PLAGIARISM DETECTOR FOR LOCAL FILES USING FLASK

Project report submitted to Bharathiar University in partial fulfillment of the requirements for the award of the degree of Bachelor of Science in Computer Science.

Submitted by

ASHIQUE.A

Register No:2022K2635

Under the Guidance of

Mr.A.D.BHARATH M.Sc., B.Ed., M.Phil., (Ph.D)

Assistant Professor, Department of Computer Science.

DEPARTMENT OF COMPUTER SCIENCE

Government Arts and Science College,

Gudalur , The Nilgiris 643212

APRIL - 2023

CERTIFICATE

This is to certify that the bonafide record of project work entitled "**PLAGIARISM DETECTOR FOR LOCAL FILES USING FLASK**"submitted to Bharathiar University, in partial fulfillment of the requirements for the award of the degree in Bachelor of Science in Computer Science is the original record work done by **ASHIQUE.A(Reg No: 2022K2635)** under my supervision and guidance this project work has not formed basics for the award of any Degree/Diploma/Associate ship/Fellowship or similar title to any candidate of any university

Guide **Head of the department**

Submitted for Viva-Voce Examination held on _____

Internal Examiner **External Examiner**

DECLARATION

I **ASHIQUE.A,** hereby declare that the project, entitled "**PLAGIARISM DETECTOR FOR LOCAL FILES USING FLASK**" submitted to Bharathiar University, in partial fulfillment of the requirements for the award of the Degree of Bachelor of Sciencein Computer Science is a record of original work done by me under the supervision and guidance of **Mr. A.D. BHARATH M.Sc., B.Ed., M.Phil., (Ph.D)** Asst.prof. in Computer Science, Government Arts & Science College, Gudalur and it has not formed the basis for the award of any Degree/Diploma/Associate ship/Fellowship or similar title to any candidate of any university.

Date :17-04-2023

Place : Gudalur Signature of the Candidate

ACKNOWLEDGEMENT

The satisfaction and ecstasy that accompany the successful completion of a task would be incomplete without mentioning the people, who made it possible, whose constant assistance. guidance and encouragement crowded my efforts with success.

I wish to express my thanks to The Principal, **Dr.R.RAJENDRAN., MA., M.Phil., Ph.D.** Government Arts & Science College, Gudalur, for having given an opportunity for taking up the project

I am thankful to, Head the department of computers science **Dr. T. SURESHKUMAR MCA., MSc., M.Phil., PGDCA., D.T.Ed., Ph.D.** for being a source inspiration throughout the project

I take the opportunity to express my gratitude to, **Mr. A.D. BHARATH M.Sc., B.Ed., M.Phil., (Ph.D)** Asst.prof. in Computer Science, Government Arts & Science College, Gudalur for his valuable guidance, encouragement and keen interest in completion of my research work.

1 take this opportunity to offer my gratitude to My Family Members and Friends who made me face everything and enabled me to frame this fabulous work.

Thanking you

ASHIQUE.A

CONTENTS

SYNOPSIS

Plagiarism is a serious issue in the field of academia, where the unauthorized use of someone else's work can result in severe consequences. The project aims to develop a web- based application To address this problem, a Plagiarism Detection System for Local Files has been proposed. The Plagiarism Detection System for Local Files utilizes machine learning techniques and natural language processing (NLP) to detect potential cases of plagiarism in local files, such as text documents, reports, and research papers. The system employs NLP techniques, to analyze the text documents and identify patterns that may indicate plagiarism. The system will be accessible through a user-friendly web interface, allowing users to upload their local files for plagiarism detection. The results will be generated in real-time, providing immediate feedback on potential cases of plagiarism. The system has the potential to be a valuable tool for academic institutions, researchers, and writers to ensure the originality of their work and uphold academic integrity.

CHAPTER-1

INTRODUCTION

OVERVIEW OF THE PROJECT

This project is a file similarity detector that has been implemented using Flask, which is a popular Python web framework. The main goal of this project is to compare the similarity between two text files or a text file and a folder containing multiple text files. This can be useful in various scenarios, such as plagiarism detection, content similarity analysis, and document clustering. The project utilizes various natural language processing (NLP) libraries to preprocess the text data and calculate similarity scores.

The project consists of several modules that are responsible for different tasks. The preprocessing module is designed to clean and normalize the input text data. It converts the text to lowercase to ensure case-insensitive comparison, removes punctuation to eliminate irrelevant characters, normalizes spacing to standardize the text format, and filters out stop words. Stop words are common words that do not carry much meaning, such as "the", "and", "in", etc. By removing stop words, the preprocessing module aims to reduce noise in the text data and focus on more meaningful words.

The text extraction module is responsible for extracting relevant text information from the uploaded files. It reads the content of the files and extracts the text data, which is then passed to the preprocessing module for further cleaning and normalization. This module ensures that the input data is processed consistently and meaningfully, regardless of the format of the uploaded files, such as plain text files, Word documents, or PDF files.

The project also includes several routes in Flask for different pages, such as the home page, file input page, and folder input page, to provide a user-friendly interface for uploading and comparing files. The home page serves as the landing page of the application, providing an overview of the project and its features. The file input page allows users to upload two text files for similarity comparison, while the folder input page allows users to upload a folder containing multiple text files for batch comparison.

The similarity checking module is the core component of the project, responsible for calculating the similarity scores between the text documents. It uses TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, which is a widely used technique in NLP for text representation. TF-IDF represents the importance of each term in a document relative to a collection of documents. The higher the term frequency (TF) of a term in a document, and the lower the document frequency (DF) of the term in the entire collection, the higher the TF-IDF value of the term in the document. TF-IDF is commonly used in text similarity analysis as it captures the relative importance of terms in a document and is effective in handling both short and long documents.

The TfidfVectorizer class from the sklearn library is utilized to implement the TF-IDF vectorization. It converts the text data into a matrix of TF-IDF features, where each row represents a document and each column represents a term with its corresponding TF-IDF value. The module then calculates the cosine similarity between the TF-IDF feature vectors using the cosine_similarity() function from sklearn. Cosine similarity is a measure of similarity between two vectors that calculates the cosine of the angle between them. It ranges from -1 (completely dissimilar) to 1 (completely similar), with 0 indicating no similarity. The resulting cosine similarity scores are normalized to a percentage scale, representing the similarity between the text documents in percentage.

CHAPTER-2

SYSTEM STUDY

2.1 EXISTING SYSTEM

The existing market for plagiarism checking software primarily focuses on comparing files against internet sources. While there are many options available in this space, there are very few software solutions that support comparison between local files. The ones that do exist are often costly and inefficient when it comes to detecting plagiarism in local files. This creates a gap in the market, as there is a lack of specific software dedicated to comparing local files for plagiarism. This limitation can be particularly challenging for users who need to compare local files, such as academic institutions, content creators, and businesses that want to ensure originality and integrity in their documents. As a result, there is a need for more effective and affordable plagiarism checking solutions that cater specifically to local file comparison, providing accurate and reliable results for users in various industries.

2.1.1 DRAWBACKS

- Costly and inefficient solutions: The few software solutions that do support local file comparison are often costly and may not provide efficient and accurate results
- Limited file format support: Existing systems have limitations for supported file formats
- Existing systems primarily focus on comparing files against internet sources
- Lack of specific software dedicated to local file comparison

2.2 PROPOSED SYSTEM

The proposed system for local file comparison is a comprehensive and advanced solution that aims to address the challenges associated with comparing and detecting similarities between local files. This system is designed to provide accurate and reliable results while offering a user-friendly interface, and scalability to cater to the needs of various industries.To ensure accurate comparison, the proposed system incorporates advanced text processing capabilities. This includes normalization, stopword removal, and stemming, which standardize the text and eliminate irrelevant words that may affect the similarity results. The system also utilizes regular expression modules for efficient text processing, ensuring that the extracted text contents from uploaded files are cleaned and processed effectively.The user interface of the proposed system is designed to be user-friendly and intuitive. It is developed using the Flask library, which provides a seamless and interactive interface for users to upload, compare, and analyze files. The interface offers clear instructions and feedback to guide users through the file comparison process, making it easy for both technical and non-technical users to utilize the system effectively. Its ability to support various file formats and provide accurate and reliable results makes it a valuable tool for industries that require robust file comparison capabilities. With its innovative features and capabilities, the proposed system has the potential to revolutionize the way local files are compared and similarities are detected, providing significant advantages over existing systems.

2.2.1 ADVANTAGES

- Supports various file formats for greater flexibility
- Free of cost for all users
- Advanced text processing capabilities for accurate plagiarism detection
- Scalable and performant, even with large volumes of files
- User-friendly interface for ease of use
- Suitable for users in various industries due to its advanced, reliable, and user-friendly nature.

2.3 SYSTEM SPECIFICATION

2.3.1 HARDWARE SPECIFICATION

- INTEL i3 OR ABOVE
- 15 GB OF HARD DISK CAPACITY OR ABOVE
- 4 GB OF INTERNAL MEMORY CAPACITY (RAM)
- 1.2 GHz CPU SPEED

2.3.2 SOFTWARE SPECIFICATION

- WINDOWS 7 OPERATING SYSTEM OR ABOVE
- PYTHON
- VS CODE(IDE)
- MODULES REQUIRED:
    - SCIKIT-LEARN
    - FLASK
    - OS
    - DOCX2TXT
    - PYPDF2
    - RE
    - TEMPFILE

2.3.3 SOFTWARE FEATURES

The system is developed using Python, which is a popular high-level programming language that has a number of features that make it useful for a wide range of applications. Python has a simple and easy-to-learn syntax, which makes it a great language for beginners and experts alike. Python supports object-oriented programming, which allows developers to write reusable and maintainable code.

Python comes with a large standard library that provides many useful modules for tasks such as file I/O, networking, and regular expressions. Python has a large number of third-party libraries available, which can be easily installed using package managers like pip. These libraries provide additional functionality for tasks such as data analysis, scientific computing, and machine learning. Python code can be run on a wide range of platforms, including Windows, macOS, Linux, and Unix. Python is an interpreted language, which means that code can be executed directly without the need for compilation.

The system is also developed using Vs-Code IDE, which is a free, open-source code editor developed by Microsoft. It supports multiple programming languages, offers intelligent code completion, debugging capabilities, and an integrated terminal. Its extensibility through a large marketplace of extensions, seamless integration with other development tools, and user-friendly interface make it a popular choice among developers. With features like Live Share for collaboration, a vibrant community, and continuous updates, VS Code is a powerful and versatile code editor for developers of all levels.

The system uses Flask which is a popular Python web framework that allows developers to build web applications quickly and efficiently. Developed as a micro-framework, Flask provides only the essentials for web development, allowing developers to have more control over their application's architecture and design. Flask follows the WSGI (Web Server Gateway Interface) standard, making it compatible with a wide range of web servers. It provides features such as routing, templating, and request handling, while allowing developers to choose and integrate additional libraries as needed. Flask's simplicity, flexibility, and extensive documentation make it a favorite choice for building small to medium-sized web applications in Python.

Another module of the system is Scikit-learn (sklearn) is a popular Python library for machine learning, offering a wide range of algorithms, tools for data preprocessing,

model evaluation, and hyperparameter tuning. It is known for its simplicity, ease of use, and extensive documentation. With a large community of users and contributors, scikit-learn is widely used in academia and industry for various machine learning tasks. Its efficient implementations and consistent APIs make it a go-to choice for developers looking to build machine learning models in Python.

OVERVIEW OF PYTHON

Python is a popular high-level programming language that was first released in 1991 by Guido van Rossum. Python is known for its simple and easy-to-learn syntax, as well as its focus on code readability and maintainability. It is widely used for a variety of applications, including web development, scientific computing, data analysis, machine learning, and artificial intelligence.

One of the key features of Python is its large standard library, which provides many useful modules for tasks such as file I/O, networking, and regular expressions. Python also has a large number of third-party libraries available, which can be easily installed using package managers like pip. These libraries provide additional functionality for tasks such as data analysis, scientific computing, and machine learning. Python is an interpreted language, which means that code can be executed directly without the need for compilation. It is also a dynamically typed language, which means that data types are determined at runtime rather than at compile time. Python has automatic memory management, which means that memory is allocated and deallocated automatically.

Overall, Python is a versatile and popular programming language that is well-suited for a wide range of applications. Its simple syntax, large standard library, and extensive third-party libraries make it a popular choice for developers.

VS CODE

VS Code, is a popular source code editor developed by Microsoft. It is free, open-source, and available for Windows, macOS, and Linux operating systems. VS Code has gained immense popularity among developers due to its robust features, extensibility, and user-friendly interface.

One of the standout features of VS Code is its extensive support for various programming languages, including but not limited to JavaScript, Python, C++, C#, Go, and Ruby, making it suitable for a wide range of developers. It comes with built-in IntelliSense, which provides intelligent code completion suggestions, code navigation, and error checking, helping developers write code more efficiently and with fewer errors.

In conclusion, Visual Studio Code is a powerful, extensible, and user-friendly source code editor that has gained widespread popularity among developers. Its rich features, extensive language support, and customization options make it a top choice for many developers and teams around the world. Whether you are a beginner or an experienced developer, VS Code is worth considering for your coding needs

FLASK

In this project development,Flask is used fto build web applications feed . Flask is a popular Python web framework that allows developers to build web applications quickly and efficiently. It is a micro-framework, which means it provides only the essential tools for building web applications, making it lightweight and flexible.

Flask offers features such as routing, templating, and handling HTTP requests and responses. It follows the WSGI (Web Server Gateway Interface) standard, allowing it to integrate easily with web servers like Gunicorn, uWSGI, or the built-in Flask development server.Flask also offers a built-in Jinja2 templating engine, which allows developers to create dynamic HTML templates for rendering views. This makes it easy to separate the logic and presentation aspects of a web application, following the Model-View-Controller (MVC) pattern.

In conclusion, Flask is a powerful and flexible Python web framework that makes it easy for developers to build web applications. Its simplicity, extensive documentation, and active community make it a popular choice for building web applications of all sizes, from small prototypes to large-scale production applications.

SCIKIT-LEARN

In this project, Scikit-learn (sklearn) is used to implement NLP functionalities such as TD-IDF and Cosine similarity . Scikit-learn is a popular and widely-used Python library for machine learning. It provides a comprehensive suite of tools for tasks such as classification, regression, clustering, and dimensionality reduction. Scikit-learn is built on top of other popular Python libraries, including NumPy, SciPy, and matplotlib, and offers a user-friendly API for training and evaluating machine learning models.

Scikit-learn provides built-in classes for performing TF-IDF vectorization and cosine similarity calculations. The 'TfidfVectorizer' class in scikit-learn allows you to perform TF-IDF vectorization on text documents, converting them into numerical feature vectors that can be used as input for machine learning models. The 'cosine_similarity' function in scikit-learn's 'metrics' module allows you to calculate the cosine similarity between pairs of document vectors.Together, TF-IDF and cosine similarity are powerful techniques for text analysis tasks such as document clustering, document retrieval, and text recommendation. They are widely used in natural language processing (NLP) and information retrieval applications, and scikit-learn provides convenient implementations of these techniques for developers to use in their machine learning projects.

In conclusion, scikit-learn is a powerful and user-friendly machine learning library in Python. Its extensive collection of algorithms, tools for data preprocessing and model evaluation, simplicity, and strong community support make it a top choice for machine learning tasks. Whether you are a beginner or an experienced practitioner, scikit-learn provides a solid foundation for developing machine learning models in Python.

CHAPTER-3

SYSTEM DESIGN AND DEVELOPMENT

3.1 INPUT DESIGN

Input design is one of the most important phases of the system design. The input design of the project is crucial for the user interface, as it determines how users will provide input for the comparison process. The intuitive, user-friendly, and efficient, making it easy for users to select and upload files, specify comparison options, and configure the comparison process.

The file inputs are taken using Python library called Flask . Flask provides the tools and functionality to take file inputs,it is designed to handle different file formats, including .pdf, .docx, .doc, and .txt, allowing users to upload a wide range of text documents for comparison. The files are uploaded using the file upload buttons, which are easy to locate and use.The input design also includes error handling mechanisms to validate and verify the uploaded files, such as checking for correct file formats and handling cases where errors occur during the file upload process. This ensures that users receive accurate and reliable results.

The objectives considered during input design are:

- The ability for users to upload file.
- Ensuring that supports multiple file formats.
- Ensuring compatibility and responsiveness across different devices.

Overall, the objectives of the input design phase are to provides a user-friendly and efficient way for users to upload their files for similarity comparison. that supports different file formats, and utilizes powerful text analysis techniques for accurate results.

3.2 OUTPUT DESIGN

The output design of this project includes the similarity scores between the query file and the files in the folder, presented in a visually appealing and informative manner to users. The project generates a result page that displays the similarity scores in a clear and organized format for easy interpretation.

The main output is a similarity score, usually expressed as a percentage, which indicates the degree of similarity between the documents. This score is calculated using the cosine similarity metric, which measures the cosine of the angle between two vectors representing the documents' content. A higher similarity score indicates a higher degree of content similarity, while a lower score indicates less similarity.

The objectives considered during output design are:

- The primary objective is to accurately calculate the similarity .
- Ensure it is efficient in terms of processing time .
- Ensure it is in easily understandable format.

Overall, the output of this project provides users with a clear and organized presentation of the similarity scores, allowing them to easily understand and interpret the results of the similarity comparison. It is designed to be visually appealing, user-friendly, and informative, facilitating a seamless and efficient user experience.

3.3 SYSTEM DEVELOPMENT

3.3.1 DESCRIPTION OF MODULES

- Text Extraction module
- Text preprocessing module
- Similarity detecting module
- User interface module

Text Extraction Module

The User interface moduleext extraction module in this project is a crucial component that enables the extraction of text from different file formats. It is implemented as a Python function that takes a file name as input and uses various modules to extract text from different file types, including Word documents, PDFs, and plain text files.The module first checks the file extension to determine the file format. If the file is a Word document with a ".docx" extension, the docx2txt module is used to process the document and extract the text. If the file is a PDF with a ".pdf" extension, the PyPDF2 module is used to read the PDF and extract the text from each page. For plain text files with ".doc" or ".txt" extensions, the module reads the contents of the file using standard Python file reading operations.

The extracted text is then stored in a variable and returned as the output of the function.Themodule also handles unsupported file formats by returning an appropriate error message.The text extraction module is a versatile and essential tool in this project, allowing for the processing of different file types and extracting text for further analysis or processing. It enhances the project's functionality by enabling text extraction from a wide range of file formats.

Text Preprocessing Module

Text Preprocessing module automates several essential text preprocessing steps. It takes input text and performs a series of operations to clean and transform the text into a format that is suitable for text analysis or natural language processing tasksIt starts by converting the input text to lowercase ensuring that the text is case consistent and eliminating potential discrepancies due to letter casing. It then removes punctuation marks which helps to eliminate unnecessary noise from the text data. The function also

normalizes whitespace by replacing multiple consecutive spaces with a single space using the regular expression pattern \s+, standardizing spacing in the text. Additionally, it removes stop words, such as "the", "and", "is", etc., which are commonly considered less informative for text analysis.

Overall, the Text preprocessing module provides a comprehensive text preprocessing solution for cleaning and transforming text data for various text analysis or natural language processing tasks.

Similarity Detecting module

The Similarity Detecting module designed using Python Libraries called scikit-learn and Flask framework. It serves as a file similarity detector, allowing for comparison of text content between an uploaded files The function follows a series of steps to achieve this comparison.

Firstly, the uploaded file is received from the request object and saved to a temporary file path. The text content of the file is then read and preprocessed using a function called "preprocess_text()". Then, for each file in the folder, cosine similarity is calculated using TF-IDF. The "TfidfVectorizer" class from the "sklearn.feature_extraction.text" module is used to convert the text documents into numerical vectors based on their TF-IDF scores. The query file and the current file in the loop are transformed into TF-IDF vectors using the vectorizer, and the cosine similarity between these two vectors is calculated using the "cosine_similarity" function from the "sklearn.metrics.pairwise" module.

The TF-IDF and cosine similarity are used to measure the similarity between the files based on the frequency and rarity of terms in the documents. Higher cosine similarity scores indicate higher similarity between the documents, while lower scores indicate lower similarity. The advantage of using TF-IDF and cosine similarity in this project is that they provide a robust measure of similarity that accounts for both the term frequency and the rarity of terms, making it suitable for comparing text documents of varying lengths and content.

Finally, the function renders a templates with the similarity dictionary as a context variable, which can be displayed in the resulting webpage to present the similarity scores to the user.

In summary, this mdule is a crucial component of this web application that enables comparison of text content between an uploaded file and a list of folder files using cosine similarity. It involves preprocessing of text, vectorization of text content, calculation of similarity scores, temporary file handling, and template rendering for presenting the results to the user.

User Interface module

The user interface of the project is designed using Flask, which is a popular Python web framework. Flask provides a flexible and efficient way to create web applications with Python, including building user interfaces.

The user interface of this project is designed with usability and simplicity in mind. It provides an intuitive and straightforward way for users to upload their query file and a folder of files to compare for similarity. The interface offers clear and easy-to-use file upload buttons that allow users to select and upload their files with just a few clicks.

Once the files are uploaded, the interface provides feedback to the users by displaying the similarity scores as a result. The similarity scores are calculated using cosine similarity and Td-Idf vectorization techniques, which are powerful methods for measuring text similarity. The result is presented in a visually appealing way which displays the similarity scores in a tabular format, making it easy for users to interpret the scores and understand the level of similarity between the files.

Furthermore, the interface is designed to be responsive and compatible with different devices and screen sizes, making it accessible to users across different platforms, including desktop computers, tablets, and mobile devices. This ensures that users can access and use the project's functionality seamlessly, regardless of the device they are using.

In summary, the user interface of this project is user-friendly, providing a smooth and efficient experience for users to upload and compare files for similarity. It offers clear feedback in the form of similarity scores and presents the results in a visually appealing manner, making it easy for users to interpret the results. The interface also includes error handling mechanisms and is responsive, ensuring accessibility across different devices.

CHAPTER-4

SYSTEM TESTING AND IMPLEMENTATION

SYSTEM TESTING

Testing is an important aspect of this project as it helps to ensure that the developed system is functioning properly and meeting the desired requirements. Testing helps to identify and correct errors and bugs in the system, which can improve the overall performance and user experience. It ensures that the software system meets the requirements, is free of bugs, and performs as expected. In the case of the real-time plagiarism detector project using Python, Scikit-learn, and Falsk.

Software testing is a philosophy that emphasizes the importance of identifying and correcting defects in software products. It is a proactive approach that involves continuous improvement and refinement of the software development process. The goal of software testing is to ensure that the software meets the requirements and specifications of the users, and that it is reliable, secure, and user-friendly. Testing is not a one-time event, but rather an ongoing process that begins with the planning phase and continues throughout the entire software development lifecycle. It requires a disciplined and structured approach that involves careful analysis, design, execution, and reporting of test results. It is a collaborative effort that involves developers, testers, and other stakeholders working together to deliver high-quality software products that meet the needs of users. Ultimately, software testing is about delivering value to the end-users by ensuring that the software meets their needs and expectations.

TYPES OF TESTING

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing
- Performance Testing

UNIT TESTING

This involves testing each module or component of the system individually to ensure that it performs as expected. For example, the text extraction module can be tested by providing text files with different extentions (.pdf,.txt,.doc,.docx) and checking the contents of file if it is extracted properly.

INTEGRATION TESTING

This involves testing the integration of different modules to ensure that they work together correctly. For example, the integration of the text extraction and preprocessing modules can be tested by passing the contents to preprocessing module which is extracted by text extraction module and checking if the contents are processed correctly.

SYSTEM TESTING

This involves testing the entire system to ensure that it meets the requirements and performs as expected. For example, the entire system can be tested by providing different different types of text files and checking if the results are obtained

PERFORMANCE TESTING

Performance testing is a type of testing that is done to determine the performance of the system under different loads and stress conditions. To conduct performance testing, different scenarios are created to simulate various situations, and the system's performance is evaluated against each of them. The testing can be carried out using a large number of samples and under different conditions to determine the reliability and performance of the system. Performance testing can help identify any bottlenecks or issues with the system's performance and allow for optimizations to be made to improve the system's overall performance.

ACCEPTANCE TESTING

This involves testing the system with the end-users to ensure that it meets their requirements and is user-friendly. For example, the user interface module can be tested with a group of users to ensure that they can easily use the application to find plagiarism between their text files.

IMPLEMENTATION

The implementation phase of a project involves the development of various modules using different software and hardware tools, as identified in the earlier phases. These modules are designed to perform specific tasks and work together to create a functional system. Let's take a closer look at the overview of the modules developed during the implementation phase.

One of the key modules developed during the implementation phase is the Text Extraction module. This module is responsible for extracting text contents from uploaded files, such as PDF and Word documents. It utilizes libraries such as PyPDF2 and Docx2txt to parse the files and extract the text contents. PyPDF2 provides functionalities to extract text from PDF documents, while Docx2txt is used to extract text from Word documents. This module is crucial as it forms the foundation for the subsequent processing of text data in the system.

Another important module developed during the implementation phase is the Preprocessing module. This module is responsible for processing the extracted text by removing unnecessary elements such as stopwords, blank spaces, and special symbols. Regular expressions, a powerful tool for pattern matching and manipulation, are commonly used in this module to identify and remove these elements from the text data. The Preprocessing module ensures that the text data is cleaned and prepared for further processing, such as feature extraction and similarity checking.

The User Interface module is another significant module developed during the implementation phase. This module is responsible for creating a user-friendly interface for the web application. It is developed using libraries such as Flask, which is a popular web framework in Python. The User Interface module allows users to interact with the system, upload files for processing, and initiate similarity checking. A well-designed user interface is crucial for providing a seamless and intuitive user experience, making the system accessible and easy to use for end-users.

The heart of the application is the Similarity Checking module, which is also developed during the implementation phase. This module is responsible for comparing the extracted text contents of files with each other to check for similarity. It utilizes machine learning techniques provided by the Scikit-Learn library to perform similarity checking. Scikit-

Learn is a widely used machine learning library in Python, providing a wide range of algorithms for text processing, including text similarity measures. The Similarity Checking module employs appropriate algorithms and techniques to calculate the percentage of similarity between files, which is a critical feature of the system.

Once all the modules are developed, the system undergoes rigorous testing to ensure that it meets all the requirements and specifications defined in the earlier phases. Various test cases are executed to identify and fix any bugs or errors in the system. Testing is a critical step in the implementation phase as it helps to ensure that the system functions as intended and delivers accurate results. Once the system is thoroughly tested and validated, it is made available for users to access and utilize for their needs.

In conclusion, the implementation phase of a project involves the development of various modules using different software and hardware tools. These modules include the Text Extraction module, Preprocessing module, User Interface module, and Similarity Checking module, among others. These modules work together to create a functional system that is capable of extracting text from files, preprocessing the text data, providing a user-friendly interface, and performing similarity checking using machine learning techniques. Rigorous testing is conducted to ensure that the system meets all the requirements and specifications, and once validated, the system is made available for users to access and utilize for their needs. The successful implementation of these modules is crucial for the overall success of the project and the achievement of its objectives.

CHAPTER-5
CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION

In conclusion, the development of a "**Plagiarism detector for local files using Flask"** For local file comparison has addresses the limitations of existing plagiarism checking solutions by offering advanced text processing capabilities, support for various file formats, sophisticated similarity metrics, scalability, user-friendliness, enhanced security features, and comprehensive customer support. This makes it a reliable and efficient option for users in various industries who require local file comparison for plagiarism detection. With its unique features and advantages, the proposed system offers an improved and more effective solution for ensuring originality and integrity in documents, providing a valuable tool for plagiarism detection and prevention.

However, the system has some limitations that need to be addresse such as i)Computational Resources,The project's performance may depend on the availability of computational resources, such as processing power and memory, which may limit its scalability or efficiency for large-scale text comparison tasks.

ii)Language and Domain Specificity: The project's accuracy and effectiveness may vary depending on the language and domain of the text documents being compared. It may perform better on certain languages or domains compared to others, and may not be as effective in detecting similarity in highly technical or specialized domains.

FUTURE ENHANCEMENT

There are several potential areas for future enhancement of this project, including:

- Multilingual support: Expanding the project to support multiple languages would greatly enhance its applicability and usefulness in diverse academic contexts. This could involve training the model on a larger and more diverse corpus of texts in different languages, enabling it to detect similarities and overlaps in documents written in languages other than English.
- Domain-specific customization: Allowing users to customize the project for specific domains, such as scientific literature, medical texts, or legal documents, could improve its accuracy and relevance for specific academic

disciplines. This could involve fine-tuning the model with domain-specific data and incorporating specialized knowledge or vocabulary related to specific fields.

- Enhanced sensitivity analysis: Incorporating more advanced sensitivity analysis techniques to better identify paraphrasing and rephrasing of texts could enhance the project's ability to detect similarities and overlaps in documents..
- Enhanced user interface and visualization: Improving the user interface and visualization capabilities of the project could make it more user-friendly and accessible to a wider range of users, including students, researchers, and instructors

Overall, this project has the potential to greatly improve the communication and accessibility for the deaf and hard-of-hearing community, and future enhancements could further improve its functionality and impact.

BIBLIOGRAPHY

BOOKS REFERRED

1. Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc.
2. Natural Language Processing with Pythonby Steven Bird, Ewan Klein, and Edward Loper. O'Reilly Media, Inc.

PAPER REFERRED

1. Li Junfei. (2019). Research on the Application of Natural Language Processing Toolkit in College English Teaching. Education Modernization 92, 136-137.
2. S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation, 9(8):1735–1780, 1997.
3. Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. Nature, 521(7553):436–444, 2015.

WEBSITES REFERRED

1. https://scikit-learn.org
2. https://flask.palletsprojects.com/en
3. http://stackoverflow.com
4. https://pypi.org
5. https://github.com

APPENDICES

A. DATAFLOW DIAGRAM

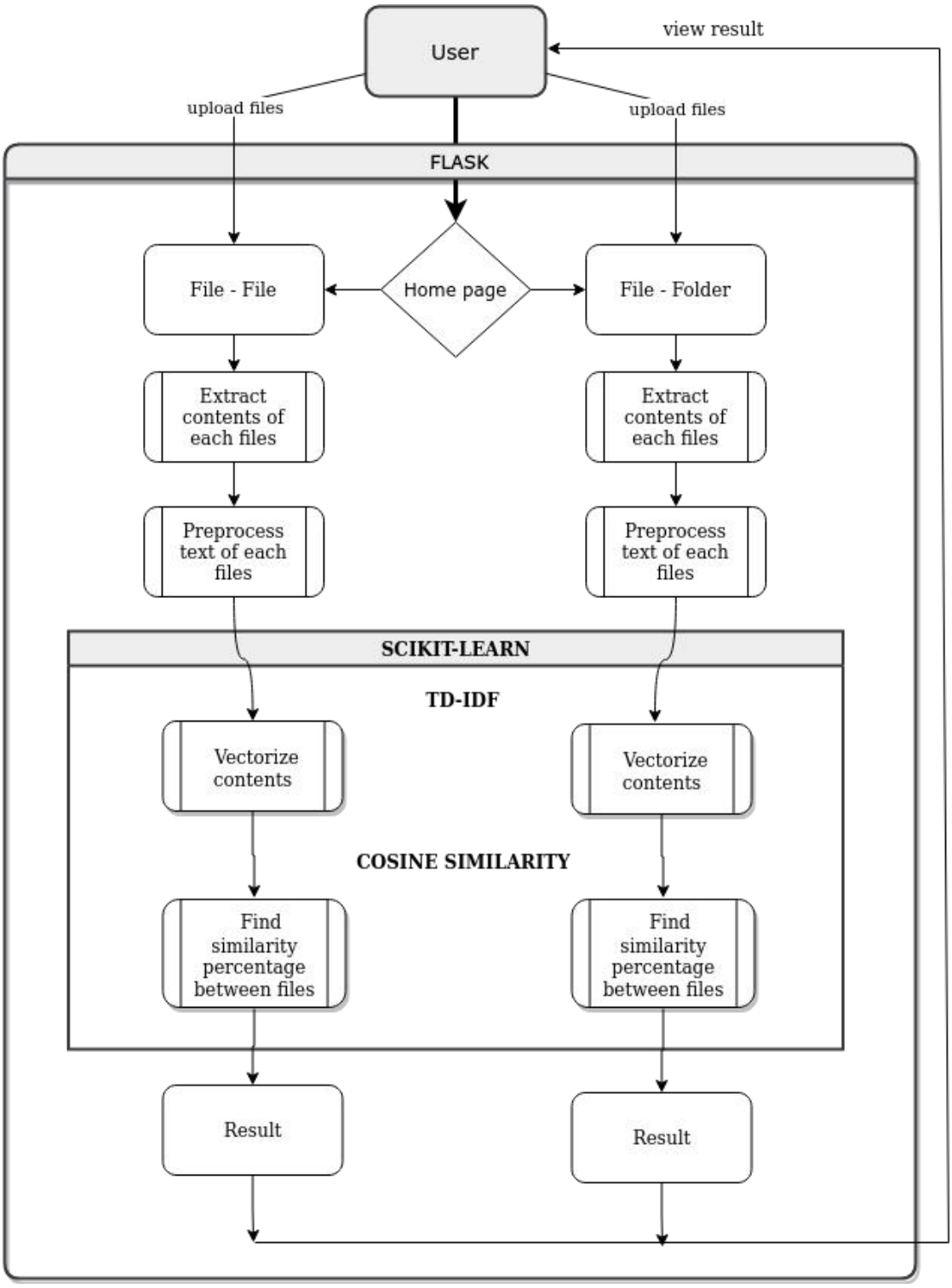

Fig A.1 Dataflow Diagram

B. SAMPLE CODING

1.Import Neccesory Packages

```
import os
from flask import *
import re
import string
import docx2txt
import PyPDF2
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import tempfile
```

2. Fuction for extracting texts

```
def read_file(filename):
    file_extension = os.path.splitext(filename)[1]
    if file_extension == ".docx":
        text = docx2txt.process(filename)

    elif file_extension == ".pdf":
        with open(filename, 'rb') as file:
            reader = PyPDF2.PdfReader(file)
            text = ""
            for i in range(len(reader.pages)):
                page = reader.pages[i]
                text += page.extract_text()

    elif file_extension in [".doc", ".txt"]:
        with open(filename, 'r') as file:
            text = file.read()
    else:
        return"Unsupported file format"
```

```
    return text
```

3.Function for preprocessing texts

```
stop_words = set(['a', 'about', 'above', 'across', 'after', 'again', 'against', 'all', 'almost', 'alone',
'along', 'already', 'also', 'although', 'always', 'among', 'an', 'and', 'another', 'any', 'anybody',
'anyone', 'anything', 'anywhere', 'are', 'area', 'areas', 'around', 'as', 'ask', 'asked', 'asking', 'asks',
'at', 'away', 'b', 'back', 'backed', 'backing', 'backs', 'be', 'became', 'because', 'become',
'becomes', 'been', 'before', 'began', 'behind', 'being', 'beings', 'best', 'better', 'between', 'big',
'both', 'but', 'by', 'c', 'came', 'can', 'cannot', 'case', 'cases', 'certain', 'certainly', 'clear', 'clearly',
'later', 'latest', 'least', 'less', 'let', 'lets', 'like', 'likely', 'long', 'longer', 'longest', 'm', 'made',
'make', 'making', 'man', 'many', 'may', 'me', 'member', 'members', 'men', 'might', 'more',
'most', 'mostly', 'mr', 'mrs', 'much', 'must', 'my', 'myself', 'n', 'necessary', 'need', 'needed',
'needing', 'needs', 'never', 'new', 'new', 'newer', 'newest', 'next', 'no', 'nobody', 'non', 'noone',
'not', 'nothing', 'now', 'nowhere', 'number', 'numbers', 'o', 'of', 'off', 'often', 'old', 'older',
'oldest', 'on', 'once', 'one', 'only', 'open', 'opened', 'opening', 'opens', 'or', 'order', 'ordered',
'ordering', 'orders', 'other', 'others', 'our', 'out', 'over', 'p', 'part', 'parted', 'parting', 'parts', 'per',
'perhaps', 'place', 'places', 'point', 'pointed', 'pointing', 'points', 'possible', 'present',
'presented', 'presenting', 'presents', 'problem', 'problems', 'put', 'puts', 'q', 'quite', 'r', 'rather',
'really', 'right', 'right', 'room', 'rooms', 's', ])

def preprocess_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\s+', ' ', text).strip()
    tokens = text.split()
    filtered_tokens = []
    for i in tokens:
        if i not in stop_words:
            filtered_tokens.append(i)
    text = ' '.join(filtered_tokens)
    return text
```

4.Object of FLASK

```
app=Flask(__name__)
```

5.Routes of home and other input pages

```
@app.route('/')
def thome():
    return render_template('home.html')

@app.route('/file')
def home():
    return render_template('index.html')

@app.route('/folder')
def index():
    return render_template('index2.html')
```

6. Route and function of file-file detector

```
@app.route('/result', methods=['POST'])
def result():
    file1 = request.files['file1']
    file2 = request.files['file2']
    # Save uploaded files to a temporary directory
    temp_dir1 = tempfile.TemporaryDirectory()
    file1_path = os.path.join(temp_dir1.name, file1.filename)
    file2_path = os.path.join(temp_dir1.name, file2.filename)
    file1.save(file1_path)
    file2.save(file2_path)

    text1 = read_file(file1_path)
    text2 = read_file(file2_path)
    text1 = preprocess_text(text1)
    text2 = preprocess_text(text2)
```

```
    vectorizer = TfidfVectorizer()

    X = vectorizer.fit_transform([text1, text2]).toarray()
    similarity = cosine_similarity(X[0].reshape(1,-1), X[1].reshape(1,-1))[0][0]
    percentage_similarity = round(similarity * 100, 2)
    return render_template('result.html', percentage_similarity=percentage_similarity)
```

7. Route and Function of File-Folder detector

```
@app.route('/result2', methods=['POST','GET'])
def upload():

    file = request.files['file']
    folder = request.files.getlist('folder')
    temp_file=tempfile.TemporaryDirectory()
    temp_dir= tempfile.TemporaryDirectory()

    file_path = os.path.join(temp_file.name, file.filename)
    file.save(file_path)
    path,qwery_filename=os.path.split(file_path)

    folder_paths = []
    for f in folder:
        folder_path = os.path.join(temp_dir.name, f.filename)
        f.save(folder_path)
        folder_paths.append(folder_path)

    query_file_text = read_file(file_path)
    query_file_text = preprocess_text(query_file_text)
    similarity_dict = {}
    for filename in folder_paths:
          if filename.endswith('.pdf') or filename.endswith('.docx') or
filename.endswith('.doc') or filename.endswith('.txt'):
```

```
            file_text = read_file(filename)
            file_text = preprocess_text(file_text)
            vectorizer = TfidfVectorizer().fit_transform([query_file_text, file_text])
            vectors = vectorizer.toarray()
            similarity = cosine_similarity(vectors[0].reshape(1, -1), vectors[1].reshape(1, -
1))[0][0]
            f,name=os.path.split(filename)
            similarity_dict[name] = round(similarity * 100, 2)

    for folder_path in folder_paths:
        os.remove(folder_path)
    return render_template('result2.html', similarity_dict=similarity_dict)
    if __name__ == '__main__':
        app.run(debug=True,port=5000)
```

8. Home page

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>PLAGCHEK</title>
        <style>
        body {
                background-color: #232c30;font-family: Arial, sans-serif;
                }
        h1 {
                text-align: center;color:darkorange ;margin-top: 50px;margin-bottom: 5px;
                font-size: 45px;font-weight: bolder;text-shadow: 2px 2px 4px #212121;
                }
        p{
                color: #fff;font-size: 24px;font-weight: normal;text-align: center;
                margin-top: 10px;margin-bottom: 50px;text-shadow: 2px 2px 4px #212121;
                }
```

```
        a {
                text-decoration: none;color: #fff;font-size: 18px;font-weight: normal;
                display: block;width: 300px;margin: 0 auto;padding: 15px;border-radius:
        5px;
                background-color: #1a65d6;box-shadow: 2px 2px 4px #212121;
                text-align: center;margin-bottom: 20px;
                }
        a:hover {
                background-color: #3679df;
                }
         .footer {
                background-color: #1c1c1c;color: #fff;font-size: 14px;text-align: center;
                padding: 15px;position: fixed;bottom: 0;left: 0;right: 0;
                }
        </style>
</head>
<body>
        <h1>PLAGCHEK</h1>
        <p></p>
        <a href="/file">File-File</a>
        <a href="/folder">File-Folder</a>
   <div class="footer">
                PLAGCHEK&trade; created by Ashique  2023&copy;
        </div>
</body>
</html>
```

9.File-File input page

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>File-File</title>
                <style>
```

```
body {
        background-color: #232c30;font-family: 'optima', sans-serif;
        }
h1 {
        text-align: center;color:darkorange ;margin-top: 50px;margin-bottom:
5px;font-       size: 45pxfont-weight: bolder;text-shadow: 2px 2px 4px #212121;
        }
 h2 {
        text-align: center;color:#f3f1f1 ;margin-top: 5px;margin-bottom: 5px;
        font-size: 20px;font-weight: normal;text-shadow: 2px 2px 4px #212121;
        }
button {
        background-color: #1a65d6;color: #fff;font-size: 20px;font-weight:
normal;border:          none;margin-left:8%;  margin-right:auto;border-radius:
5px;padding: 10px      20px;box-shadow: 2px 2px 4px #212121;cursor: pointer;
        }
button:hover {
        background-color: #3679df;
        }
.form-container {
        background-color: #2c3b42;box-shadow: 2px 2px 4px #1d1e1f;border-
radius: 13px;padding: 20px;margin-left:auto;margin-right:auto;width:85%;margin-
top:     15px;margin-bottom: 50px;text-align: center;
        }
label{
        font-size: 17px;color: rgb(236, 238, 240);
        }
form {
        display: inline-block;font-size:18px;text-align:left;
        }
input[type="file"] {
        display: block;margin: 0 auto;margin-bottom: 20px;padding: 10px;width:
        80%;border-radius: 5px;box-shadow: 2px 2px 4px #212121;border:
```

```
        none;outline:   none;background-color: #9ba3a7;color: #2e2d2d;font-size:
        19px;font-weight:       normal;cursor: pointer;
                }
        input[type="submit"] {
                background-color: #1a65d6;color: #fff;font-size: 20px;align:center;font-
        weight:bold;margin-left:30%;border: none;border-radius: 5px;padding: 10px
        20px;box-       shadow: 2px 2px 4px #212121;display:inline-block;cursor: pointer;
                }
        input[type="submit"]:hover {
                background-color: #3679df;}

        .footer {
                background-color: #1c1c1c;color: #fff;font-size: 14px;text-align:
        center;padding:          15px;position: fixed;bottom: 0;left: 0;right: 0;
                }
        </style>

</head>
<body>
        <h1>PLAGCHEK</h1>
        <h2>File-File compare</h2>
        <a href="/folder"><button>File-Folder</button></a>
        <div class="form-container">
                <form action="/result" method="POST" enctype="multipart/form-data">
                        <label for="file1">First File:</label><br>
                        <input type="file" id="file1" name="file1"
accept=".txt,.doc,.docx,.pdf" required="TRUE"><br>
                        <label for="file2">Second file:</label><br>
                        <input type="file" id="file2" name="file2"
accept=".txt,.doc,.docx,.pdf" required="TRUE"><br>
                        <input type="submit" value="CHECK">
                </form>
        </div>
        <div class="footer">
```

```
            PLAGCHEK&amp;trade; created by Ashique  2023&amp;copy;
        </div>
</body>
</html>
```

10.File-Folder input page

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>File-Folder</title>
        <style>
        body {
                background-color: #232c30;font-family: 'optima', sans-serif;
                }
        h1 {
                text-align: center;color:darkorange ;margin-top: 50px;margin-bottom:
        5px;font-       size: 45pxfont-weight: bolder;text-shadow: 2px 2px 4px #212121;
                }
         h2 {
                text-align: center;color:#f3f1f1 ;margin-top: 5px;margin-bottom: 5px;
                font-size: 20px;font-weight: normal;text-shadow: 2px 2px 4px #212121;
                }
        button {
                background-color: #1a65d6;color: #fff;font-size: 20px;font-weight:
        normal;border:          none;margin-left:8%;  margin-right:auto;border-radius:
        5px;padding: 10px      20px;box-shadow: 2px 2px 4px #212121;cursor: pointer;
                }
        button:hover {
                background-color: #3679df;
                }
        .form-container {
```

```
        background-color: #2c3b42;box-shadow: 2px 2px 4px #1d1e1f;border-
    radius: 13px;padding: 20px;margin-left:auto;margin-right:auto;width:85%;margin-
    top:    15px;margin-bottom: 50px;text-align: center;
        }
    label{
        font-size: 17px;color: rgb(236, 238, 240);
        }
    form {
        display: inline-block;font-size:18px;text-align:left;
        }
    input[type="file"] {
        display: block;margin: 0 auto;margin-bottom: 20px;padding: 10px;width:
        80%;border-radius: 5px;box-shadow: 2px 2px 4px #212121;border:
    none;outline:  none;background-color: #9ba3a7;color: #2e2d2d;font-size:
    19px;font-weight:    normal;cursor: pointer;
        }
    input[type="submit"] {
        background-color: #1a65d6;color: #fff;font-size: 20px;align:center;font-
    weight:bold;margin-left:30%;border: none;border-radius: 5px;padding: 10px
    20px;box-    shadow: 2px 2px 4px #212121;display:inline-block;cursor: pointer;
        }
    input[type="submit"]:hover {
        background-color: #3679df;}

    .footer {
        background-color: #1c1c1c;color: #fff;font-size: 14px;text-align:
    center;padding:      15px;position: fixed;bottom: 0;left: 0;right: 0;
        }
    </style>

</head>
<body>
  <h1>PLAGCHEK</h1>
  <h2>File-Folder compare</h2>
```

```
   <a href="/file"><button>File-File</button></a>
        <div class="form-container">
        <form action="/result2" method="POST"  enctype="multipart/form-data">
        <label for="file">Qwery File:</label><br>
        <input type="file" name="file" id="file" accept=".pdf,.docx,.doc,.txt"
required="TRUE"><br>
        <label for="folder">Select multiple files to compare:</label><br>
        <input type="file" name="folder" id="folder" multiple required="TRUE"><br>
        <input type="submit" value="CHECK">
     </form>
   </div>
   <div class="footer">
                PLAGCHEK&trade; created by Ashique  2023&copy;
        </div>
</body>
</html>
```

11.File-File Result page

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Plagiarism Checker - Result</title>
        <style>
        body {
                background-color: #232c30;font-family: 'optima', sans-serif;
                }
        h1 {
             text-align: center;color:darkorange ;margin-top: 50px;margin-bottom: 5px;
                font-size: 45px;font-weight: bolder;text-shadow: 2px 2px 4px #212121;
                }
         h2 {
                text-align: center;color:#f3f1f1 ;margin-top: 5px;margin-bottom: 5px;
```

```
            font-size: 25px;font-weight: normal;text-shadow: 2px 2px 4px #212121;
            }
        .result-container {
            background-color: #2c3b42;box-shadow: 2px 2px 4px #1d1e1f;
            border-radius: 13px;    padding: 20px;margin-top: 45px;margin-bottom:
            50px;margin-left:auto; margin-right:auto;width:85%;text-align: center;color:
        #fff;
            }
        h3{
            font-size: 20px;font-weight: normal;margin-bottom: 18px;
            }
        p {
            font-size: 50px;color: darkorange;font-family: Lucida Sans;
            font-weight: bold;margin-bottom: 30px;text-shadow: 0px 0px 5px #f3a75f;
            }
        button {
            background-color:#1a65d6;    color: #fff;       font-size: 20px;font-weight:
                bold;border: none;border-radius: 5px;padding: 10px 20px;
            box-shadow: 2px 2px 4px #212121;cursor: pointer;margin-top: 20px;
            }
        button:hover {
            background-color: #3679df;
            }
        .footer {
            background-color: #1c1c1c;color: #fff;font-size: 14px;text-align: center;
            padding: 15px;position: fixed;bottom: 0;left: 0;right: 0;
            }
        </style>
</head>
<body>
        <h1>PLAGCHEK</h1>
        <h2>RESULT</h2>
        <div class="result-container">
            <h3>Similarity between two files</h3>
```

```
            <p> {{ percentage_similarity }}%</p>
            <button onclick="window.history.back()">Check another</button>
        </div>
        <div class="footer">
            PLAGCHEK&trade; created by Ashique  2023&copy;
        </div>
</body>
</html>
```

12.File-Folder Result page

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Results</title>
        <style type="text/css">
        body {
            background-color: #232c30;font-family: 'optima', sans-serif;padding-
bottom:75px;
            }
        h1 {
            text-align: center;color:darkorange ;margin-top: 50px;
            margin-bottom: 5px;font-size: 45px;font-weight: bolder;
            text-shadow: 2px 2px 4px #212121;
            }

         h2 {
            text-align: center;color:#f3f1f1 ;margin-top: 5px;margin-bottom: 5px;
            font-size: 25px;font-weight: normal;text-shadow: 2px 2px 4px #212121;
            }
        table {
             border-collapse:collapse;border-radius:10px;margin-top: 50px;
            margin-bottom: 50px; margin-left:auto; margin-right:auto;width:75%;
```

```
        overflow:hidden;box-shadow: 0px 0px 8px #1d1e1f;
        }
th{
        padding: 20px;border-bottom: 1px solid darkorange ;text-align: center;
        font-size: 22px;font-weight: bold;color: #fff;background-color: #156b8a;
        }
 td{
        padding: 15px;text-align: left;border-top: 1px solid #AAAAAA ;
        font-size: 20px;font-weight: bold;color: #fff;background-color: #2c3b42;
        }
th:first-child,td:first-child {
        width: 50%;
        }
tr:nth-child(even) {
        background-color: #1f2c30;
        }
tr:hover {
        background-color: #3f3f3f;
         }
 td.similarity {
        font-size: 24px;color: darkorange; text-align:center;font-weight: bold;
        margin-bottom: 30px;text-shadow: 0px 0px 2px #f3a75f;
        }
button {
        background-color:#1a65d6;color: #fff;font-size: 20px;font-weight: bold;
        border: none;border-radius: 5px;padding: 10px 20px;
        box-shadow: 2px 2px 4px #212121;cursor: pointer;
        margin-top: 20px;margin-bottom:50px; display: block;margin: 0 auto;
        }
button:hover {
        background-color: #3679df;
        }
.footer {
        background-color: #1c1c1c;color: #fff;font-size: 14px;
```

```
            text-align: center;padding: 15px;position: fixed;bottom: 0;left: 0;right: 0;
            }
        </style>
</head>
<body>
        <h1>PLAGCHEK</h1>
        <h2>RESULT</h2>
        <table>
            <thead>
                <tr>
                    <th>File Name</th>
                    <th>Similarity Percentage</th>
                </tr>
            </thead>
            <tbody>
                {% for filename, similarity in similarity_dict.items() %}
                <tr>
                    <td>{{ filename }}</td>
                    <td class="similarity">{{ similarity }}%</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        <button onclick="window.history.back()">Check another</button>
   <div class="footer">
            PLAGCHEK&amp;trade; created by Ashique  2023&amp;copy;
        </div>
</body>
</html>
```

C.SAMPLE INPUT

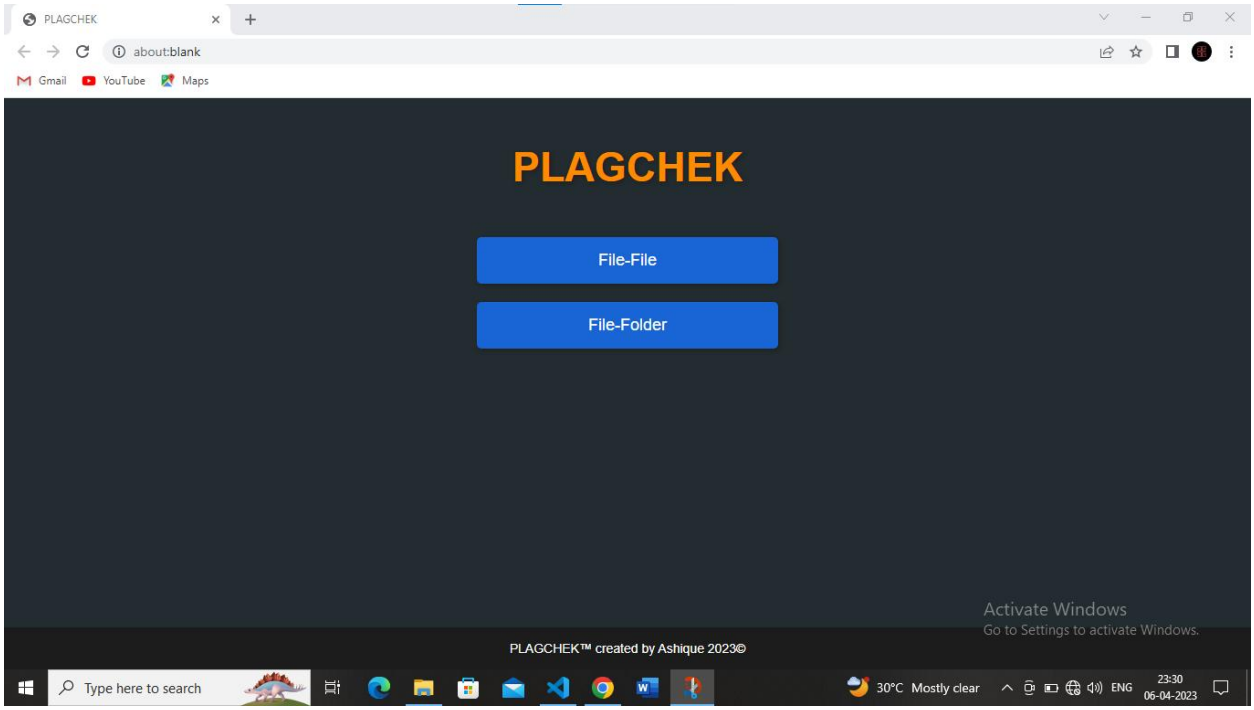

Fig1:Screenshot of home page

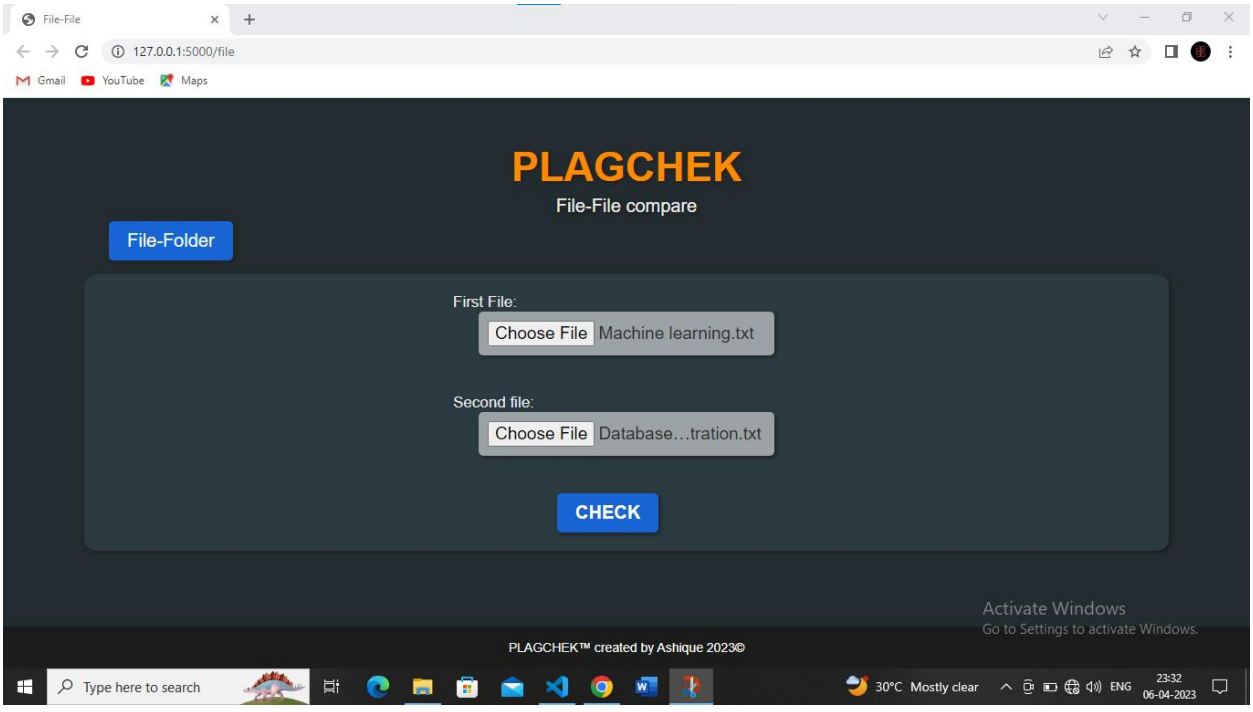

Fig2:Screenshot of File-File page

Fig3:Screenshot of File-Folder page

D.SAMPLE OUTPUT

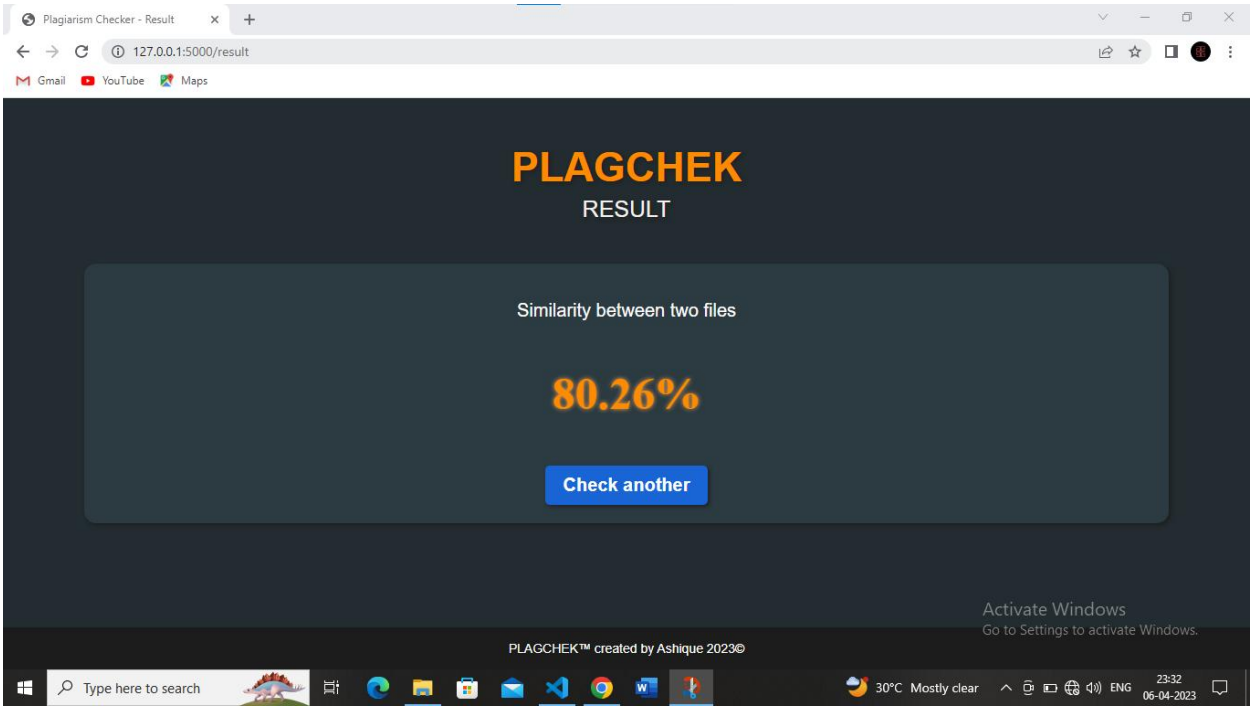

Fig4:Screenshot of File-File Result page

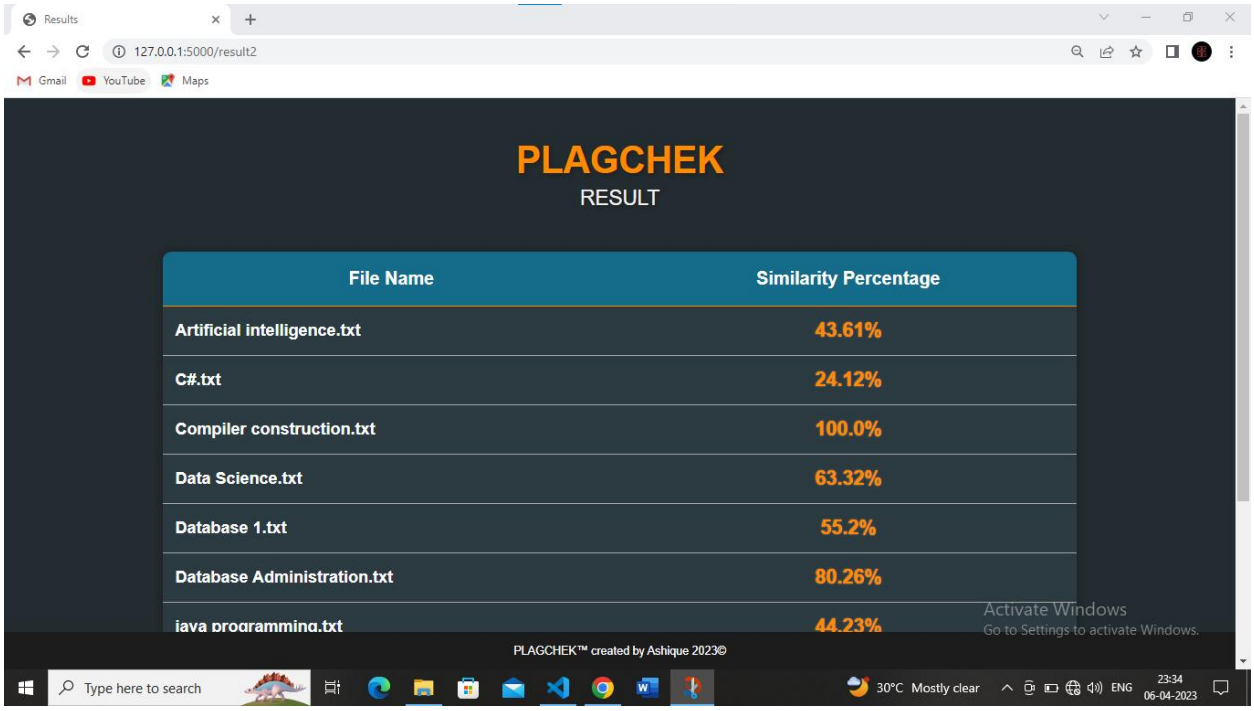

Fig5:Screenshot of File-Folder Result page