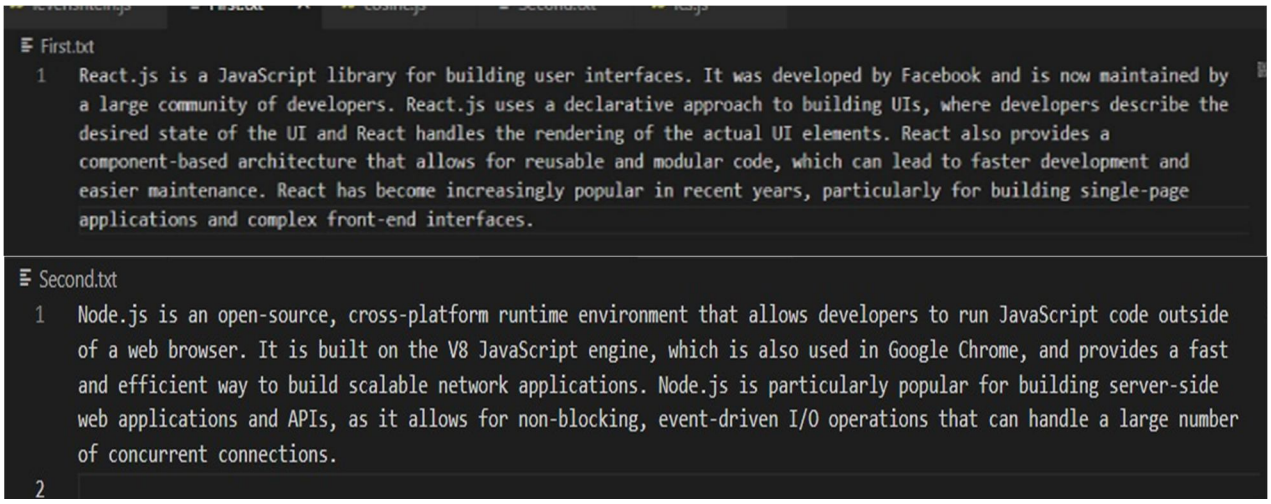
Our Objective is to build a clean, simple and efficient web application that lets a student upload their assignments in a protected form on a database, so that a teacher can access their own personal dashboard to compare the assignments of two or more students. Firstly, a working code to compare texts is needed, then file upload must be enabled, after which .pdf, .docx, .jpeg, .png file upload along with conversion will be required and lastly, dashboards for both students and teachers to upload and assess assignments will complete the application.

A. Ideation & Workflow Building

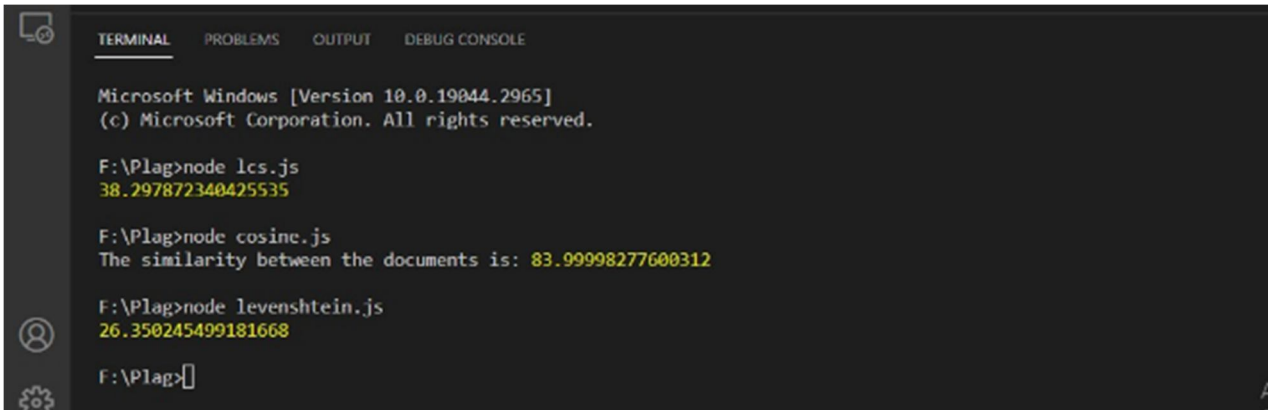


Flow of web app

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com



Sample texts used for algorithm comparison, along with results



Accuracy Result of Algorithm Comparison

B. Tech Stack and Tools

Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. For file uploading and file parsing, Multer, a Node.js middleware, the PDF-parse library (to overcome the buffer between simple text files and .pdf files), the Mammoth library (for .docx/MS-Word files), and lastly, node-pdf-ocr, along with Tesseract.js & GhostScript for Image Parsing and Optical Character Recognition was used. All these libraries were downloaded & installed through Node Package Manager (npm). For the development of the skeleton of the web application, Jade/Pug, CSS & Bootstrap Libraries were chosen, with GitHub for version control, MongoDB & MongoDB Compass for Database Management, Heroku for Web Hosting and MS Visual Studio Code as a development platform.

C. Algorithm Selection

The first task at hand was to choose the right algorithm for similarity check. For easier implementation, the goal of the algorithm was simplified to just check the amount of similarity between two strings. After researching on such algorithms, three algorithms were chosen for comparison, namely the *Levenshtein Distance Algorithm*, *Cosine Similarity Algorithm*, and the *Longest Common Subsequence (LCS) Algorithm*. After simulating a similarity check using all the three algorithms, it was discovered that the *Levenshtein Distance Algorithm* provided results with maximum accuracy, as shown in the screenshots below, in which two files, first.txt and second.txt were compared using the three algorithms, with the results displayed below. Hence, this algorithm was chosen for further development.

D. Development Cycle

After trying out various existing algorithms for string comparison, such as the *Cosine Similarity Algorithm*, *LCS algorithm* & *Levenshtein Algorithm*, the *Levenshtein Algorithm* was chosen due to its high accuracy and superior results. Thus, the next milestone was to build a working code that utilizes the Levenshtein Algorithm. It reads two .txt files, converts their data into strings and outputs the similarity between those two strings as a percentage. For this, the fs module of Node.js was utilized.

After the algorithm was created, our next objective was to take the input of two .txt files & store it locally as well as into the remote database (in our case, the MongoDB database). The Multer Middleware was put to use to handle file upload. It saves the .txt files locally as well as to the remote database. In the database, the files are not saved directly. Instead, the files are saved in a buffer format after compressing, in order to facilitate storage management and security. Text was extracted from the files, encoded in a base64 buffer and then stored in the database. After storage, entries were fetched from the database and algorithm was run on it successfully.

The focus was then shifted to .pdf files. There were a few problems encountered. The first problem was that the fs module that was being used for .txt files, doesn't work for .pdf files. To deal with this, the PDF-parse Library was added to accept PDF file uploads as inputs, since the library helps in extracting plaintext from PDF files. The next problem was that it was possible to save .pdf files in a buffer format but that would make their decryption non-feasible. Thus, there were two options to choose: one was to remove the concept of buffer files, which would not be the best approach to solve this problem. The second option was chosen, i.e., converting the .pdf files to .txt files at the backend, which would enable buffer files as well as their decryption.

After the program worked correctly for .pdf files as well, it was time to optimize the database. Initially, data was being stored for each student in a separate collection. This was resulting in reference errors. The user had to direct the program to the right collection to store data in. However, dynamically storing files in the right collection as per the logged in user was not possible. To overcome this problem, the schema was changed. A key was added to detect data related to a specific user and the data of all users was stored in a single collection.

Once a rudimentary system for .txt and .pdf files was established, to compare two files and produce the results accordingly, it was time to expand the project, to facilitate similarity check for a batch of files, in multiple formats. Since the core algorithm for comparing two files was already developed, creating a loop for a batch of files was a simple task. A custom window was created to display the results of the similarity check run for the entire batch of files, and the same is e-mailed at the address provided by the user, using the node-mailer library.

Furthermore, to increase the usability of the web application, the next objective was set to include more file formats. The same concept of converting files to .txt files, and then storing the data was applied. The mammoth library was installed and used to extract text from MS-Word formats and the data was then saved as a .txt file, which was read again and stored in a buffer format in the database. As for images, there were two possibilities to be considered: the images could either be digitally created, or the images could be scanned, both of which can have their text extracted through Optical Character Recognition. For this, the Tesseract library was deployed, which extracted text from native or scanned images, and the same process of creating a .txt file was followed.

Lastly, the .pdf files that contain scanned images required handling. Earlier, PDF-parse was utilized to detect text in .pdf files and extract it in form of .txt files. However, that was the case for native documents, which were digitally created. For scanned documents, OCR was required. The node-pdf-ocr library utilized Tesseract, along with GhostScript to do just that, and it was deployed.

IV. RESULT

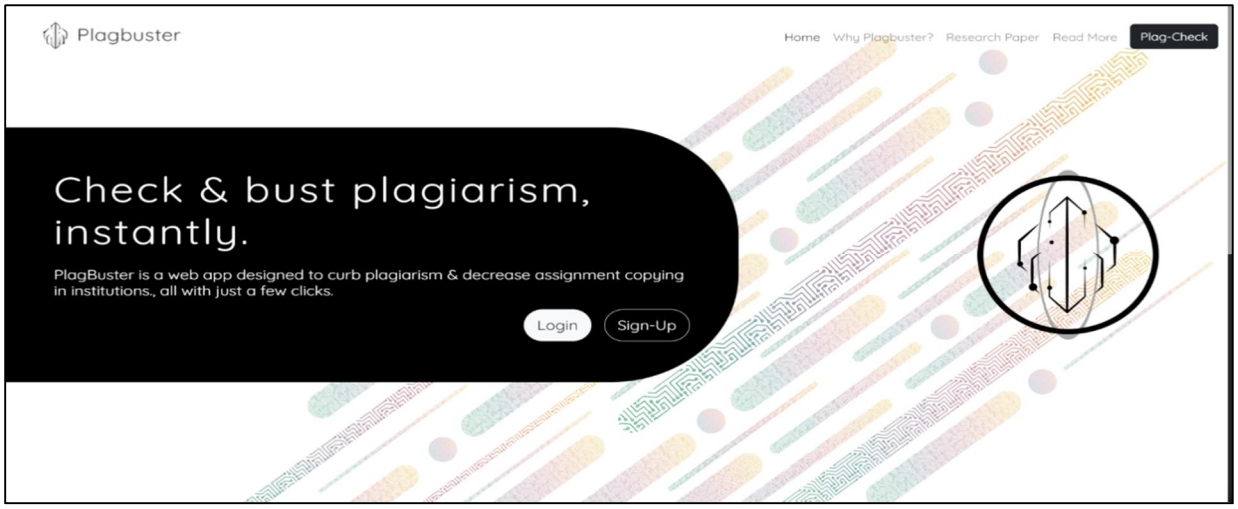
By utilizing the Levenshtein Distance algorithm, the prototype web application is capable of finding out the difference between the content of two documents by breaking it down into singular strings and generating an accurate percentage that represents the amount of similarity in the given two documents.

V. FUTURE SCOPE

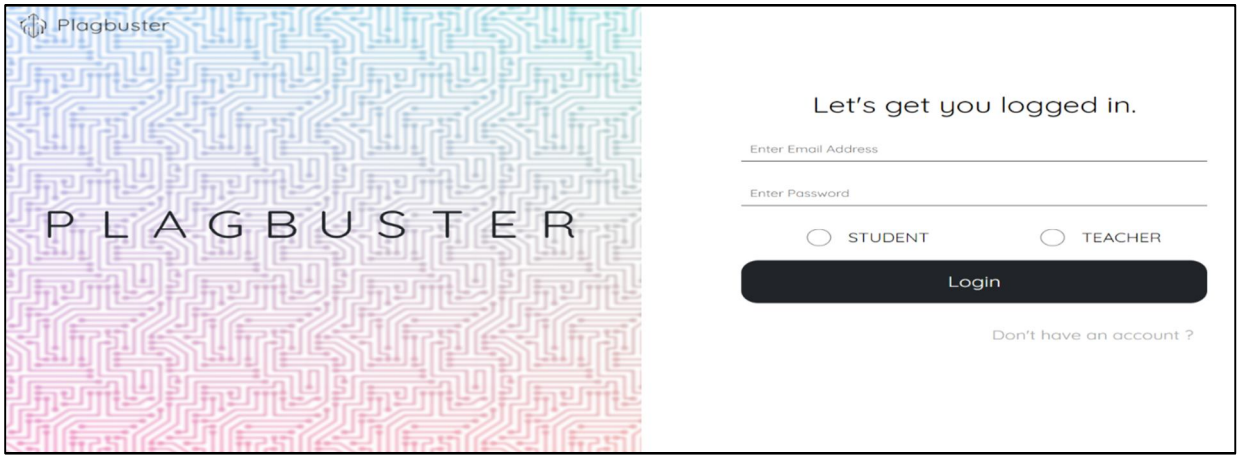
For future development of this project, support for multiple algorithms can be added. Also, along with file-to-file similarity checking mechanism, an algorithm that compares the given documents to the web can also be added. A downloadable report can be generated with analysis of their work, including information on the sources that the application identified as potentially copied and suggestions for how to properly attribute those sources. Along with this, a feature can be added that lets the application work across multiple devices. Customization options for modifying the sensitivity for similarity detection & improved user interface are along some additional features that can be worked on.

VI. CONCLUSION

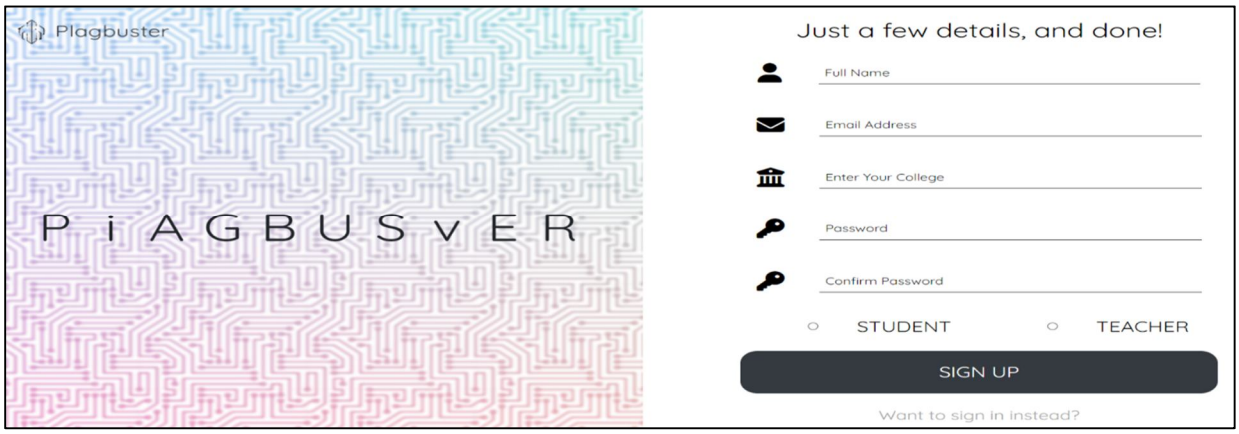
The web application can accurately predict the amount of similarity between two file uploads selected by the administrator. The following are some snapshots from the currently developed web application:



Home Page

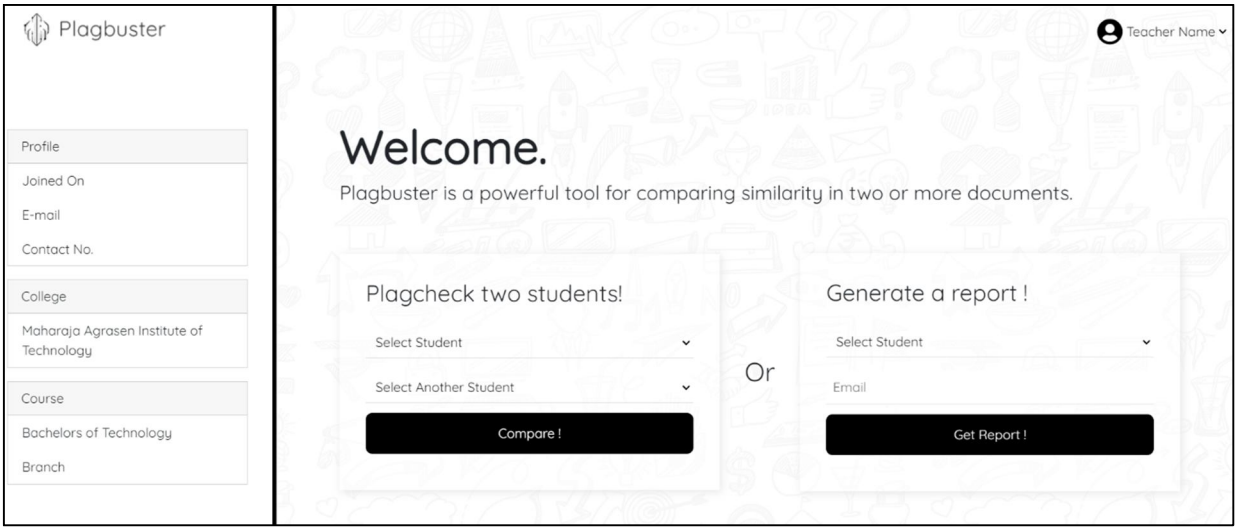


Sign-in Window, with encrypted password display.

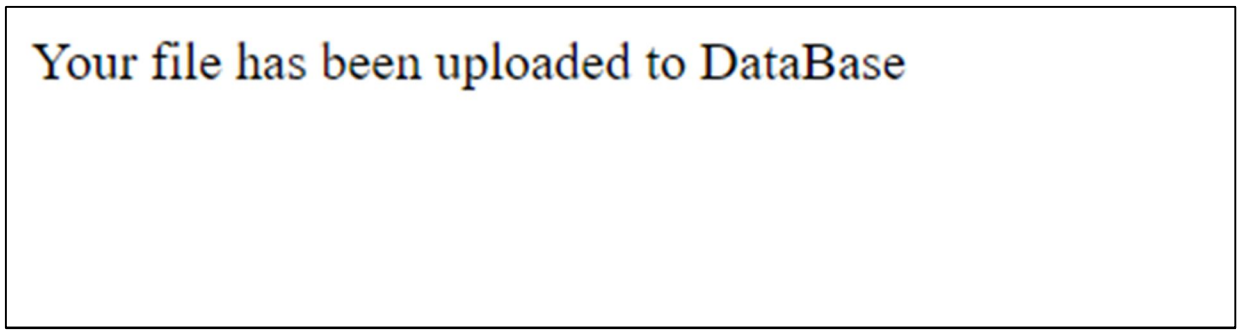


Registration Form for filling in basic details.

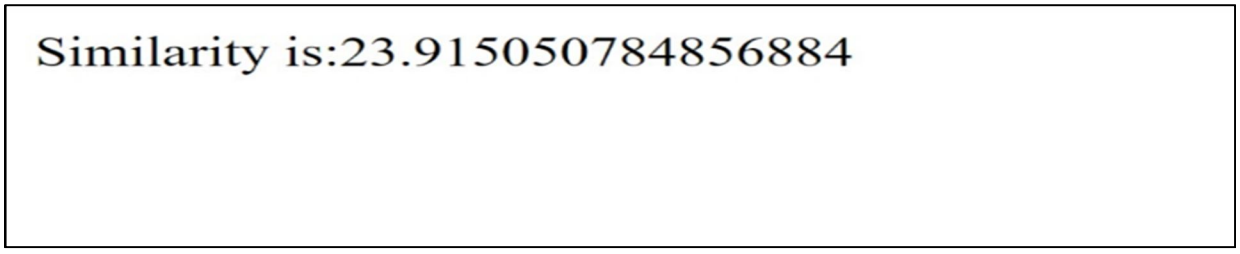
The Student Dashboard after logging in. Here, the student can upload their assignments.



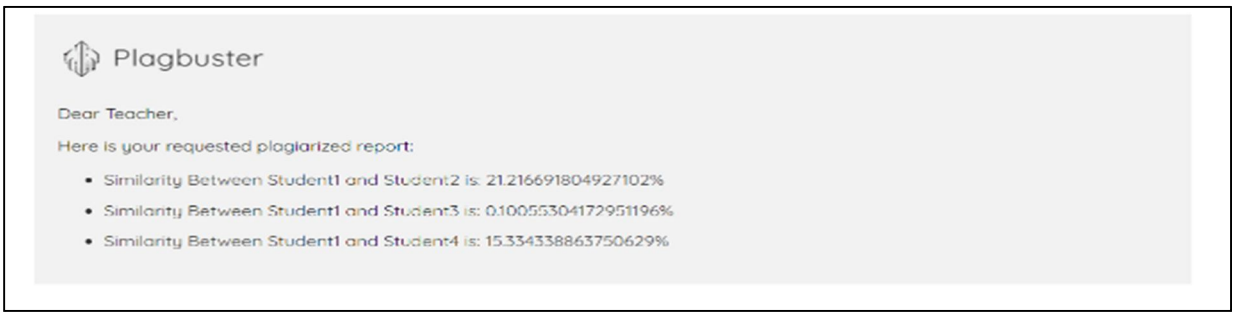
The Teacher Dashboard after logging in. Here, the teacher can choose two students whose assignments need to be compared.



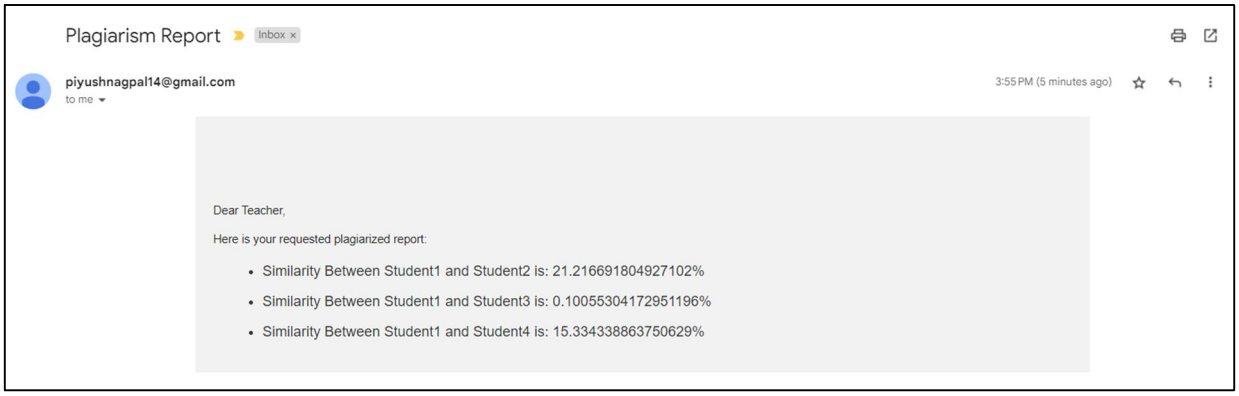
Prompt message for successful file upload.



Output shown as percentage after comparing the assignments of two students.

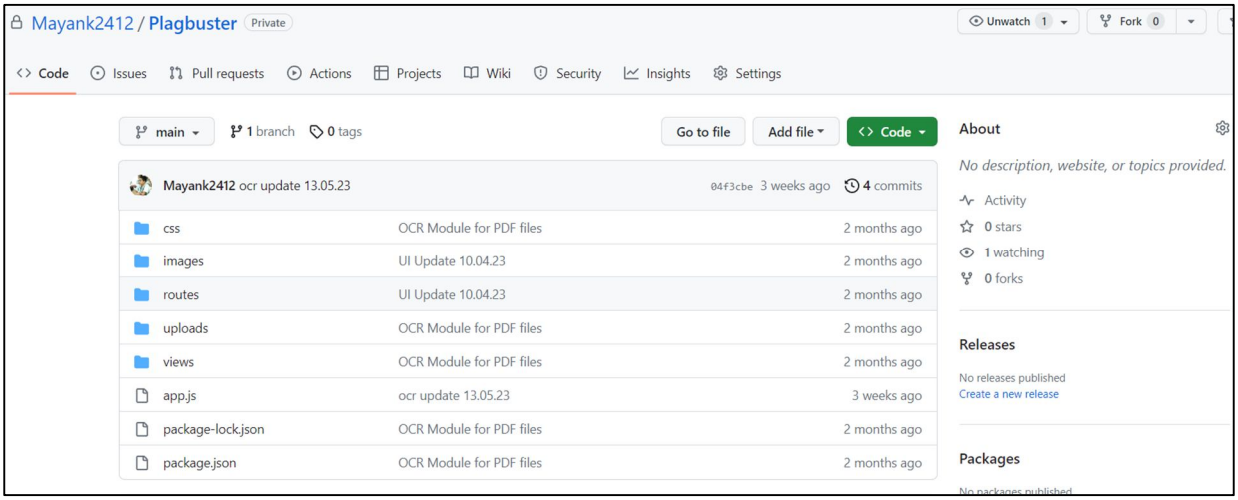


Output of similarity check for assignments of several students.

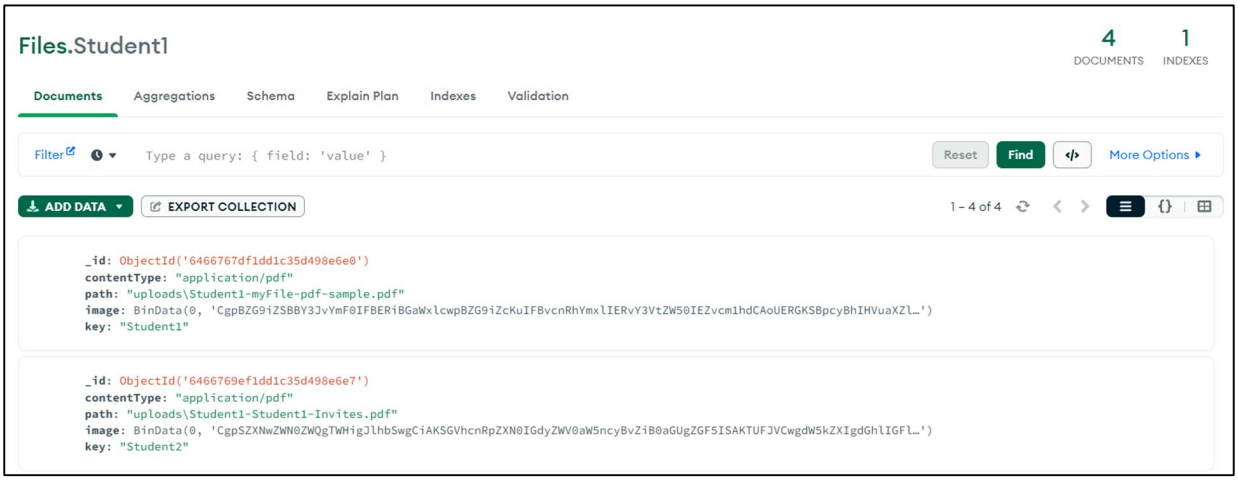


E-mail report for similarity check for assignments of several students.

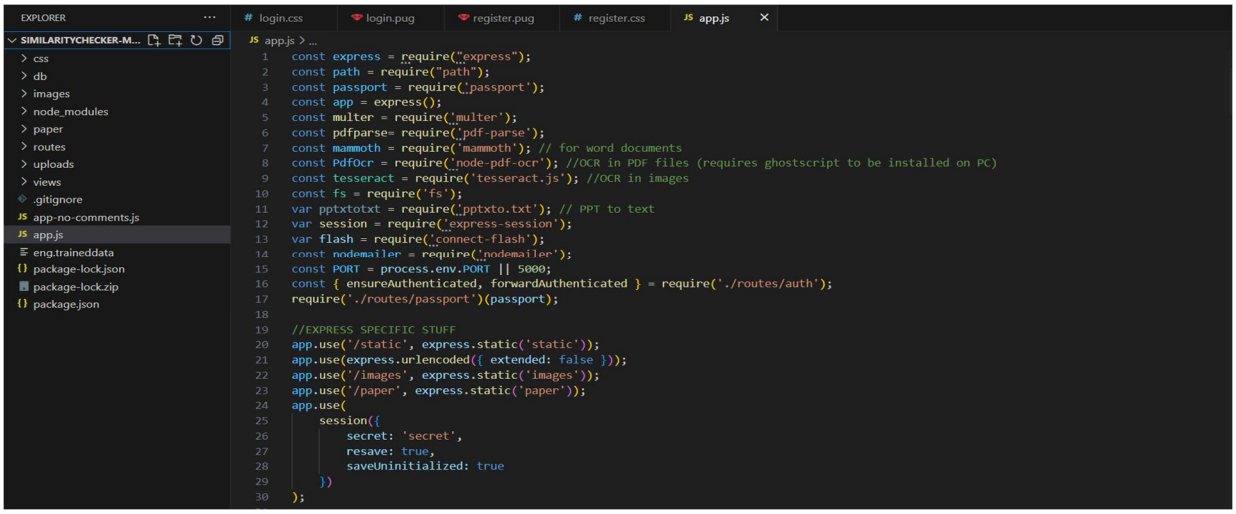
The collection in which the details of all accounts are stored. The passwords are encrypted and stored, for enhanced security.



The Github repository used for version control.



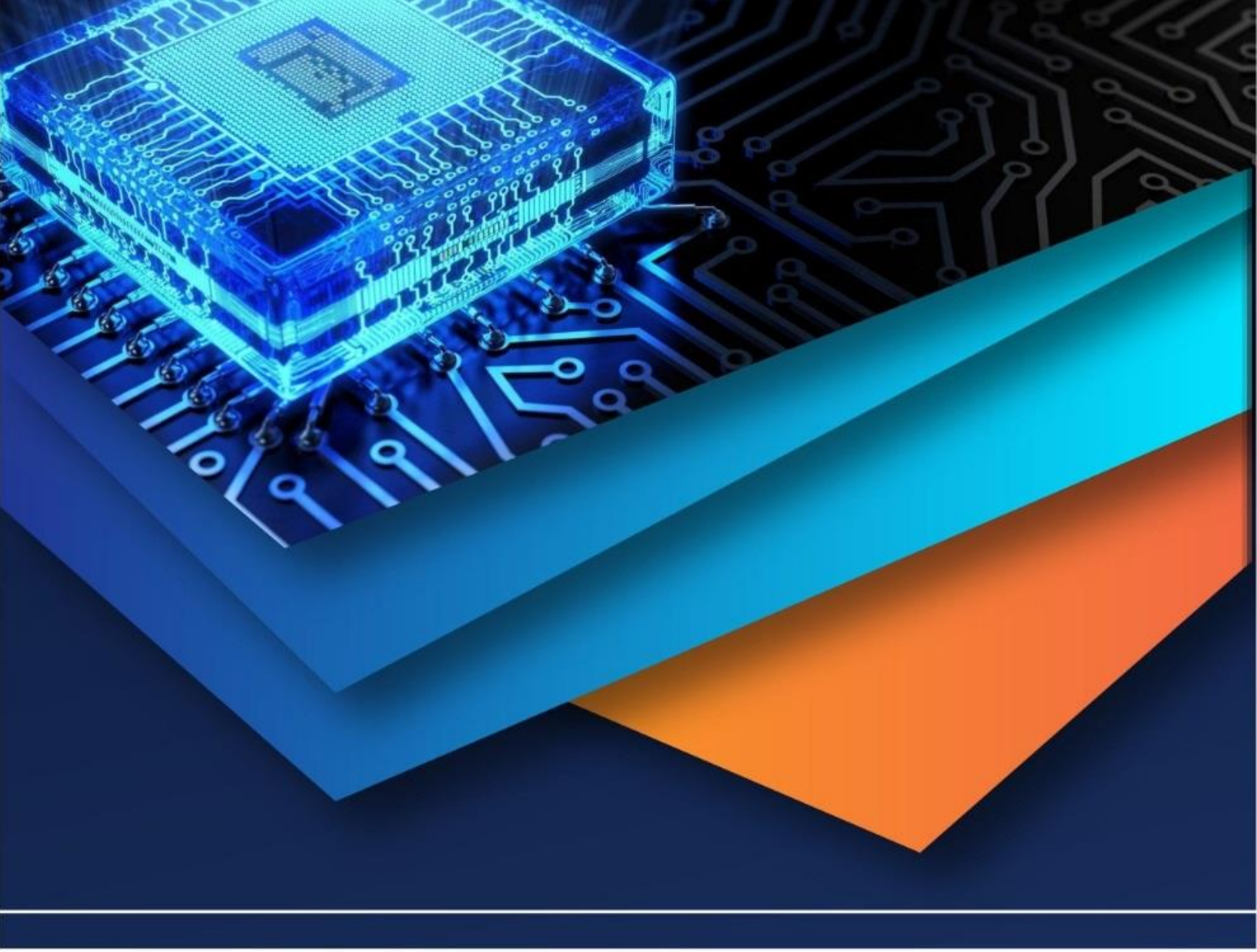
The collection in which the uploaded files of all students are stored.



The primary driver codes.

REFERENCES

[1] M. E. T. Ali, H. M. D. Abdulla, and V. Snasel, “Overview and comparison of plagiarism detection tools,” pp. 161–172, 2011.
[2] N. Meuschke and B. Gipp, “State-of-the-art in detecting academic plagiarism,” Int. J. Educ. Integr., vol. 9, no. 1, 2013.
[3] A. H. Osman, N. Salim, and A. Abuobieda, “Survey of text plagiarism detection,” Computer Engineering and Applications Journal, https://comengapp.unsri.ac.id/index.php/comengapp/article/view/5
[4] M. Khalil and E. Er, “Will CHATGPT get you caught? rethinking of plagiarism detection,” arXiv.org, doi: 10.48550/arXiv.2302.04335.
[5] A. Parker and J. O. Hamblen, “Computer algorithms for plagiarism detection,” IEEE Transactions on Education, doi: 10.1109/13.28038
[6] S. Zhang, Y. Hu and G. Bian, "Research on string similarity algorithm based on Levenshtein Distance," 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2017, pp. 2247-2251, doi: 10.1109/IAEAC.2017.8054419.
[7] L. Bergroth, H. Hakonen and T. Raita, "A survey of longest common subsequence algorithms," Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000, A Curuna, Spain, 2000, pp. 39-48, doi: 10.1109/SPIRE.2000.878178.
[8] Gunawan, D., Sembiring, C. A., & Budiman, M. A. (2018, March). The implementation of cosine similarity to calculate text relevance between two documents. In Journal of physics: conference series (Vol. 978, No. 1, p. 012120). IOP Publishing.
[9] Z. Ceska and C. Fox, “The influence of text pre-processing on plagiarism detection,” Research Repository, repository.essex.ac.uk/id/eprint/4019



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY