

Analysis of Algorithms

Spring 2020

Members Details

Group ID	CS311-G22
Registration Number of Group Members	2018-CS-124 2018-CS-131
Section	C

Project Details

<i>Project</i>	
Project Title	Plagiarism Checking Tool
Executive Summary	<p>Project is about plagiarism checking between two texts. Such that it checks, how much of the content is same. It gives you the same content of text, number of match words and the percentage of plagiarism. Text, to check plagiarism, may be enter by user (in two text boxes) or may user add in the form of two text files. if user want to add text directly than two boxes are there on one page, or if it wants to add files than there is another option of check plagiarism between files if user clicks on it, another page will open having two options of “add file1” and “add file2”. After entering text user click on “check plagiarism” button after clicking on that button a new page open which shows the result of plagiarism of that texts. Working principle is such that, code take one line from the 1st file and then compare that line with each line of second file and find the LCS (long common substring) if the LCS of that line is greater than LCS of that line with the new line of second file, it store that line in a separate string. In other words we store that line of second file with which LCS of each line is greater. Then we move to next line of 1st file and repeat the process for that new line. After the whole processing we get a result array which contain the matching words of file one and file two.</p> <p>Code also work for cpp files or text but conditions are different for text files and cpp files. “.cpp” file give the output of plagiarism on the bases of percentage of plagiarism. This is because many of us implement the same algorithm and use same name that are in algorithm and also key words like (for, while, if, else,..etc) would be same in the coding file, so we consider that if percentage is</p>

	more than 20% than text is copied or plagiarized.
Business Case	
Outline the business need for the project	In this modern area, plagiarism has become serious issue. Internet advent makes plagiarism easier. Many of the students and researchers use others works instead of utilizing their own abilities. So we have different strategies to prevent it, one of them is developing plagiarism checking tools. Also helpful for those who want to know or check that their work is not plagiarized.
End user of the product	<ul style="list-style-type: none"> • Students • Researchers • Teachers <p>Almost all students took help from the goggle to understand the concept and to learn new things. Student can use it to check that their work is not plagiarized. Researchers during research read different articles and then write their own research about that topic so they can use it to check that their paper is not matching to any of the research paper available on the internet. Students copy each other work so teachers also use it to punish them.</p>
Motivation for Project	<i>[This section to contain a clear statement of motivation which drives you to this project]</i>
Description of the project objective(s)	<p>➤</p> <p><i>[Identify the key objectives of the project]</i></p>
State the level of impact expected should the project proceed and implications of not proceeding	<i>[State whether the implementation would have an impact at an operational level and/or strategic level and state the impact(s) in 2-3 lines]</i>
Functional Requirements	<ul style="list-style-type: none"> ➤ We provide two options to user <ul style="list-style-type: none"> ➤ check plagiarism between files ➤ check plagiarism between text enter by user ➤ Sign Up functionality but not mandatory for the user to sign up before using the application ➤ User can Login if he/she sign up once. ➤ User can Logout

	<ul style="list-style-type: none">➤ Application will show the number of matched word➤ Application will show the matched text➤ Application will show the percentage of similarity between text																																								
Benefits																																									
What benefits are expected/ anticipated?	<ul style="list-style-type: none">• It gives the percentage of similler content• Useful for writing new content within short time• Improve our paraphrasing powers.• It is also very useful for web writers																																								
Implementation Details																																									
Link to Github Repository	https://github.com/Ayesha-Azam/CS311S20PID22																																								
Total Number of commits in repository before 5 th August 2020	26																																								
Exact contribution of each member	2018-CS-124 (13 commits) 2018-CS-131 (13 commits)																																								
Commits in github repository by each member																																									
<table><tr><th>Member Registration No.</th><th>Total Commits</th></tr><tr><td>2018-CS-124</td><td>13 commits</td></tr><tr><td>2018-CS-131</td><td>13 commits</td></tr></table>		Member Registration No.	Total Commits	2018-CS-124	13 commits	2018-CS-131	13 commits																																		
Member Registration No.	Total Commits																																								
2018-CS-124	13 commits																																								
2018-CS-131	13 commits																																								
Details of commits																																									
<table><tr><th>Sr. No.</th><th>Details of commit</th><th>Date</th><th>Member Reg No.</th></tr><tr><td>1</td><td>Added Sign Up, sign in and logout functionalities</td><td>2 days ago</td><td>2018-CS-131</td></tr><tr><td>2</td><td>Time complexity and analysis of algorithm</td><td>27 days ago</td><td>2018-CS-131</td></tr><tr><td>3</td><td>Made some changes in the UI</td><td>4 days ago</td><td>2018-CS-131</td></tr><tr><td>4</td><td>Pseudo Code</td><td>30 days ago</td><td>2018-CS-131</td></tr><tr><td>5</td><td>Plagiarism Checker Code File</td><td>25 days ago</td><td>2018-CS-131</td></tr><tr><td>6</td><td>Final Code For file handling</td><td>25 days ago</td><td>2018-CS-124</td></tr><tr><td>7</td><td>Rough working on pseudo code</td><td>28 days ago</td><td>2018-CS-124</td></tr><tr><td>8</td><td>Front End Design</td><td>18 days ago</td><td>2018-CS-131</td></tr><tr><td>9</td><td>Final Code File</td><td>22 days ago</td><td>2018-CS-131</td></tr></table>		Sr. No.	Details of commit	Date	Member Reg No.	1	Added Sign Up, sign in and logout functionalities	2 days ago	2018-CS-131	2	Time complexity and analysis of algorithm	27 days ago	2018-CS-131	3	Made some changes in the UI	4 days ago	2018-CS-131	4	Pseudo Code	30 days ago	2018-CS-131	5	Plagiarism Checker Code File	25 days ago	2018-CS-131	6	Final Code For file handling	25 days ago	2018-CS-124	7	Rough working on pseudo code	28 days ago	2018-CS-124	8	Front End Design	18 days ago	2018-CS-131	9	Final Code File	22 days ago	2018-CS-131
Sr. No.	Details of commit	Date	Member Reg No.																																						
1	Added Sign Up, sign in and logout functionalities	2 days ago	2018-CS-131																																						
2	Time complexity and analysis of algorithm	27 days ago	2018-CS-131																																						
3	Made some changes in the UI	4 days ago	2018-CS-131																																						
4	Pseudo Code	30 days ago	2018-CS-131																																						
5	Plagiarism Checker Code File	25 days ago	2018-CS-131																																						
6	Final Code For file handling	25 days ago	2018-CS-124																																						
7	Rough working on pseudo code	28 days ago	2018-CS-124																																						
8	Front End Design	18 days ago	2018-CS-131																																						
9	Final Code File	22 days ago	2018-CS-131																																						

10	Second page web design	11 days ago	2018-CS-124
11	Code implementation_3	22 days ago	2018-CS-124
12	Final Commit (plagiarismCheckingToolFinal.zip)	10 days ago	2018-CS-131
13	Final commit	10 days ago	2018-CS-124
Have you used built in algorithms or you have implemented yourself?		We use memorized version of LCS and for checking the matched words we design our own algorithm and implemented it.	
Formats of input		<p>Our designed web app takes input from the user into two different ways. One way is to enter the text in the text field. Our designed app have provide two options to user</p> <ul style="list-style-type: none">• Enter text by yourself• Upload files to check plagiarism <p>User can select one of these two. “Enter text By yourself” opens a new page having two columns for plagiarism checking. User will enter text and then click “Check Plagiarism” button.</p> <p>By clicking on “Upload files to check plagiarism” a new page will open having two browse file buttons. User will select files from its computer and upload it for checking the plagiarism.</p> <p><i>In which format, input will be given to your system? Provide complete details on input formats.</i></p>	
Validations		<p>We asked user to select one option from two. User can either upload files or enter text at one time. It is the validation that user can select one option at a time. We don’t apply validations of sign up for using our web app. It is the user’s choice if he/she wants to sign up.</p> <p><i>List the validations that you have applied on input with complete details</i></p>	
Format of output		<p>When end user click the “Check Plagiarism” button a new page open containing information about the files or the text entered by user. This opened page shows the percentage of matched words, matched words or statements and the length of the match words.</p> <p><i>In which format, output will be expected?</i></p>	
Deployment		<p>No.</p> <p><i>Have you deployed your project in any format? If yes, provide the details</i></p>	
Details of algorithms			

We used two algorithms one is to store the matched words and second is to convert the input into a format in which want it to passes to a function. We write the LCS(longest common substring) using memorization.

// pseudo code (we write it ourselves)

// pseudo code for checking the plagiarism between entered text

```

1. string1,string2
2. str1[],str2[]
3. s1,s2
4. while(!(length of string1))
5. s1=string1
6. i=0;
7. while(s1[i]!=".")
    i. str1[i]=s1[i]
    ii. i++
8. while(!(length of string2))
    i. s2=string2
    ii. j=1
    iii. while(s2[j]!=".")
        str2[j]=s[j]
        j++
    iv. temp=0;
    v. if((LongestCommonSubstring(str1,str2,lengthOfString1,
        lengthOfString2)>temp)
        temp=LongestCommonSubstring(str1,str2,lengthOfString1,
        lengthOfString2)
9. // file2 whole text being comapre with file1 line by line and the line which has the longest
    common substring with file1 line of file2 it will be stored in resultArray[]
10.         resultArray[]=str2;
11.         // empty str1[] str2[]
12.
```

Description: it will take two strings from the user and check plagiarism between them. While. loop will iterates until the length of string1. While loop of line 7 takes one statement of string1(statement ends at '.'). while loop at line 8 also tale the one statement of string 2 and then memorized LCS call on it and store the length in temp. in net iteration of while loop of line 8 it will take line 2 of string 2 and compare it with line1 of string1 and if the number of matched words between these lines is greater than the previous value of temp then replace them and check until string 2 ended and store the longest matched string in the resultArray, and so on

// this is the pseudo code for plagiarism checking tool , it will take 2 files

// this is the pseudo code for plagiarism checking tool , it will take 2 files

```

13. file1,file2
14. str1[],str2[]
15. s1,s2
16. while(!(end of file1))
17. file1<<s1
18. i=0;
19. while(s1[i]!=".")
    i. file1<<str1[i];
    ii. i++
20. while(!(end of file2))
    i. file2>>s2;
    ii. j=1
    iii. while(s2[j]!=".")
        file2<<str2[i]
        j++
    iv. temp=0;
    v. if((LongestCommonSubstring(str1,str2,lengthOfString1,
        lengthOfString2)>temp)
        temp=LongestCommonSubstring(str1,str2,lengthOfString1,
        lengthOfString2)
21. // file2 whole text being comapre with file1 line by line and the line which has the longest
    common substring with file1 line of file2 it will be stored in resultArray[]
19.         resultArray[]=str2;
20.         // empty str1[] str2[]

```

Description:

It will take two files as file1 and file2. While. loop will iterates until the end of file1. In line 4 we are reading the data from file and placing it in str1. In line 6 while loop terminates when it finds the .(end of statement). Line 7 while loop also doing the same, fetching one line and placing it in the str2. This way we get one statement of file1 and one statement of file2 we are then calling memorized LCS on these two statements and storing the length in temp. in next iteration it will read statement2 of file2 and call memorized LCS on it and if temp previous value is smaller than this time then store this value in the temp otherwise again takes next statement and start comparing it with the first statement of file1. We are trying to compare one line of file1 with whole file2 and the line whose matching length is highest we are considering it as plagiarized statement and storing it in the resultArray.

// this is the pseudo code for plagiarism checking tool , it will take 2 or more than 2 files

```

1. str1[],str2[]
2. ArrayOfFile[]           //it is dynamic array which contains the address of the files selected
    by user //e.g at 0 index file1 address is placed, at 1 file1 address lies and so on.
3. for t=0 to ArrayOfFile.length

```

```

4.   while(!(end of ArrayOfFile[t]))
5.       ArrayOfFile[t]<<str1
6.       i=0;
7.   while(s1[i]!=".")
8.       ArrayOfFile[t]<<str1[i];
9.       i++
10.  while(!(end of ArrayOfFile[t]))
11.      ArrayOfFile[t+1]>>str2[0];
12.      j=1
13.      while(s2[j]!=".")
14.          ArrayOfFile[t+1]<<str2[j]
15.          j++
16.      temp=0;
17.      if((LongestCommonSubstring(str1,str2,lengthOfString1, lengthOfString2)>temp)
18.          temp=LongestCommonSubstring(str1,str2,lengthOfString1, lengthOfString2)
19.      // ArrayOfFile[t+1] whole text being compared with ArrayOfFile[t] line by line and
      the line which has the longest common substring with ArrayOfFile[t] line of ArrayOfFile[t+1]
      it will be stored in resultArray[]
20.      resultArray[]=str2;
21.      // empty str1[] str2[]

```

Description:

It will take two or more than two files. Array of file is the file containing the pointers. Each pointer points towards a file. For example, ArrayOfFile[0] holding the address of file1 and so on. For the first iteration it will take file1 and file2 and perform the following processing.

While loop will iterate until the end of file1. In line 4 we are reading the data from file and placing it in str1. In line 6 while loop terminates when it finds the .(end of statement). Line 7 while loop also doing the same, fetching one line and placing it in the str2. This way we get one statement of file1 and one statement of file2 we are then calling memorized LCS on these two statements and storing the length in temp. in next iteration it will read statement2 of file2 and call memorized LCS on it and if temp previous value is smaller than this time then store this value in the temp otherwise again takes next statement and start comparing it with the first statement of file1. We are trying to compare one line of file1 with whole file2 and the line whose matching length is highest we are considering it as plagiarized statement and storing it in the resultArray. One file1 reaches to its end the most outer while loop terminates and then for loop increments and the same procedure continues again for the file2 and file3.

Complexity Analysis:

1. str1[],str2[]	-----1)
2. ArrayOfFile[]	-----1)
3. for t=0 to ArrayOfFile.length	----- (numberOfFiles)
4. while(!(end of file[t]))	----- (n+1)
5. ArrayOfFile[t]<<str[0]	----- (n)
6. i=0;	----- (n)
7. while(str1[i]!=".")	----- n/li+1)

```

8.      ArrayOfFile[t]<<str1[i]; -----(n/li)
9.      i++ -----(n/li)
10.     while(!(end of file2)) -----(n+1)
11.      ArrayOfFile[t+1]>>str2[0]; -----(n)
12.      j=1 -----(n)
13.      while(str2[j]!=0) -----
(n/li+1)
14.      ArrayOfFile[t+1]<<str2[i] -----(n/li)
15.      j++ -----
(n/li)
16.      temp=0; -----(n)
17.      if((LongestCommonSubstring(str1,str2,lengthOfString1,
lengthOfString2)>temp)
18.      temp=LongestCommonSubstring(str1,str2,lengthOfString1,
lengthOfString2) -----(n)
19.      resultArray[]=str2; -----(n)

```

Here loop in the line 3 iterates equal to length of the number of files user enter. Like if user enter 3 it will executes 3 times. Loop in the lines number 4 iterates n+1 times.

li is equal to the number of lines or number of statements the file has.

Now,

Running time is $=1+1+(n+1)+n+n+(n+1)*n/li+n/li+n/li+(n+1)*(n+1)+n+(n+1)*n/li+n/li+n/li+n(LCS\ time)+n+n$

LCS is taking $O(n)$ time proved later in this file.

Put it into the equation we are solving for finding the running time of algorithm.

Running time is $=1+1+(n+1)+n+n+(n+1)*n/li+n/li+n/li+(n+1)*(n+1)+n+(n+1)*n/li+n/li+n/li+n(LCS\ time)+n+n$

$=1+1+(n+1)+n+n+(n+1)*n/li+n/li+n/li+(n+1)*(n+1)+n+(n+1)*n/li+n/li+n/li+n(n)+n+n$

$=2+n+1+2n+n^2/li+n/li+2n/li+n^2+2n+1+n^2/li+n/li+2n/li+n^2+2n$

$=4+7n+6n/li+2n^2/li+2n^2$

$=O(n^2)$

//pseudo code for memorized LCS

1. constant MAX_SIZE=10000


```

2.  resultArray[MAX_SIZE]           //make it dynamic in our code
3.  memoizedArray[MAX_SIZE][MAX_SIZE]
4.  //Longest Common Substrings
5.  function LongestCommonSubString(string string1[],string string2[], lengthOfString1,
    lengthOfString2)

6.      if(lengthOfString1==0 OR lengthOfString2==0)

7.          return 0

8.      //if the data is already in the table then return it instead of recomputing it
9.      if(memoizedArray[m-1][n-1]!=-1)           //for this place -1 in the whole 2d array

10.         return memoizedArray[m-1][n-1]

11.     //case 1: if matches
12.     if(string1[m-1]==string2[n-1])

13.         return 1+LongestCommonSubString(string1,string2, lengthOfString1-1,
    lengthOfString2-1)

14.     //if do not match
15.     else
16.         return max(LongestCommonSubString(string1,string2, lengthOfString1-1,
    lengthOfString2),LongestCommonSubsequence(string1,string2, lengthOfString1,
    lengthOfString2-1))

```

Description: This algorithm takes two strings and their length. It has 3 cases. If string1's or string2's length is equal to zero then there is no need to check for plagiarism simply return 0. We fill the memoizedArray with -1 and if the memorizedArray at that particular index != -1 then it will definitely have some value and it will simply return it.

Otherwise case2 comes in which starts matching from the end if last character matches then again call the same function for second last character +1. We are adding because last character matches. And if none of the above conditions are true the else case will run which will call itself for length(str1-1) and then with length(str2-1). And return the max of these 2 return values. This is a recursive function with memorization. Following is the complexity analysis of this algorithm.

Complexity Analysis:

```

constant MAX_SIZE=10000           -----(1)

resultArray[MAX_SIZE]             -----(1)

memoizedArray[MAX_SIZE][MAX_SIZE] -----(1)

function LongestCommonSubString(string string1[],string string2[], lengthOfString1, lengthOfString2)

```

```

if(lengthOfString1==0 OR lengthOfString2==0) -----(1)
    return 0

//if the data is already in the table then return it instead of recomputing it
if(memoizedArray[m-1][n-1]!=-1) -----(1)
    return memoizedArray[m-1][n-1]

//case 1: if matches
if(string1[m-1]==string2[n-1])

lengthOfString2-1)    return 1+LongestCommonSubstring(string1,string2, lengthOfString1-1,
----- (i)

//if do not match
else
    return max(LongestCommonSubstring(string1,string2, lengthOfString1-1,
lengthOfString2), ----- (ii)
LongestCommonSubsequence(string1,string2, lengthOfString1, lengthOfString2-1))

```

Recurrence equation for equation (i) is:

$$T(n) = \begin{cases} 0 \\ T(n1-1, n2-1) + 1 \\ \max(T(n1-1, n2), T(n1, n2-1)) + 1 \end{cases}$$

it is 0 when n1=0 or n2=0 it is T(n1-1, n2-2) + 1 when matches max(T(n1-1, n2), T(n1, n2-1)) + 1 when do not matches.

T(n1-1, n2-1) + 1

↓

T(n1-2, n2-2) + 2

↓

T(n1-3, n2-3) + 3

↓

⋮

↓

T(n1-k, n2-k) + k

Assume, n1-k=0 => n1=k similarly, n2-k=0 => n2=k

As k=n1 as well as k=n2 so let k=n

So, $T(n1-n1, n2-n2)+n \rightarrow T(0) + n \rightarrow$ we know $T(0) = 0$ which is base case so is it n.
Do same for other recursive equation.

$$T(n1-1, n2)+1$$

$$\downarrow$$

$$T(n1-2, n2)+2$$

$$\downarrow$$

$$T(n1-3, n2)+3$$

$$\downarrow$$

$$\vdots$$

$$\vdots$$

$$\vdots$$

$$\downarrow$$

$$T(n1-k, n2)+k$$

Assume, $n1-k=0 \Rightarrow n1=k$

As $k=n1$ as well as $k=n2$ so let $k=n$

So, $T(n1-n1, n2)+n \rightarrow T(0, n2) + n \rightarrow$ we know $T(0) = 0$ which is base case so is it n.

$$T(n1, n2-1)+1$$

$$\downarrow$$

$$T(n1, n2-2)+2$$

$$\downarrow$$

$$T(n1, n2-3)+3$$

$$\downarrow$$

$$\vdots$$

$$\vdots$$

$$\vdots$$

$$\downarrow$$

$$T(n1, n2-k)+k$$

Assume, $n2-k=0 \Rightarrow n2=k$

As $k=n1$ as well as $k=n2$ so let $k=n$

So, $T(n1, n2-n2)+n \rightarrow T(n1, 0) + n \rightarrow$ we know $T(0) = 0$ which is base case so is it n.

So

$$1+1+1+1+1+n+n \rightarrow 5+2n \rightarrow O(n)$$

Correctness of algorithm:

Correctness of algorithm can be shown using contradiction and inductive hypothesis. But we will show that our algorithm is correct using loop invariant. For this we will claim that before and after each iteration of loop `it(result array)` has the longest common matched substring.

Initialization:

Before the first iteration, `lcs` was not called and `temp = 0` which means number of matched words is 0. Which is the correct answer.

Maintenance:

Line one of `file1` will be selected and `lcs` was called on line 1 of `file1` with each line of `file 2` one by one. Simply, line1 of `file` matches with line1 of `file 2` then matched words length stored in `temp`. again line 1 of `file 1` matches with line 2 of `file2` and match words length store in `temp` if it is greater than previous value of `temp`. this procedure continues until the `file2` ends. Hence the line of `file2` which matches with greatest matched words with line 1 of `file1` then stored in `resultArray`. Here loop invariant maintains because after the first iteration `result array` has the longest common words from both files.

Termination:

As we explained in maintenance how our algo works for one line of file 1. It will do the same for each and every line of file1. In the end result array wil have al the longest common substrings or words.

Lets suppose two files.

File1:	File2:
Lorem ipsum is simply dummy text. There are many variations of passages available.	Lorem ipsum is not simply random text. Here are some variations of pages available.

In the beginning temp=0 because no files text gets compare with other. And this is correct result. Then our algorithm will take line of file which is “lorem ipsum is simply dummy text” and take line no 1 of file 2 which is “lorem ipsum is not simply random text” and then apply lcs on both of these. It returns 5 as 5 strings are common and store this value in temp. in the next iteration it took second line of file2 which is ‘here are some variations of pages available’ and apply lcn on them. It returns 0 as no string is common. And this inner loop terminates and result array has “lorem ipsum is simply text”. After first iteration outer loop takes line 2 of file and coampre it with whole file2 and store 5 in temp and finally the outer loop also terminates as file2 ended.

In the result array has “ lorem ipsum is simply text are variations of passages available”. And length =10.

So our algorithm is correct.

//Bonus Task Cpp Plagiarism Checking (python code)

function CPPdataReadingAndLCS(file1, file2)

```

global resultArray
global f1, f2
global res
char_arr1 []
char_arr2 []
string1 []
string2 []
f1 = file1.read()
f2 = file2.read()
l1 = length(f1)
l2 = length(f2)
for i in range(0, l1):
    temp=0
    if f1[i] != ';':
        char_arr1.append(f1[i])
    else:
        string1 = ""

```

```

for x in char_arr1:
    string1 += x
for f in range(0, len(char_arr1)):
    if (char_arr1[f] == ' '):
        char_arr1[f] = '\0'
for j in range(0, l2):

    if f2[j] != ';':
        char_arr2.append(f2[j])
    else:
        string2 = ""
        for y in char_arr2:
            string2 += y
        #temp = 0
        #for f in range(0, len(char_arr1)):
        #    if (char_arr1[f] == ' '):
        #        char_arr1[f] = '\0'

        for f in range(0, len(char_arr2)):
            if (char_arr2[f] == ' '):
                char_arr2[f] = '\0'
        s=LongestommonSubstring(char_arr1,char_arr2,len(char_arr1),len(char_arr2))
        if(s>temp):
            temp =s
            char_arr2 = []
            char_arr1 = []
        resultArray = string2
        #print(string2, end=" ")
        res += temp
main Function:
memoizedArray = [[-1 for i in range(MAX_SIZE)]
    for i in range(length(file1))]
CPPdataReadingAndLCS(file1, file2)
print("matched words are: ")
print(resultArray)
numerator=0
if(p1<p2):
    numerator=p2
elif(p1==p2):
    numerator=p1
else:
    numerator=p1
print(res)
percent=(res/numerator)*100
if(percent>30):
    print("these are plagiarised files")

```

else:

```
print("this is not plagiarised text")  
print(percent)
```

Description:

This is the code for plagiarism checking between cpp files. It will take two cpp files. We are using “;” as end of one statement. It takes one statement of file1 and one statement of file2 and apply memorized LCS on them. Then it will take second statement of file2 and apply memorized LCS on it. Basically it comparing one statement of file1 and with whole file2 and storing the line which has maximum matched words. Then in main file we are setting numerator equals to the length of file having maximum length and diving the match words length by numerator and multiplying the result with 100 for calculating the percentage. As keywords of the language are same in all files and mostly programmers use I,j,k,x,yz as loop variables so we are setting a margin of 30% to 20% as not plagiarized text.

Interfaces for your project



This is the front page of our web application user have two options. User will select one of them. If user selects “Between text enter by user in text area” the following Page will opens:

Plagiarism Checker

Enter some text:

Enter some text:

Check Plagiarism

Here user will enter its text and click the check plagiarism button which will open the following page showing the information including matched words, length of matched words and percentage of plagiarized text.

Result page

But if user select the option of “between text stored in files”. The following page will open.

Plagiarism Checker

Upload File:

Choose File No file chosen

Choose File No file chosen

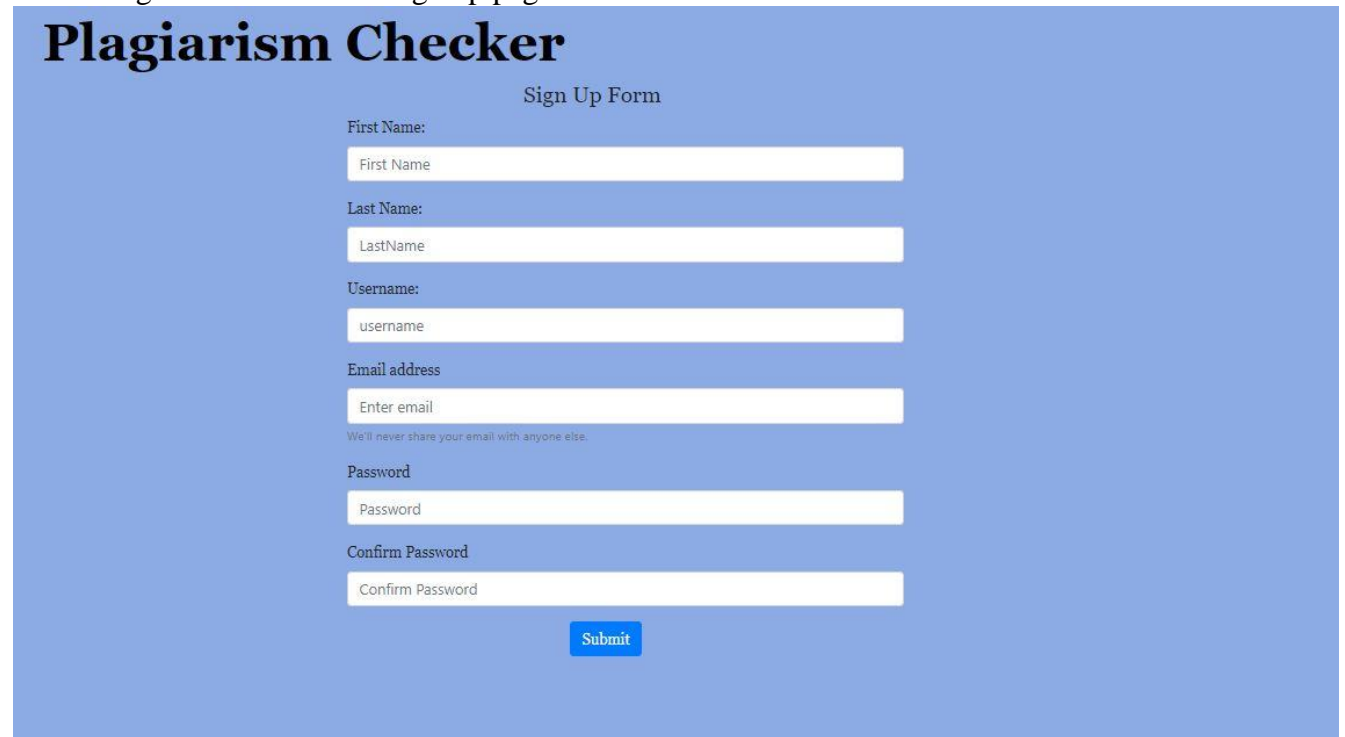
Check Plagiarism

User will upload files and click on the “check plagiarism button”. When click on the button a new page will open showing the information including matched words, length of matched words and percentage

of plagiarized text.

We also integrate our website with database for login and sign up functions. But it is not mandatory for the user to sign up before using the website.

Following is the interface of sign up page which is connect to a database.



Plagiarism Checker

Sign Up Form

First Name:

Last Name:

Username:

Email address

We'll never share your email with anyone else.

Password

Confirm Password

We are using postgresql and pgAdmin for using the GUI database interface. We migrate the data to a databases that we get from this page. User data is store in the database by the next time he/she will only use login button and login into the application. We set some validations like no two user can have the same username, no two user can have the same email. For login user will click on the login button shown in the first page of website.

We added one more functionality. If the username that is user trying to enter during sign up process is already present in the database then it will generate an error message “user name already present.” in the red color. Similarly for email it will show tha error message “email already exists” if entered eail present in the database.

Plagiarism Checker

Login

Username:

Password

User will enter the username and password when click on the submit button this information is compare with the stored data in database if it matches user can successfully login.

We also added the logout functionality in the application but due to an error logout page is not opening so we do not add it in our report.

Integration

We faced too many problems when we are integrating our implemented python code with our UI. Since we are using django for this purpose. We have not used it or learned it before. We took some help from the google and watch many tutorials first and then start integrating it. Creating UI is not a headache we designed our pages in a very short time but integration part took a lot of time. We faced many errors and all of them are new to us but we try our best to solve them, but still we have an error of “MultiValueDictKeyError”. We try to resolve it but unfortunately we are not succeeded. Due to this single error we are not able to check the proper functionality of our sign up and sign in pages as well as for sign out, but we are hoping that it works well when this error resolve.

During the implementation of algorithm as we write it ourselves we also face some logical errors and we resolved many of them. Due to the some other workloads and working from home in this pandemic it is also difficult for us to share our ideas clearly. But we put our best in this project.

Change Requests

Yes, we made some changes in the UI. We proposed the follwing interface in the deadlines.



Plagiarism Checker

Enter Text to check plagiarism:

Enter Text to check plagiarism:

OR

Upload File:

Choose File

No File Chosen

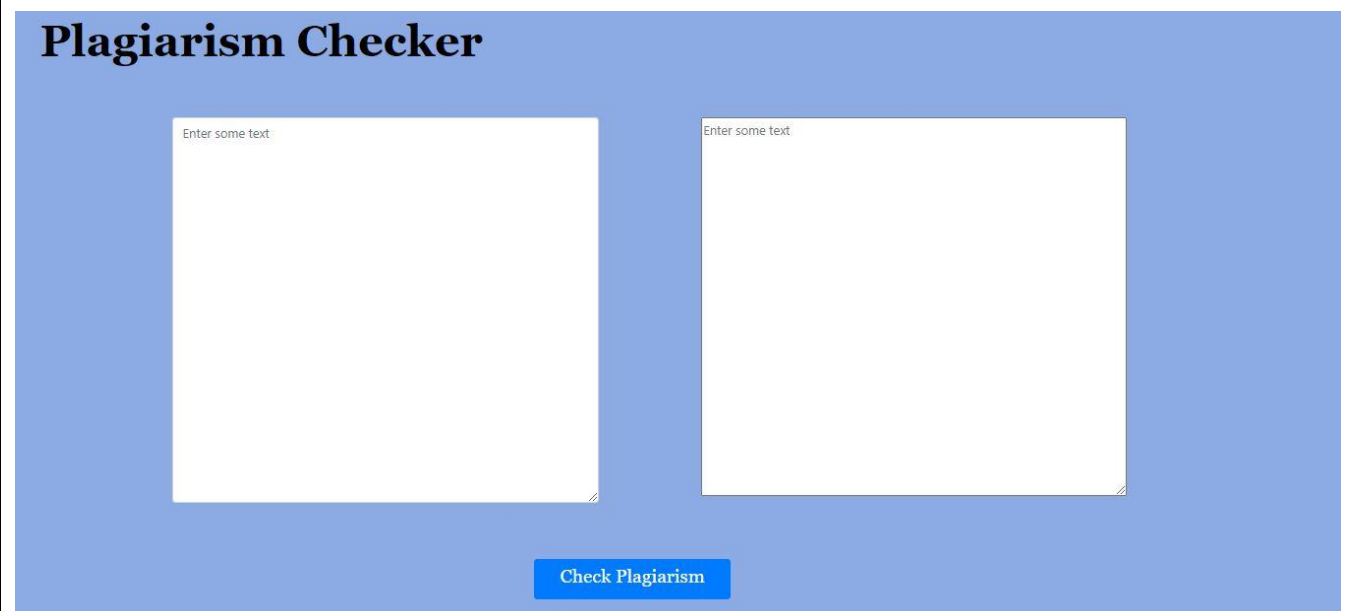
Check Plagiarism

In this UI design we were decided to add both options in a single page and we did not decided to add login, sign up and logout functionality but later due to the date extension of submitting the project we

decided to add it. Above UI is our proposed UI and we submitted it in the deadlines but now we changed it and make separate pages for both option(file upload, enter text). We change it and convert it into the following designs.



This is a UI design for a Plagiarism Checker. It features a blue header with the title "Plagiarism Checker" on the left and "Sign Up Login" on the right. Below the header, there are two light blue horizontal bars. The first bar contains the text "Between text stored in Files" and the second bar contains the text "Between text enter by user in text area". The main body of the page is a solid blue color.



This is a UI design for a Plagiarism Checker. It features a blue header with the title "Plagiarism Checker". Below the header, there are two white text input areas, each with the placeholder text "Enter some text:". At the bottom of the page, there is a blue button with the text "Check Plagiarism".

Plagiarism Checker

Upload File:

Choose File No file chosen

Choose File No file chosen

Check Plagiarism

Plagiarism Checker

Sign Up Form

First Name:

First Name

Last Name:

LastName

Username:

username

Email address

Enter email

We'll never share your email with anyone else.

Password

Password

Confirm Password

Confirm Password

Submit

Plagiarism Checker

Login

Username:

Password

Testing

In this section, you are required to mention the issues report and solution proposed.

Technology

Programming Language

Python

Platform

Web Application