Assignment No : B2

1 Title :

Plagiarism Detection

2 Problem Statement :

Write a web application using Scala/ Python/ Java /HTML5 to check the plagiarism in the given text paragraph written/ copied in the text box. Give software Modeling, Design, UML and Test cases for the same using Analysis Modeling (Static Modeling, Object Structuring, Dynamic Modeling).

3 Learning Objectives :

1. Understanding concepts of Plagiarism Detection
2. Learn how to implement Plagiarism detection algorithms Web application development.

4 Software and Hardware Requirement :

1. 64 bit Machine i3/i5/i7
2. 64-bit open source Linux OS Fedora 20
3. Text editor - gedit
4. Java

5 Theory:

5.1 Plagiarism Detection:

Plagiarism detection is the process of locating instances of plagiarism within a work or document. The widespread use of computers and the advent of the Internet has made it easier to plagiarize the work of others. Most cases of plagiarism are found in academia, where documents are typically essays or reports. However, plagiarism can be found in virtually any field, including scientific papers, art designs, and source code.

Detection of plagiarism can be either manual or software-assisted. Manual detection requires substantial effort and excellent memory, and is impractical in cases where too many documents must be compared, or original documents are not available for comparison. Software-assisted detection allows vast collections of documents to be compared to each other, making successful detection much more likely.

The practice of plagiarizing by use of sufficient word substitutions to elude detection software is known as rogeting.

Systems for text-plagiarism detection implement one of two generic detection approaches, one being external, the other being intrinsic. External detection systems compare a suspicious document with a reference collection, which is a set of documents assumed to be genuine. Based on a chosen document model and predefined similarity criteria, the detection task is to retrieve all documents that contain text that is similar to a degree above a chosen threshold to text in the suspicious document. Intrinsic PDS solely analyze the text to be evaluated without performing comparisons to external documents. This approach aims to recognize changes in the unique writing style of an author as an indicator for potential plagiarism. PDS are not capable of reliably identifying plagiarism without human judgment. Similarities are computed with the help of predefined document models and might represent false positives.

6 Testing:

6.1 White Testing:

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).
In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). White-box

testing can be applied at the unit, integration and system levels of the software testing process.
White-box test design techniques include the following code coverage criteria:

Control flow testing

Data flow testing

Branch testing

Statement coverage

Decision coverage

Modified condition/decision coverage

Prime path testing

Path testing

6.2 Positive Testing:

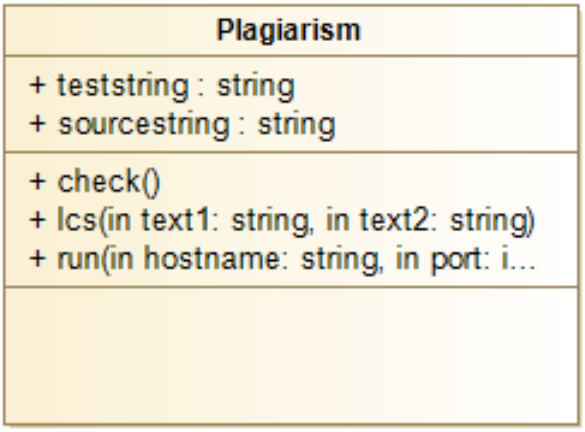
Positive Testing is testing process where the system validated against the valid input data. In this testing tester always check for only valid set of values and check if a application behaves as expected with its expected inputs. The main intention of this testing is to check whether software application not showing error when not supposed to and showing error when supposed to.
The system will show the appropriate output when it is tested against correct mathematical values. For example, for trigonometric functions, the system will show appropriate output in floating point numbers. Also for basic mathematical operations like addition, subtraction, it will show appropriate output.

6.3 Negative Testing:

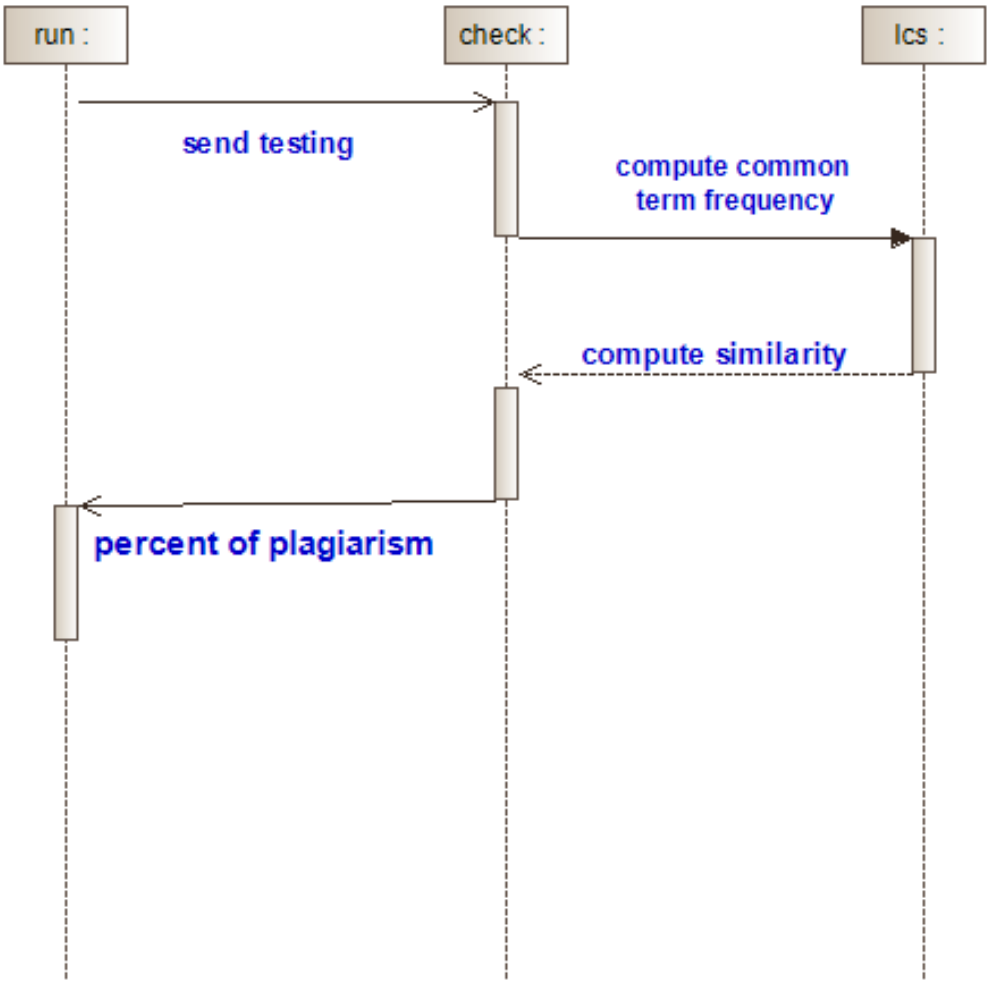
Negative Testing is testing process where the system validated against the invalid input data. A negative test checks if a application behaves as expected with its negative inputs. The main intention of this testing is to check whether software application not showing error when supposed to and showing error when not supposed to.
While performing the negative testing, the system will show incorrect output while performing the basic arithmetic operations if the user enters only a single number and performs addition without entering the second number. The system will give an error when the user attempts a divide by zero operation.

7 UML Diagrams:

7.1 Class Diagram

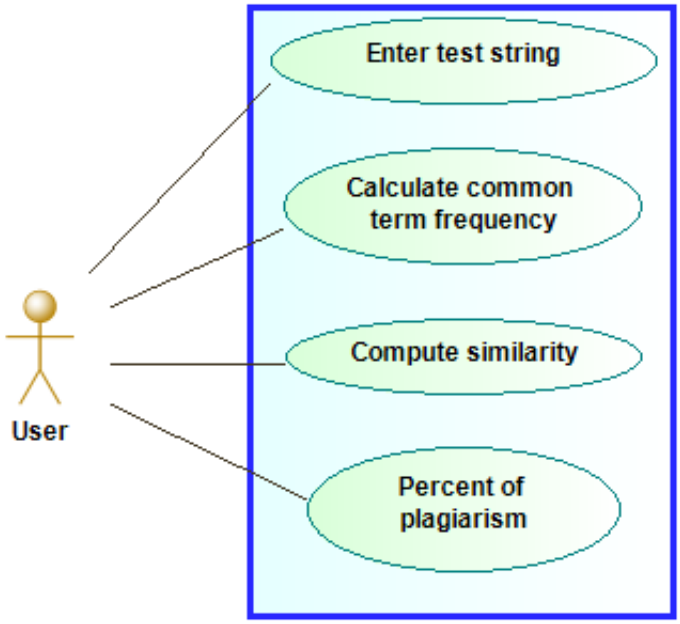


7.2 Sequence Diagram:



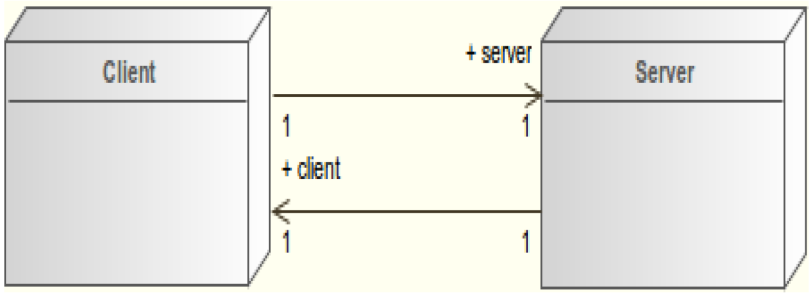
7.3

Use Case Diagram:

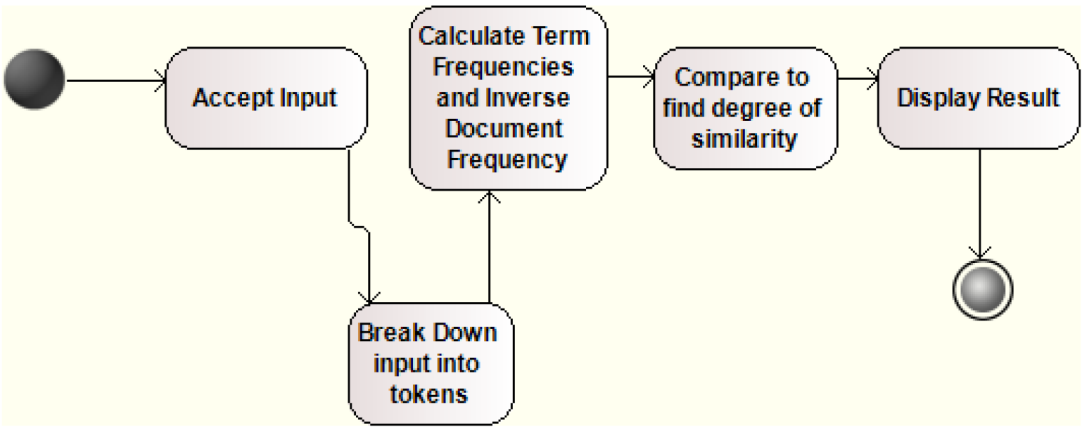


7.4

Diagram:



7.5 State Diagram:



8 Mathematical Model:

Let S be the system of solution set for given problem statement such that,
S={ s,e,X,Y,F,DD,NDD,Su,Fu }
where,
s= Start State, where Y={}

e= End State, where Y={Y1|Y2}
where, Y1= Plagiarism Detected , Y2= Plagiarism Not Detected

X= Input Set,
X={X1,X2}
where,
X1= Predefined File with which comparison takes place
X2= User Input Text

Y= Output Set,
Y={Y1|Y2}
where, Y1= Plagiarism Detected , Y2= Plagiarism Not Detected

DD= Deterministic Data Set
DD={User Input is same or different than the predefined file}

NDD= Non Deterministic Data Set
NDD={User Input is null || File content is null}

Su= Success Case

{User Input is not empty & File is not empty}

Fu= Failure Case
{Either User input of File is Empty}

F= Set of functions
F= {F1,F2,F3}
where,
F1= Accept User Input
F2=Compare with File
F3=Display Output

9 Program Code:

B2.py :

```
from flask import Flask , request, render_template

app = Flask(__name__)
#Flask constructor takes the name of current module (__name__) as argument.

@app.route('/')
#The route() function of the Flask class is a decorator, which tells the
application which URL should call the associated function.

def fun():
    return render_template('h.html',msg="")
    #render the template without any message

@app.route('/',methods=['POST'])
#this url calls the checker function for evaluating plagarism

def check():
    a = checker(request.form['string'])
    return render_template('h.html',msg=a)
    #render the template with the plagarism percentage

def checker(str1):
    file_data=""
    with open('data.txt','rt') as f:
        file_data = f.read()

    # remove unwanted characters from both strings
    unwanted_chars = ":?.-!_/\,;"
    for char in unwanted_chars:
```

```
        file_data = file_data.replace(char,"")
        #remove the inwanted characters from the data and the knowledge
        str1 = str1.replace(char,"")

    a = file_data.split()
    print a
    b = str1.split()
    print b

    copy_count = 0

    for i in b:
        if i in a:
            copy_count = copy_count + 1

    print "copy_count = ", copy_count
    percentage = str(float(copy_count)/len(a)*100.0) + "%"
    return percentage

if __name__=="__main__":
    app.run()
```

data.txt:

```
Hello world. This is my first program. I study in PICT.
```

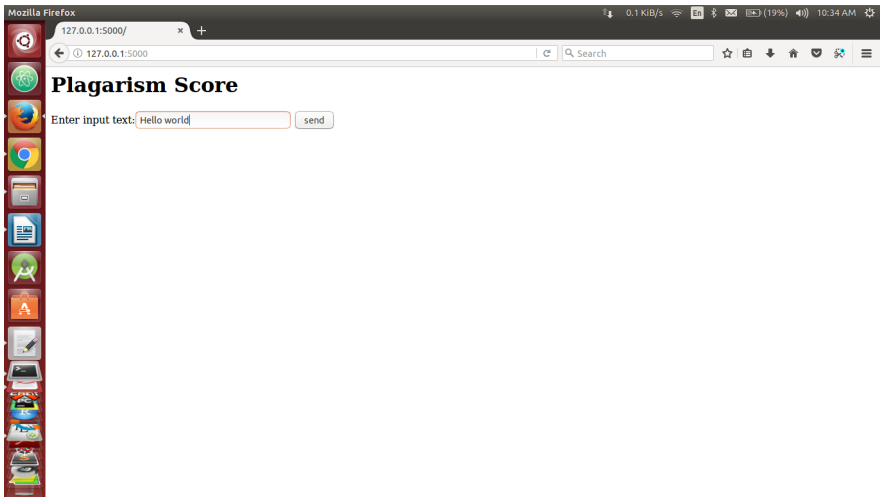
h.html:

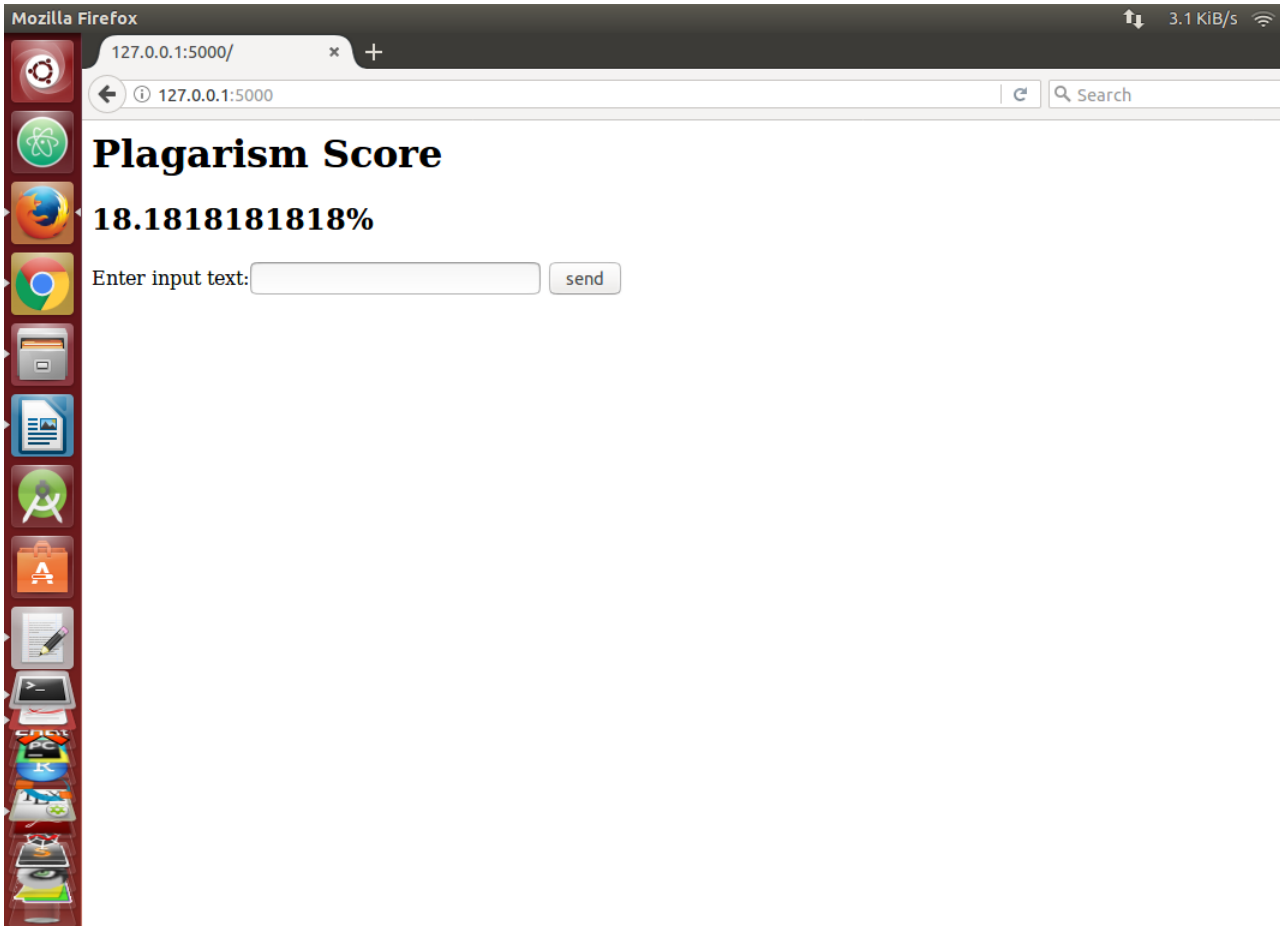
```
<html>
<h1>Plagarism Score</h1>
<h2>{{msg}}</h2>
<body>
<form action='.' method='POST'>
Enter input text:<input type="text" name="string">
<input type="submit" value="send">
</form>
</body>
</html>
```

10 Output:

```
shivani@shivani-Inspiron-3543:~/Documents/BE/CL3/4265/B2$ python B2.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [05/Apr/2018 10:34:08] "GET / HTTP/1.1" 200 -
```

```
127.0.0.1 - - [05/Apr/2018 10:34:09] "GET /favicon.ico HTTP/1.1" 404 -
['Hello', 'world', 'This', 'is', 'my', 'first', 'program', 'I', 'study', 'in', 'PICT']
[u'Hello', u'world']
copy_count =  2
127.0.0.1 - - [05/Apr/2018 10:34:59] "POST / HTTP/1.1" 200 -
```





11 Conclusion:

Thus, we have successfully implemented a plagiarism checker web application in Python using Flask. Here we are checking the user input with different files in the database and concluding whether the given text input is plagiarised or not.