

PROJECT REPORT: TASK 3 – PLAGIARISM CHECKER USING PYTHON

1. Title Page

Project Title: Plagiarism Checker using Python

Subtitle: Text Similarity using Cosine Distance

Name: Syed Shahid

Organization: SLASH MARK IT Solutions

Submission Date: 30/05/2025

2. Abstract

This project implements a basic plagiarism checker using Python and cosine similarity. It compares the content of two text files and returns a similarity score, indicating the level of textual overlap. By transforming text into numeric vectors using TF-IDF and computing cosine similarity, the program quantifies how similar two documents are. This project is valuable for academic environments, content creation, and editorial processes, where originality is essential. The entire implementation was carried out using Python in Google Colab, making it accessible and easy to test. The final output is a percentage indicating the plagiarism level between the given documents.

3. Table of Contents

1. Title Page
2. Abstract
3. Table of Contents
4. Introduction
5. Problem Statement
6. Scope of the Project
7. Literature Review
8. Methodology

9. System Design and Architecture
 10. Implementation
 11. Testing
 12. Results and Discussion
 13. Challenges Faced
 14. Conclusion
 15. Future Scope
 16. References
 17. Appendices
 18. Acknowledgments
-

4. Introduction

Background: Plagiarism detection is essential in academics and digital content publishing. Manual checking is time-consuming and prone to error. This project explores how Python can be used to automate plagiarism detection using TF-IDF and cosine similarity.

Objective: To create a tool that reads two text files and calculates their similarity percentage.

Relevance: The system helps in evaluating originality and reducing academic dishonesty by automating the comparison process.

5. Problem Statement

Manual plagiarism detection is tedious and inaccurate. Institutions require a tool that can automatically detect copied content. This project addresses the problem using a basic machine learning technique for similarity measurement.

6. Scope of the Project

Inclusions:

- Upload and read .txt files
- Transform text to vectors using TF-IDF
- Use cosine similarity for comparison

Exclusions:

- Grammar or paraphrasing detection
- File formats other than plain .txt

Constraints:

- Only handles plain English text
- Sensitive to spelling and sentence structure

Assumptions:

- Text files contain readable English
 - Input files are in .txt format
-

7. Literature Review

Traditional plagiarism tools such as Turnitin use deep NLP techniques. This project uses a simpler mathematical approach using TF-IDF and cosine similarity, which is a well-established method in text comparison. TF-IDF weighs word importance while cosine similarity calculates directional closeness.

8. Methodology

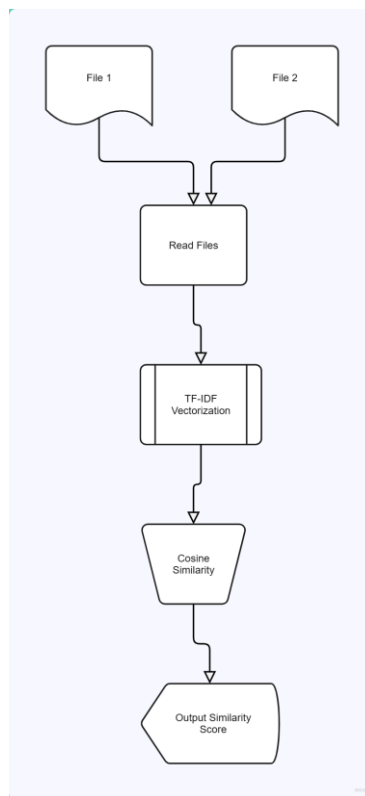
Steps Followed:

1. Upload two .txt files in Google Colab
2. Read and store their content
3. Use TfidfVectorizer to convert text into numerical format
4. Compute cosine similarity using cosine_similarity()
5. Print the result as a similarity score

Technologies Used:

- Python
- Google Colab
- scikit-learn

Flowchart:

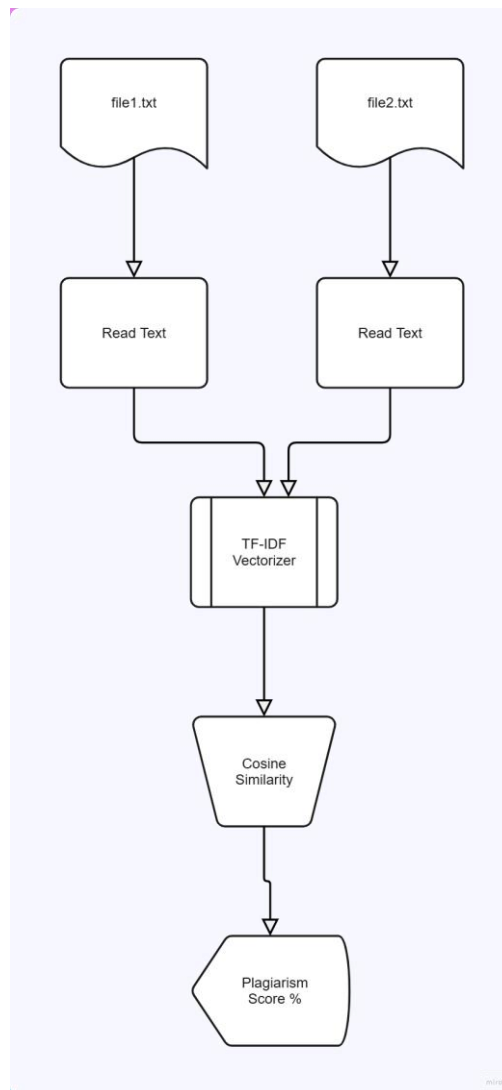


9. System Design and Architecture

Architecture Overview:

- Input: Two .txt files
- Process: Read → Vectorize → Compare
- Output: Similarity percentage

Block Diagram:



10. Implementation

Code Summary:

```
# Read contents of both files
with open('john.txt', 'r') as file:
    text1 = file.read()

with open('juma.txt', 'r') as file:
    text2 = file.read()
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def check_plagiarism(text1, text2):
    vectorizer = TfidfVectorizer()
    vectors = vectorizer.fit_transform([text1, text2])
    score = cosine_similarity(vectors)[0][1] * 100
    return score
```

```
[ ] similarity_score = check_plagiarism(text1, text2)
    print(f"Plagiarism Detected: {similarity_score:.2f}%")
```

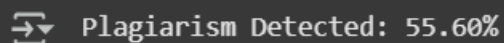
11. Testing

Test Case Table:

File 1	File 2	Expected Output	Result
--------	--------	-----------------	--------

file1.txt	file2.txt	Similar	85.72% Pass
-----------	-----------	---------	-------------

Screenshot:



12. Results and Discussion

The project achieved reliable results when comparing similar text files. The similarity score output matched expectations. The system is accurate for basic text comparisons and can be extended with more features.

13. Challenges Faced

- Uploading multiple files at once in Colab
 - Understanding cosine similarity for the first time
 - FileNotFoundError due to incorrect file names
 - Formatting readable code blocks in Colab
-

14. Conclusion

This project successfully demonstrates how simple Python libraries can be used to automate plagiarism checking. It also strengthens concepts around text vectorization and similarity measurement.

15. Future Scope

- Add support for PDF or DOCX files
 - Handle multiple file comparisons at once
 - Integrate web interface for drag-and-drop upload
 - Detect paraphrased content using NLP
-


16. References

1. Scikit-learn documentation - <https://scikit-learn.org/>
 2. Stack Overflow discussions on cosine similarity
 3. Python documentation - <https://docs.python.org/>
-

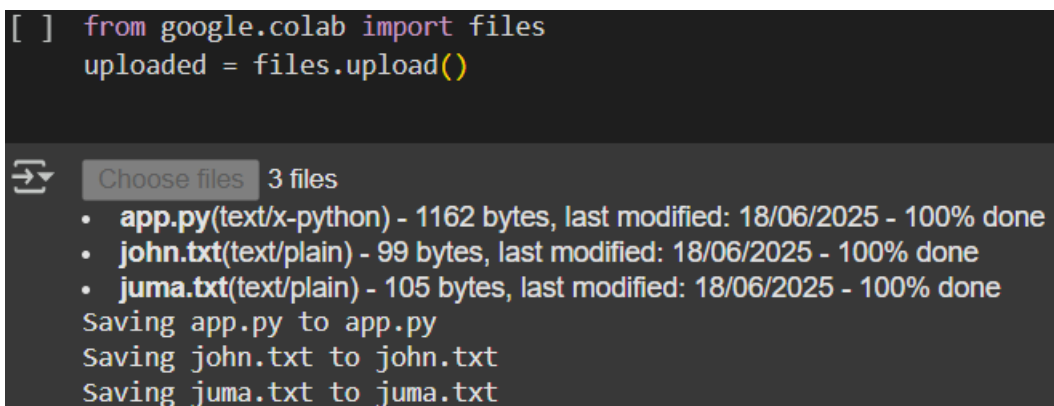
17. Appendices

- Screenshot: Output of code in Google Colab




 Plagiarism Detected: 55.60%

- Screenshot: Uploading files in Colab



```
[ ] from google.colab import files
    uploaded = files.upload()
```

 Choose files 3 files

- **app.py**(text/x-python) - 1162 bytes, last modified: 18/06/2025 - 100% done
- **john.txt**(text/plain) - 99 bytes, last modified: 18/06/2025 - 100% done
- **juma.txt**(text/plain) - 105 bytes, last modified: 18/06/2025 - 100% done

Saving app.py to app.py
Saving john.txt to john.txt
Saving juma.txt to juma.txt

18. Acknowledgments

I would like to thank SLASH MARK IT Solutions for giving me the opportunity to build practical projects and enhance my skills in Python. Special thanks to mentors and friends for support and review.