Project Assessment Documentation

On

Plagiarism Checker

For the Assessment of

MCA First Semester (Division - 2)

In the Course

Problem Solving with Python Lab (MCP544-3)

Developed by

Jayesh Lalit Nandanwar

Ashish Shailesh Mishra

Under the Guidance of

Prof. Pravin Y. Karmore

Assistant Professor, RCOEM

Department of Computer Application

Shri Ramdeobaba College of Engineering & Management

Nagpur-13

Index

Sr. No.	Particulars	Page No.
1.	Introduction	1
2.	Aim and objective	1
3.	Flow Charts	2
4.	Coding	3
5.	Input and Output Screens	6
6.	Conclusion	8

1. Introduction

The "Plagiarism Checker" is a graphical user interface (GUI) application developed in Python using the Tkinter library. The purpose of this application is to facilitate the comparison of two text samples to determine their similarity. It allows users to either input text directly or load text from files, providing a quick and easy way to analyze textual content for potential plagiarism.

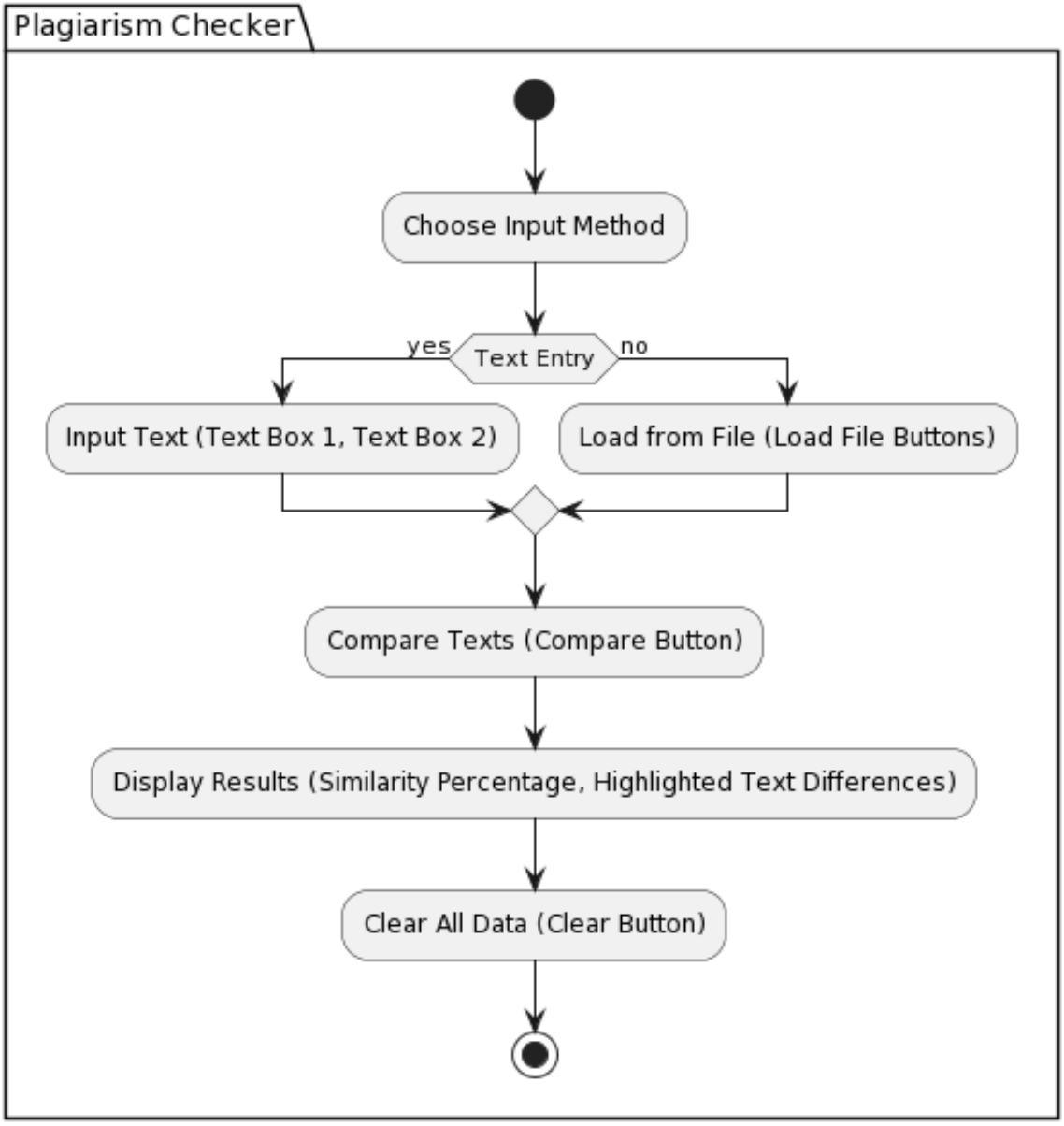
2. Aim and objective

Aim: The primary aim of the "Plagiarism Checker" project is to develop a robust and user-friendly tool that empowers users to perform efficient and accurate comparisons between two text samples. This application seeks to address the increasing importance of textual content integrity and originality, providing a reliable means for users to identify potential instances of plagiarism or content similarity.

Objectives:

1. **User-Friendly Interface:**
    - To design an intuitive graphical user interface using the Tkinter library, ensuring ease of navigation and a seamless user experience.
    - To create a visually appealing layout that facilitates clear input of text samples and easy interpretation of comparison results.
2. **Text Comparison Mechanism:**
    - To implement a sophisticated text comparison mechanism using the SequenceMatcher class from the difflib module.
    - To calculate and present a similarity percentage that reflects the likeness between the two input text samples.
    - To generate a comprehensive list of operations (opcodes) detailing the differences and similarities between the compared texts.
3. **Versatile Input Options:**
    - To allow users the flexibility of entering text manually into designated text boxes for Text 1 and Text 2.
    - To incorporate functionality enabling users to load text from external files, promoting convenience and versatility in data input.
4. **Highlighting Similarities:**
    - To highlight similar portions between the two input texts, utilizing visual cues such as color-coded text and background, aiding users in identifying common content easily.
5. **Educational and Analytical Value:**
    - To serve as an educational tool for users seeking to understand the nuances of text comparison algorithms and plagiarism detection.
    - To cater to professionals, students, and researchers by offering a practical and insightful solution for content analysis.

3. Flow Charts



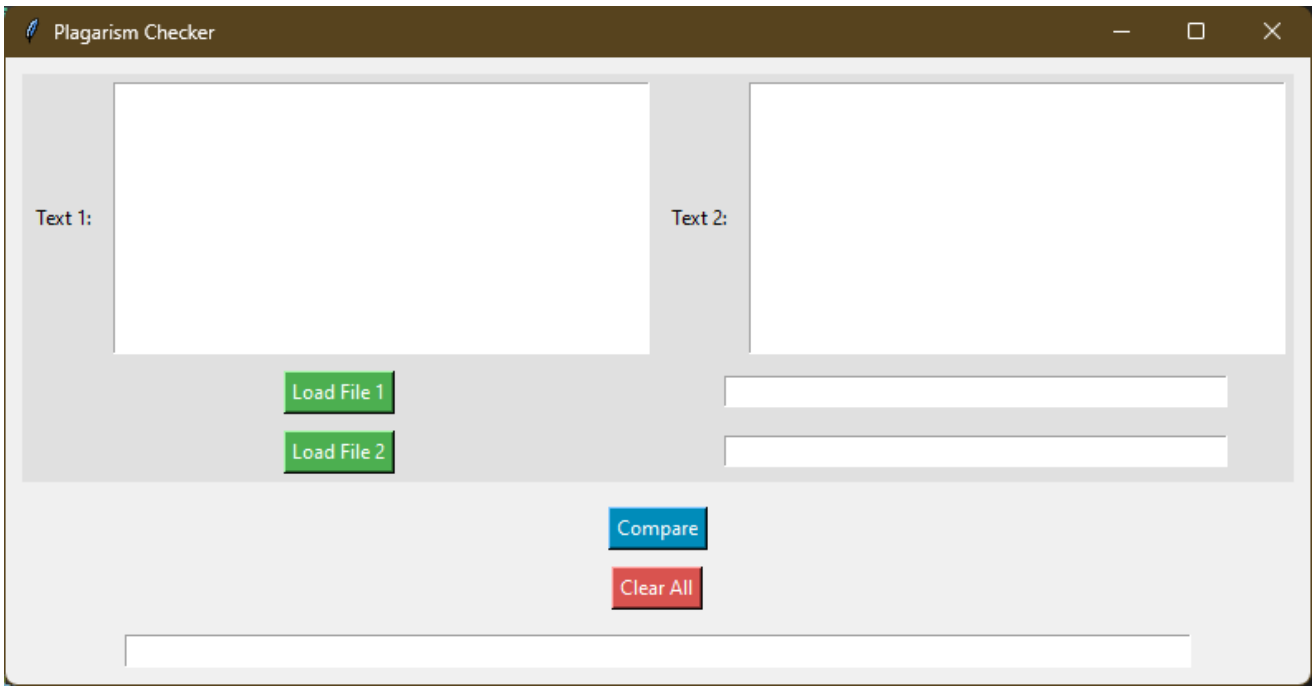
4. Coding

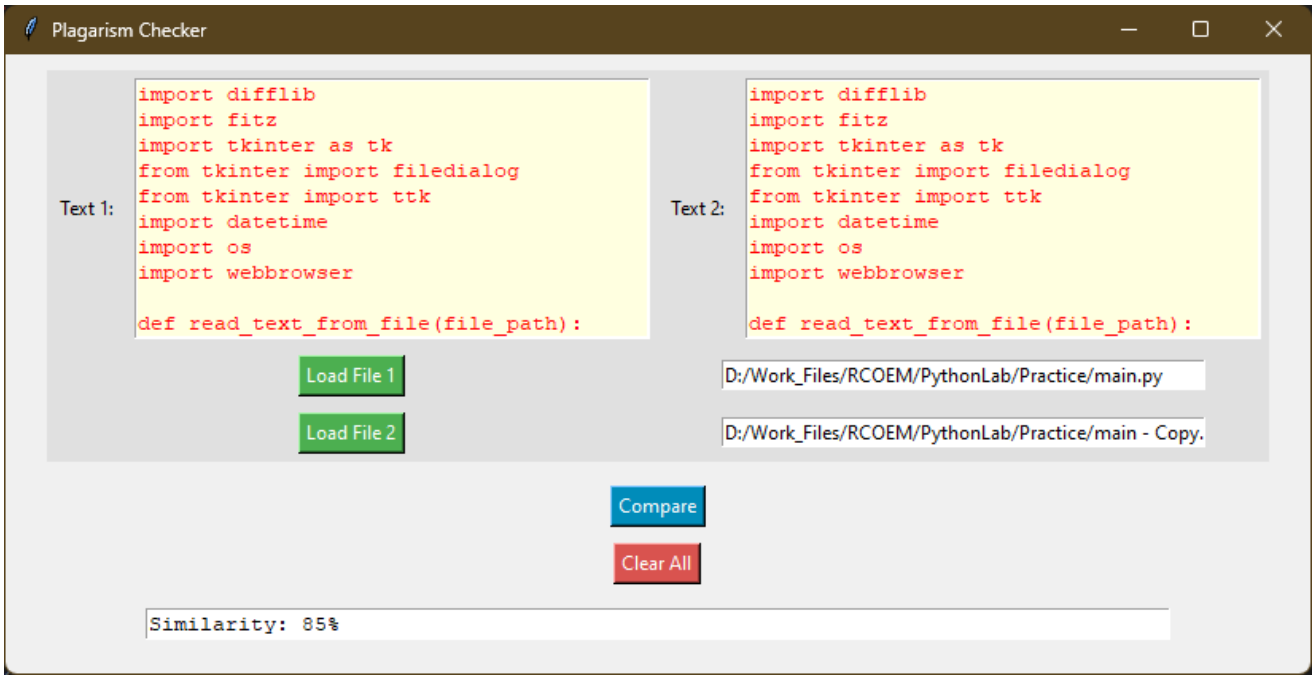
```
import tkinter as tk
from tkinter import filedialog
from difflib import SequenceMatcher

def load_file_or_display_contents(entry, text_widget):
    file_path = entry.get()

    if not file_path:
        file_path = filedialog.askopenfilename()

    if file_path:
        entry.delete(0, tk.END)
        entry.insert(tk.END, file_path)
        with open(file_path, 'r') as file:
            text = file.read()
            text_widget.delete(1.0, tk.END)
            text_widget.insert(tk.END, text)

def compare_text(text1, text2):
    d = SequenceMatcher(None, text1, text2)
    similarity_ratio = d.ratio()
    similarity_percentage = int(similarity_ratio * 100)

    diff = list(d.get_opcodes())
    return similarity_percentage, diff

def show_similarity():
    text1 = text_textbox1.get(1.0, tk.END)
    text2 = text_textbox2.get(1.0, tk.END)
    similarity_percentage, diff = compare_text(text1, text2)

    text_textbox_diff.delete(1.0, tk.END)
    text_textbox_diff.insert(tk.END, f"Similarity: {similarity_percentage}%")
```

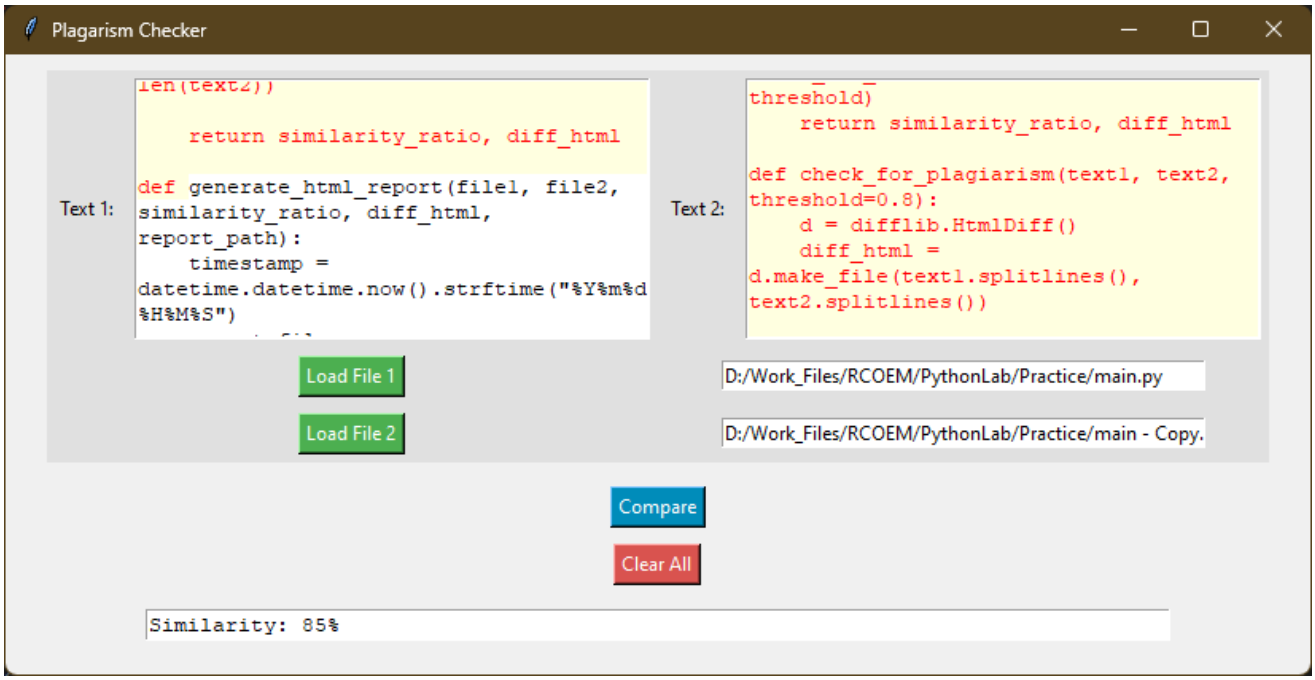
```
    text_textbox1.tag_remove("same", "1.0", tk.END)
    text_textbox2.tag_remove("same", "1.0", tk.END)

    for opcode in diff:
        tag = opcode[0]
        start1, end1, start2, end2 = opcode[1], opcode[2], opcode[3], opcode[4]

        if tag == "equal":
            text_textbox1.tag_add("same", f"1.0+{start1}c", f"1.0+{end1}c")
            text_textbox2.tag_add("same", f"1.0+{start2}c", f"1.0+{end2}c")

def clear_all():
    text_textbox1.delete(1.0, tk.END)
    text_textbox2.delete(1.0, tk.END)
    text_textbox_diff.delete(1.0, tk.END)
    file_entry1.delete(0, tk.END)
    file_entry2.delete(0, tk.END)

root = tk.Tk()
root.title("Plagarism Checker")
root.configure(bg="#F0F0F0")

frame = tk.Frame(root, bg="#E0E0E0")
frame.pack(padx=10, pady=10)

text_label1 = tk.Label(frame, text="Text 1:", bg="#E0E0E0")
text_label1.grid(row=0, column=0, padx=5, pady=5)

text_textbox1 = tk.Text(frame, wrap=tk.WORD, width=40, height=10)
text_textbox1.grid(row=0, column=1, padx=5, pady=5)

text_label2 = tk.Label(frame, text="Text 2:", bg="#E0E0E0")
text_label2.grid(row=0, column=2, padx=5, pady=5)

text_textbox2 = tk.Text(frame, wrap=tk.WORD, width=40, height=10)
text_textbox2.grid(row=0, column=3, padx=5, pady=5)
```

```
file_entry1 = tk.Entry(frame, width=50)
file_entry1.grid(row=1, column=2, columnspan=2, padx=5, pady=5)

load_button1 = tk.Button(frame, text="Load File 1", command=lambda:
load_file_or_display_contents(file_entry1, text_textbox1), bg="#4CAF50", fg="white")
load_button1.grid(row=1, column=0, padx=5, pady=5, columnspan=2)

file_entry2 = tk.Entry(frame, width=50)
file_entry2.grid(row=2, column=2, columnspan=2, padx=5, pady=5)

load_button2 = tk.Button(frame, text="Load File 2", command=lambda:
load_file_or_display_contents(file_entry2, text_textbox2), bg="#4CAF50", fg="white")
load_button2.grid(row=2, column=0, padx=5, pady=5, columnspan=2)

compare_button = tk.Button(root, text="Compare", command=show_similarity, bg="#008CBA",
fg="white")
compare_button.pack(pady=5)

clear_button = tk.Button(root, text="Clear All", command=clear_all, bg="#D9534F", fg="white")
clear_button.pack(pady=5)

text_textbox_diff = tk.Text(root, wrap=tk.WORD, width=80, height=1)
text_textbox_diff.pack(padx=10, pady=10)

text_textbox1.tag_configure("same", foreground="red", background="lightyellow")
text_textbox2.tag_configure("same", foreground="red", background="lightyellow")

root.mainloop()
```

5. Input and Output Screens






Plagarism Checker
Text 1:
len(text2))
return similarity_ratio, diff_html
def generate_html_report(file1, file2, similarity_ratio, diff_html, report_path):
timestamp = datetime.datetime.now().strftime("%Y%m%d %H%M%S")
Text 2:
threshold)
return similarity_ratio, diff_html
def check_for_plagiarism(text1, text2, threshold=0.8):
d = difflib.HtmlDiff()
diff_html = d.make_file(text1.splitlines(), text2.splitlines())
Load File 1
D:/Work_Files/RCOEM/PythonLab/Practice/main.py
Load File 2
D:/Work_Files/RCOEM/PythonLab/Practice/main - Copy.
Compare
Clear All
Similarity: 85%

6. Conclusion

In summary, the "Plagiarism Checker" project achieves its goal of providing a user-friendly tool for text comparison. The Tkinter-based graphical interface ensures ease of use for individuals with varying technical skills. The underlying SequenceMatcher algorithm accurately calculates similarity ratios, with results visually presented through highlighted text in the Tkinter widgets.

A notable feature is the application's flexibility in handling input sources. Users can input text directly or load it from files using intuitive "Load File" buttons. Error handling ensures the program's stability, addressing issues like empty file paths.

As the project concludes, it successfully combines algorithmic precision and user-centric design, offering a practical tool for diverse applications, including academia and content creation. The project underscores the potential of technology to simplify complex tasks while emphasizing the importance of accessibility and user satisfaction in application development.