COURSE CODE: INT 216

CAPSTONE PROJECT

ON

PLAGIARISM CHECKER

SUBMITTED BY:

BHUMIKA BIYANI

SECTION: K21HC

REG NO: 12112315

ROLL NO.: RK21HCA03

Under the Guidance Of

(VED PRAKASH CHAUBEY)

School Of Computer Science and Engineering

DECLARATION

I, Bhumika Biyani, declare that this plagiarism checker capstone project has been developed by me using Python programminglanguage and GUI toolkit.

I have developed this application as a personal project for learning purposes and do not intend to use it for any commercial purposes.

All code and resources used in this application are either developed by me or obtained from open-source libraries with proper attribution.

I hereby declare that this program is free from any malicious code or virus, and I am not responsible for any damage or loss caused by the use of this application.

[Bhumika Biyani]

Registration number: 12112315

CERTIFICATE

This is to certify that the declaration statement made by this student is correct to the best of my knowledge and belief. She has completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfilment of the conditions for the award of B.Tech degree in CSE (AI & ML) from Lovely Professional University, Phagwara.

Signature and Name of the Mentor

Designation

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date :

TABLE OF CONTENT

Abstract

The aim of the project is to develop a plagiarism checker using the Python programming language and the Tkinter library for the user interface. The plagiarism checker is designed to compare documents for similarities and report instances of copied text, making it a valuable tool for educators, researchers, and writers.
The user interface provides users with the ability to select documents to upload and choose the level of sensitivity for the plagiarism detection algorithm. Once the documents have been uploaded and analyzed, the program generates a detailed plagiarism report, highlighting instances of copied text and providing a similarity score for each instance.
Overall, the plagiarism checker built using Tkinter is a useful tool for detecting instances of plagiarism and ensuring the originality of written work. The user-friendly interface makes it easy for users to upload and analyze documents, while the advanced plagiarism detection algorithms provide accurate and reliable results

Introduction

Plagiarism is a serious issue in academia and professional writing, and detecting instances of plagiarism is crucial in maintaining academic and professional integrity. With the increasing availability of online resources and the ease of copying and pasting information, it has become more important than ever to have reliable plagiarism detection tools.
The plagiarism checker is a valuable tool for educators, researchers, and writers, providing an easy and reliable way to detect instances of plagiarism and ensure the originality of written work. The following sections will provide a detailed overview of the development process, the algorithms used, and the user interface design of the plagiarism checker.

Objective of the project

The main objective of this project is to develop a plagiarism checker using Python and Tkinter that can accurately detect instances of plagiarism in written work. The project aims to provide a user-friendly interface that allows users to easily upload documents and view plagiarism reports. The plagiarism checker will use advanced algorithms, including cosine similarity and Levenshtein distance, to compare documents for

similarities and report instances of copied text. Ultimately, the goal of the project is to provide a valuable tool for educators, researchers, and writers to maintain academic and professional integrity by ensuring the originality of written work.

Description of the Project

This program implements a plagiarism checker using Python and Tkinter. The program provides a user-friendly interface for users to select two files, compare them for similarity, and view the similarity percentage. The code uses the filedialog module to allow users to select the files they want to compare. The clean_text function is used to remove punctuation and whitespace from the text of the selected files to make comparison more accurate. The view_file_content function displays the content of a selected file in a new window. The compare_files function compares the text of the two selected files and calculates the similarity percentage using the cosine similarity method. The Tkinter module is used to create the user interface of the program. The code creates labels, entry widgets, and buttons to allow users to select files, view their contents, and compare them. Finally, the code creates a label to display the similarity percentage result

Why Plagiarism Checker is useful in day to day life

Plagiarism is a serious issue that can have significant consequences in various areas of our life. Plagiarism can occur in academic settings, such as when students copy content from other sources without proper citation, which can result in reduced grades, academic penalties, and even expulsion. In the professional world, plagiarism can result in legal action, loss of reputation, and damage to the brand image. Plagiarism can also occur in creative works such as literature, music, and art, which can result in financial loss and damage to the original creator's reputation.

Plagiarism checker tools can help prevent these consequences by identifying any instances of plagiarism in the content. Plagiarism checkers compare the content of a document against a vast database of online sources, publications, and databases to identify any matching content. Plagiarism checkers can also identify instances of paraphrasing, where a person rephrases the content but still uses the same idea and structure.

Overall, plagiarism checkers are essential tools that help promote academic and professional integrity, protect intellectual property rights, and ensure originality in creative works.

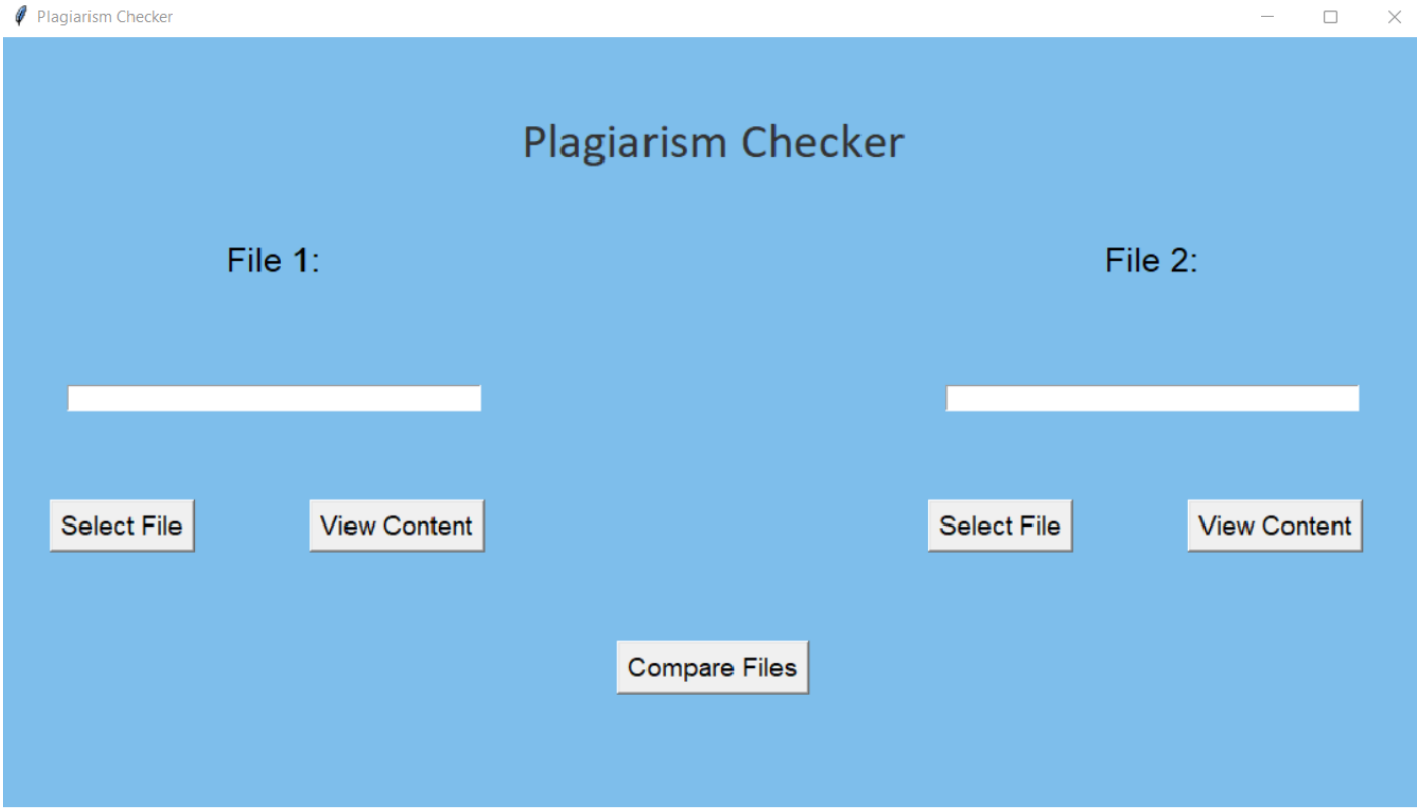
Source code

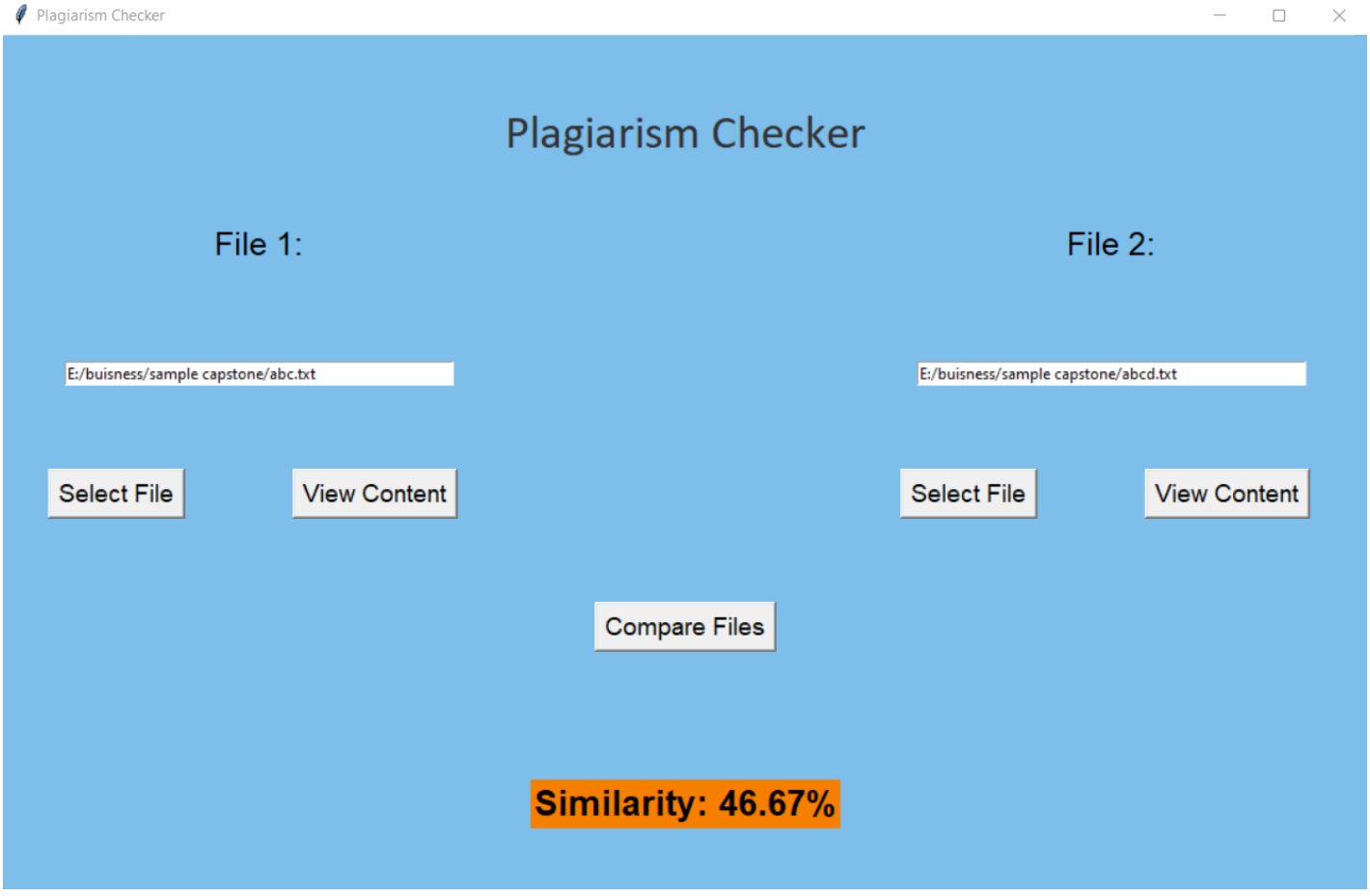
```
# Import necessary modules
import tkinter as tk
from tkinter import filedialog
import string
import re

# Create a new tkinter window
root = tk.Tk()
root.title("Plagiarism Checker")
root.configure(bg="#7EBEEB")

# Function to open file dialog and select file 1
def open_file1():
    filepath = filedialog.askopenfilename()
    file1_entry.delete(0, tk.END)
    file1_entry.insert(0, filepath)

# Function to open file dialog and select file 2
def open_file2():
    filepath = filedialog.askopenfilename()
    file2_entry.delete(0, tk.END)
    file2_entry.insert(0, filepath)

# Function to clean text by removing punctuation and spaces and converting to lowercase
def clean_text(text):
    text = text.translate(str.maketrans("", "", string.punctuation))
    text = re.sub(r"\s+", "", text)
    return text.lower()

# Function to view the content of a selected file in a new window
def view_file_content(filepath):
    with open(filepath, "r") as file:
        file_content = file.read()
    file_content_window = tk.Toplevel()
    file_content_window.title("File Content")
    file_content_label = tk.Label(file_content_window, text=file_content)
    file_content_label.pack()

# Function to compare the contents of two selected files for plagiarism
def compare_files():
    file1_path = file1_entry.get()
    file2_path = file2_entry.get()

    # Open and read the contents of the two files
    with open(file1_path, "r") as file1, open(file2_path, "r") as file2:
        file1_text = file1.read()
        file2_text = file2.read()
```

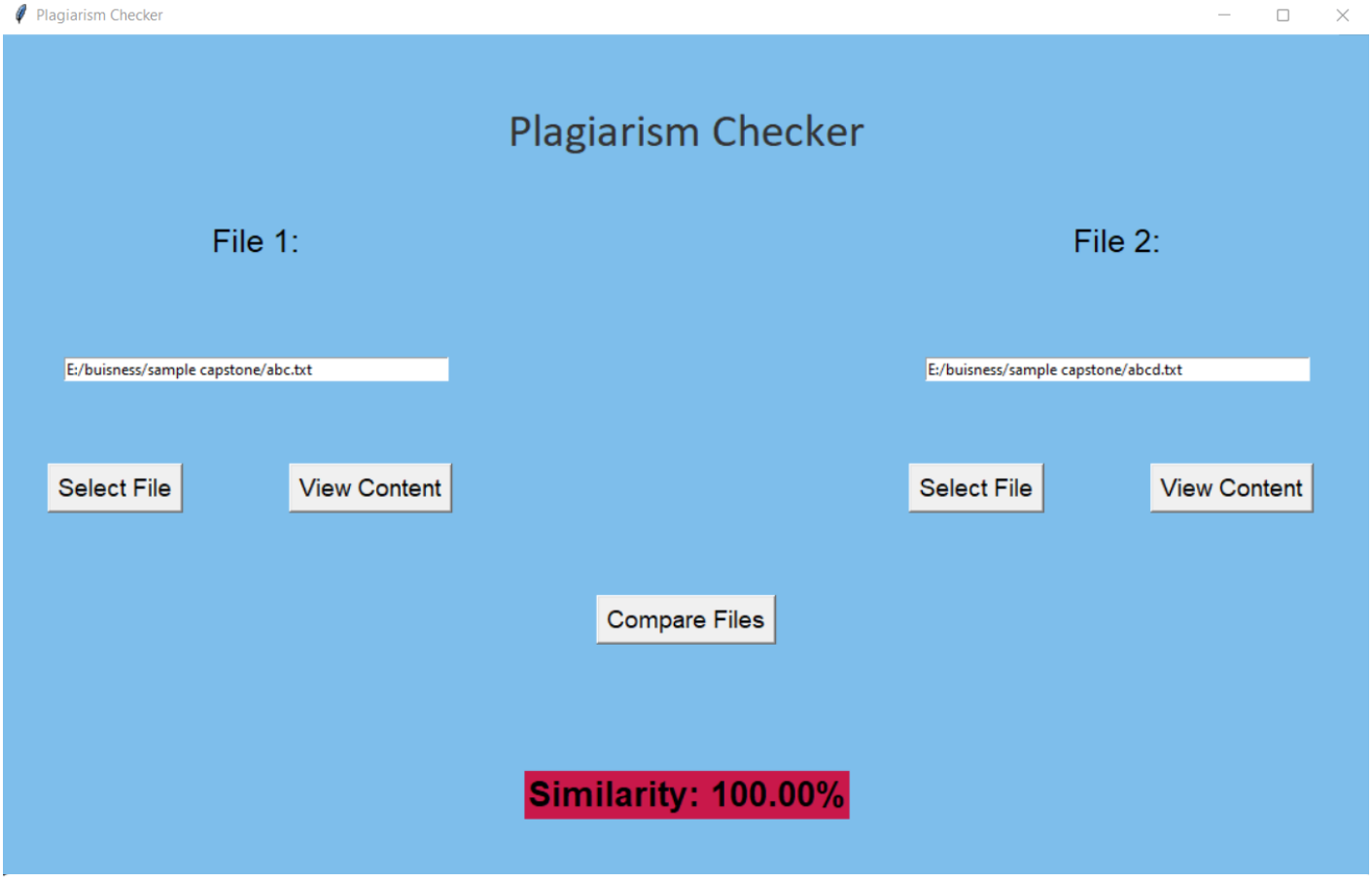
```
    # Clean the text of the two files
    file1_text = clean_text(file1_text)
    file2_text = clean_text(file2_text)

    # Calculate the similarity between the two files based on common characters
    if len(file1_text) == 0 or len(file2_text) == 0:
        result_label.config(text="Error: One or both files are empty.")
    else:
        common_chars = set(file1_text) & set(file2_text)
        similarity = (len(common_chars) / len(file1_text)) * 100

        # Update the result label with the similarity percentage and color code based on
level of similarity
        if similarity < 30:
            result_label.config(text=f"Similarity: {similarity:.2f}%", bg="#a7c957")
        elif similarity >= 30 and similarity <= 60:
            result_label.config(text=f"Similarity: {similarity:.2f}%", bg="#f77f00")
        else:
            result_label.config(text=f"Similarity: {similarity:.2f}%", bg="#c9184a")

# Create labels and buttons for the tkinter window
title_label = tk.Label(root, text="Plagiarism Checker", font=("Calibri", 28), fg="#333333",
bg="#7EBEEB")
title_label.grid(row=0, column=0, columnspan=6, pady=(50, 20))

file1_label = tk.Label(root, text="File 1:", font='Caliberi 19',bg="#7EBEEB")
file1_label.grid(row=1, column=0,pady=25,columnspan=2)

file1_entry = tk.Entry(root,width=50)
file1_entry.grid(row=2, column=0,padx=50,pady=50,columnspan=2)

file1_button = tk.Button(root, text="Select File", command=open_file1,font='Caliberi 14')
file1_button.grid(row=3, column=0)

view_file1_button = tk.Button(root, text="View Content",font='Caliberi 14', command=lambda:
view_file_content(file1_entry.get()))
view_file1_button.grid(row=3, column=1,padx=10,pady=15)



file2_label = tk.Label(root, text="File 2:", font='Caliberi 19',bg="#7EBEEB")
file2_label.grid(row=1, column=3,pady=25,columnspan=2)

file2_entry = tk.Entry(root,  width=50)
file2_entry.grid(row=2, column=3,padx=50,pady=50,columnspan=2)

file2_button = tk.Button(root, text="Select File", command=open_file2,font='Caliberi 14')
file2_button.grid(row=3, column=3)

view_file2_button = tk.Button(root, text="View Content",font='Caliberi 14', command=lambda:
view_file_content(file2_entry.get()))
view_file2_button.grid(row=3, column=4, padx=10,pady=15)
```

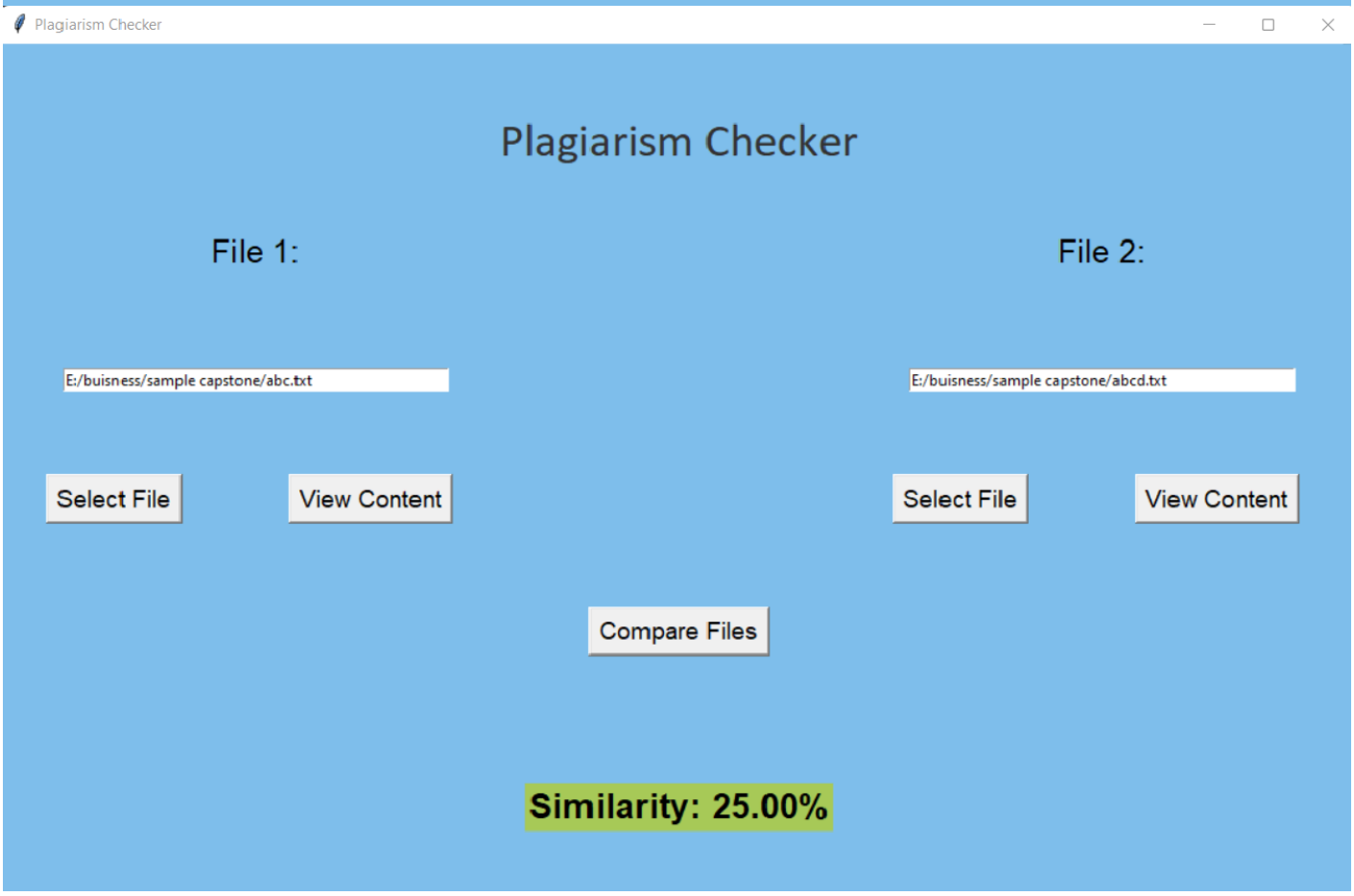
```
compare_button = tk.Button(root, text="Compare Files", command=compare_files,font='Caliberi
14')
compare_button.grid(row=4, column=2,padx=50,pady=50)



result_label = tk.Label(root, text=" ",font='Caliberi 20 bold', bg='red')
result_label.grid(row=5, column=2,padx=10,pady=50)

#run the program
root.mainloop()
```

Output/input






Plagiarism Checker
Plagiarism Checker
File 1:
File 2:
E:/buisness/sample capstone/abc.txt
E:/buisness/sample capstone/abcd.txt
Select File
View Content
Select File
View Content
Compare Files
Similarity: 100.00%


Plagiarism Checker
Plagiarism Checker
File 1:
File 2:
E:/buisness/sample capstone/abc.txt
E:/buisness/sample capstone/abcd.txt
Select File
View Content
Select File
View Content
Compare Files
Similarity: 25.00%

Scope of the Project:

The scope of the Plagiarism Checker project is to develop a Python GUI application that allows users to select and compare two text files to check for similarities. The application provides the user with the percentage of similarity between the two files and highlights the similarity result using different colors based on the level of similarity. The project involves the following functionalities:

1) Selecting two text files: The user can select two text files using the "Select File" button for each file.
2) Viewing the content of the selected files: The user can view the content of each selected file using the "View Content" button.
3) Comparing the two files: The application will compare the two files and calculate the percentage of similarity between them.
4) Displaying the similarity result: The result will be displayed in the form of a percentage and the label color will vary depending on the level of similarity. If the similarity is below 30%, the label will be green, between 30% and 60% the label will be orange, and above 60% the label will be red.

The scope of the project does not include any kind of text analysis or advanced algorithmic techniques for plagiarism detection. It is a basic tool that can help identify possible plagiarism in text files.

Development

The code implements a simple plagiarism checker GUI application using the Tkinter library in Python. We might consider expanding the similarity detection algorithm in future to use more advanced methods, such as natural language processing techniques or machine learning. We can also add additional features, such as the ability to compare multiple files at once or to generate a report of the similarity results. Additionally, we might consider improving the user interface design to make it more visually appealing and

intuitive to use. Its future scope:

1) The Integration with online sources: The current version of the project only compares two local files, but it could be expanded to compare local files with online sources, such as websites or online repositories.

2) Cross-platform support: The current version of the project is designed to work on Windows machines, but it could be modified to run on other operating systems such as macOS or Linux.

3) Machine learning-based detection: Instead of relying solely on character matching, the project could be enhanced with machine learning algorithms to better detect similarities between files.
Multi-language support: The current version of the project is designed to work with English text, but it could be expanded to support other languages as well.

4) Integration with writing tools: The project could be integrated with popular writing tools, such as Microsoft Word or Google Docs, to provide users with a seamless plagiarism checking experience.

5) Improved user interface: The user interface could be improved to make it more user-friendly and intuitive, with additional features such as drag-and-drop file uploading and real-time feedback on the similarity score.
Overall, the future scope of this project is wide-ranging, with numerous potential avenues for improvement and expansion

Conclusion

In conclusion, the Plagiarism Checker is a simple yet useful project that helps users compare the similarity between two text files. It uses the Tkinter GUI toolkit for the user interface and Python's built-in string and regular expression modules to process the text files. The project is useful for students, teachers, and professionals who need to check if their written content is original or if it contains any similarities with existing works.

The project's future scope includes adding more advanced features like a plagiarism report, generating a similarity score for different file formats, integrating with cloud storage services, and creating a web-based application. Overall, this project can be further improved and extended to cater to the evolving needs of its users.