Plagiarism Checker X Originality Report

Similarity Found: 7%

Date: Thursday, March 30, 2023
Statistics: 372 words Plagiarized / 5580 Total words
Remarks: Low Plagiarism Detected - Your Document needs Optional Improvement.

-

index.html <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta http-equiv="X-UA-Compatible" content="IE=edge"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Plagiarsm Detection</title> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous"> <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.css"> </head> <body style="padding-bottom: 200px;"> <nav class="navbar navbar-expand-lg bg-dark"> Plagiarism Detection using ML Compare F2F </nav> <h1 align="center" style="margin-top: 20px; margin-bottom: 50px;"> Plagiarsm Detection using ML</h1> <form action="/" method=post enctype=multipart/form-data>   <textarea name="text" class="form-control rounded-top shadow bg-white rounded" id="exampleFormControlTextarea1" rows="10" placeholder="Input your text here..."

```
style="width:
600px;"></textarea>                </div>            </div>            </div>          <div
class="d-flex justify-content-center">            <input name="file"
accept=".doc, .docx,.txt,.pdf" class="form-control form-control-md h-25 d-flex
justify-content-center shadow bg-white" id="formFileLg" type="file"
style="border-radius: 10px; width: 270px; margin-top: 20px; margin-bottom:
20px;">          </div>          <div class="d-flex
justify-content-center">              <button type="Submit"
formaction="/plagiarism/<name>" class="btn btn-dark" style="border-radius:
15px; margin-right: 20px; height: 45px;"><i class="bi bi-search"></i> Check
Plagiarism</button>              <button type="Submit" value="Send "
formaction="/report" class="btn btn-dark" style="border-radius: 15px;
margin-right: 20px; height: 45px; text-align: center;"><i class="bi
bi-file-earmark-text"></i> Show
Report</button>          </div>        </form>          <div id="hasil"
style="margin-top: 80px;">              <h1 align="center">Result</h1>              <div
style="padding-bottom: 30px" class="container">                  <div class="col
d-flex justify-content-center">                      <div class="p-3 border bg-dark
text-light text-center" style="width: 180px; border-radius: 15px;">Plagiarsm :
{{ hasil_persen }}</div>                  </div>            </div>        </div>            <div
class="container">                  {% for i in range(0,
hasil_link|length) %}                  <div class="col d-flex
justify-content-center">                      <div class="p-3 border bg-dark text-light
text-start w-75" style="border-radius: 15px; margin-bottom: 5px;"><a
style="text-decoration: none; color: white"
href="{{ link_output[i] }}">{{ hasil_plagiarism[i] }}</a></div>                  </div>
            {% endfor %}            </div>          </div>  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js
"
integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+
mo//f6V8Qbsw3" crossorigin="anonymous"></script>  </body>  </html>
report.html <!DOCTYPE html>  <html lang="en">      <head>        <meta
charset="UTF-8">        <meta http-equiv="X-UA-Compatible"
content="IE=edge">        <meta name="viewport" content="width=device-width,
initial-scale=1.0">        <title>Plagiarsm Report1</title>        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3
URy9Bv1WTRi" crossorigin="anonymous">        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.c
```

```
ss"> </head> <style> table, th, td {  border: 1px solid black;  border-collapse: collapse; } </style> <body style="padding-bottom: 200px;">  <nav class="navbar navbar-expand-lg bg-dark">  <div class="container-fluid">  <a class="navbar-brand text-light">Plagiarsm Report3</a>  <a class="navbar-brand text-light" href="/">Plagiarism Detection using ML</a>  </div>  </nav>  <br>  <br>  <br> <div class="container">  {{ PWM_value | safe }} </div> <br> <br> <br> <button class="btn btn-dark" align="left" align="center"style="margin-left: 100px; border-radius: 15px; height: 45px;" onclick="window.print(); return false">Print this page</button> </body> <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p" crossorigin="anonymous"></script> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF" crossorigin="anonymous"></script> </html> f2f.html
<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Plagiarsm Detection</title>  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='styles_f2f.css') }}">  <link rel="stylesheet" type="text/css" href="styles_f2f.css">  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.css"> </head> <nav class="navbar navbar-expand-lg bg-dark">  <div class="container-fluid">  <a class="navbar-brand text-light">Compare F2F</a>  <a class="navbar-brand text-light" href="/">Plagiarism Detection using ML</a>  </div>  </nav> <body style="padding-bottom: 200px;">  <div class="inline-div">  <h2 align="center" style=" margin-left: 17px; margin-top: 20px; margin-bottom: 50px;"> Source</h2>  <form method=post enctype=multipart/form-data>  <div class="form-group">  <div class="d-flex justify-content-center">  <textarea name="root_text_from_html"
```

```
class="form-control rounded-top shadow bg-white rounded"
id="textArea1"                    rows="10" placeholder="Input your text here..."

style="width: 525px; margin-left:
55px;"></textarea>                </div>            </div>          <div class="d-flex
justify-content-center">             <input name="root_file_from_html"
accept=".doc, .docx,.txt,.pdf" class="form-control form-control-md h-25 d-flex
justify-content-center shadow bg-white" id="formFileLg1" type="file"
style="border-radius: 10px; margin-left: 50.5px; width: 270px; margin-top: 20px;
margin-bottom: 20px;">           </div>           </div>           <div
class="inline-div">           <h2 align="center" style=" margin-right: -100px;
margin-top: 20px; margin-bottom: 50px;"> Target</h2>              <div
class="form-group">                <div class="d-flex
justify-content-center">                    <textarea name="plag_text_from_html"
class="form-control rounded-top shadow bg-white rounded"
id="textArea2"                    rows="10" placeholder="Input your text here..."

style="width: 525px; margin-left:
96px;"></textarea>                </div>            </div>          <div class="d-flex
justify-content-center">             <input name="plag_file_from_html"
accept=".doc, .docx,.txt,.pdf" class="form-control form-control-md h-25 d-flex
justify-content-center shadow bg-white" id="formFileLg2" type="file"
style="border-radius: 10px; margin-right: -125.5px; width: 270px; margin-top:
20px; margin-bottom:
20px;">           </div>           </div>           <br><br><br><br>           <div
class="d-flex justify-content-center">             <button type="Submit"
formaction="/f2f_kmp/onRunF2F" class="btn btn-dark" style="border-radius:
15px; width: 220px; margin-right: 20px; height: 45px;"><i class="bi
bi-search"></i> Check
Plagiarism</button>           </div>           </form>           <div style="margin-top:
80px;">             <h1 align="center">Result</h1>             <div
style="padding-bottom: 30px" class="container">                  <div class="col
d-flex justify-content-center">                      <div class="p-3 border bg-dark
text-light text-center" style="width: 180px; border-radius: 15px;">Plagiarsm :
{{ F2F_value |
safe   }}</div>                </div>            </div>       </div>          </div> <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js
"
integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+
mo//f6V8Qbsw3" crossorigin="anonymous"></script> </body> </html> App.py
```

```
from flask import Flask, render_template, request, redirect, url_for, flash from
werkzeug.utils import secure_filename from requests.adapters import
HTTPAdapter from urllib3.util.retry import Retry from bs4 import BeautifulSoup as
bs from difflib import SequenceMatcher from googlesearch import search import
pandas as pd import requests import warnings import PyPDF2 import nltk import
re import os class MyClass:     x = 5 warnings.filterwarnings("ignore",
module='bs4') # Search function # The function is used for searching the content
over the web.

def searchBing(query, num):       # URL strutcure for searching over Microsoft Bing
search engine.       urls1 = []       url1 = 'https://www.bing.com/search?q=' +
query       page = requests.get(url1, headers={'User-agent': 'Mighty
Near'})       soup = bs(page.text, 'html.parser')       for link in
soup.find_all('a'):             url1 = str(link.get('href'))             if
url1.startswith('http'):                 if not url1.startswith('https://go.m') and not
url1.startswith('https://go.m'):                     urls1.append(url1)       return
urls1[:num] def searchGoogle(query, num):       urls2 = []       url2 =
'https://www.google.com/search?q=' + query       page = requests.get(url2,
headers={'User-agent': 'John Doe'})       soup = bs(page.text, 'html.parser')       for
link in soup.find_all('a'):             url2 = str(link.get('href'))             if
url2.startswith('http'):                 if not url2.startswith('https://go.m') and not
url2.startswith('https://go.m') and not
url2.startswith(                 'https://maps.google'):                     urls2.appe
nd(url2)       return urls2[:num] # Extract Text function # The function is used for
extracting the relevant text from the web. def extractText(url):       page =
requests.get(url)       soup = bs(page.text, 'html.parser')       return soup.get_text()
nltk.download('stopwords') nltk.download('punkt') stop_words =
set(nltk.corpus.stopwords.words('english')) # Function for generating tokens #
Returns words from a string passed as input. def purifyText(string):       words =
nltk.word_tokenize(string)       return (" ".join([word for word in words if word not
in stop_words])) # Function for matching results over the web based on the text.

def webVerify(string, results_per_sentence):       sentences =
nltk.sent_tokenize(string)       matching_sites = []       for url in
searchBing(query=string,
num=results_per_sentence):             matching_sites.append(url)       for sentence
in sentences:             for url in searchBing(query=sentence,
num=results_per_sentence):                 matching_sites.append(url)       for url in
searchGoogle(query=string,
num=results_per_sentence):             matching_sites.append(url)       for sentence
```

```
in sentences:            for url in searchGoogle(query=sentence,
num=results_per_sentence):                matching_sites.append(url)       return
(list(set(matching_sites))) # Similarity function # The function calculates and
compares instances to get a ratio for them.

def similarity(str1, str2):       return (SequenceMatcher(None, str1, str2).ratio()) *
100 # Generate report function # Passed input text or file text as parameters. def
report(text):       matching_sites = webVerify(purifyText(text), 2)       matches =
{}       for i in range(len(matching_sites)):            matches[matching_sites[i]] =
similarity(text, extractText(matching_sites[i]))       matches = {k: v for k, v in
sorted(matches.items(), key=lambda item: item[1], reverse=True)}       sum =
0       for k, v in matches.items():            sum += v       matches["TOTAL
SIMILARITY"] = sum       return matches # Return Table function # Used for
returning data-frame to the final report page.

def returnTable(dictionary):       df = pd.DataFrame({'Similarity (%)':
dictionary})       # df = df.fillna(' ').T       # df = df.transpose()       return
df.to_html(classes="table ") path = os.getcwd() UPLOAD_FOLDER =
os.path.join(path, 'uploads') ALLOWED_EXTENSIONS = {'pdf'} app =
Flask(__name__) app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.add_url_rule(       "/uploads/<name>", endpoint="download_file",
build_only=True ) app.config['MAX_CONTENT_LENGTH'] = 16 * 1000 * 1000
app.config['SECRET_KEY'] = 'super secret key' def
allowed_file(filename):       return '.' in filename and \            filename.rsplit('.',
1)[1].lower() in ALLOWED_EXTENSIONS @app.route('/', methods=['GET', 'POST'])
def index():       if request.method == 'POST':            if request.form['text'] != ''
and request.files['file'].filename == '':                word =
request.form['text']                masukan = "word"                with open('word.txt',
'w', encoding='utf-8') as f:                    f.write(word)                return
redirect(url_for('plagiarism', name=masukan) + "#hasil")            elif
request.files['file'].filename != '' and request.form['text'] == '':                if 'file' not
in request.files:                    flash('No file part')                    return
redirect(request.url)                file1 = request.files['file']                # If the user
does not select a file, the browser submits an                # empty file without a
filename.                if file1.filename == '':                    flash('No selected
file')                    return redirect(request.url)                if file1 and
allowed_file(file1.filename):                    filename =
secure_filename(file1.filename)                    file1.save(os.path.join(app.config['U
PLOAD_FOLDER'], filename))                    return redirect(url_for('plagiarism',
name=filename) + "#hasil")            else:                flash('Please fill the
```

```
form')                return redirect(request.url)       return
render_template("index.html") @app.route('/plagiarism/<name>',
methods=['GET', 'POST']) def plagiarism(name):       domain =
"co.id"       link_output = []       hasil_plagiarism = []       hasil_link =
[]       hasil_persen = 0       inputan_mentah = ""       inputan = []       filename =
""       text = ""       hasil_plagiarism_final = []       hasil_link_final =
[]       link_blocked = ["id.linkedin.com", "linkedin.com", "youtube.com",
"instagram.com", "facebook.com",
"tokopedia.com",                "twitter.com", "reddit.com", "bukalapak.com",
"shopee.com", "blibli.com"]       if request.method == 'POST':            if
request.form['text'] != '' and request.files['file'].filename == '':                word =
request.form['text']                filename += "word"                with
open('word.txt', 'w', encoding='utf-8') as f:                f.write(word)           elif
request.files['file'].filename != '' and request.form['text'] == '':                if 'file' not
in request.files:                flash('No file part')                return
redirect(request.url)                file = request.files['file']                # If the user
does not select a file, the browser submits an                # empty file without a
filename.                if file.filename == '':                flash('No selected
file')                return redirect(request.url)                if file and
allowed_file(file.filename):                filename =
secure_filename(file.filename)                file.save(os.path.join(app.config['UPL
OAD_FOLDER'], filename))                pdfFileObj =
open('uploads/{}'.format(filename), 'rb')                pdfReader =
PyPDF2.PdfFileReader(pdfFileObj)                num_pages =
pdfReader.numPages                count = 0                while count <
num_pages:                pageObj =
pdfReader.getPage(count)                count += 1                text
+= pageObj.extractText()           else:                flash('Please fill the
form')                return redirect(request.url)           if filename ==
"word":                if request.method == 'POST':                inputan_mentah
+= request.form['text']                else:                f = open("word.txt",
"r")                inputan_mentah += f.read()                inputan +=
inputan_mentah.replace("\n", " ").split(".

")                for i in range(len(inputan)):                query = '"' +
inputan[i].strip().replace(".", "").replace('"', "'") + '"'                for j in
range(len(list(search(query, tld=domain, num=10, stop=10,
pause=2)))):                if i !=
j:                continue                hasil_plagiarism.append(inputa
n[i])                hasil_link.append(list(search(query, tld=domain, num=10,
```

```
stop=10, pause=2))[j])                  for i in
range(len(hasil_plagiarism)):                       for j in
range(len(hasil_link)):                         if i !=
j:                              continue                        while
True:                               for k in
range(len(link_blocked)):                                   if link_blocked[k] in
hasil_link[j]:                                  break                           else:
                        hasil_plagiarism_final.append(hasil_plagiarism[i])
                hasil_link_final.append(hasil_link[j])                                  break
                        break               count = len(inputan)                count_hasil =
len(hasil_link_final)               hasil_persen += (count_hasil / count) *
100                 for i in
range(len(hasil_link_final)):                       link_output.append(hasil_link_final[i])
        else:               inputan += text.replace("\n", " ").split(". ")                  for i in
range(len(inputan)):                    query = '"' + inputan[i].strip().replace(".",
"").replace('"', "'") + '"'                     for j in range(len(list(search(query,
tld=domain, num=10, stop=10, pause=2)))):                           if i !=
j:                              continue                        hasil_plagiarism.append(inputa
n[i])                       hasil_link.append(list(search(query, tld=domain, num=10,
stop=10, pause=2))[j])                  for i in
range(len(hasil_plagiarism)):                       for j in
range(len(hasil_link)):                         if i !=
j:                              continue                        while
True:                               for k in
range(len(link_blocked)):                                   if link_blocked[k] in
hasil_link[j]:                                  break                           else:
                        hasil_plagiarism_final.append(hasil_plagiarism[i])
                hasil_link_final.append(hasil_link[j])                                  break
                        break               count = len(inputan)                count_hasil =
len(hasil_link_final)               hasil_persen += (count_hasil / count) *
100                 for i in
range(len(hasil_link_final)):                       link_output.append(hasil_link_final[i])
  else:             if name == "word":                  if request.method ==
'POST':                         inputan_mentah +=
request.form['text']                    else:                       f = open("word.txt",
"r")                        inputan_mentah += f.read()                  inputan +=
inputan_mentah.replace("\n", " ").split(".

")                  for i in range(len(inputan)):                       query = '"' +
inputan[i].strip().replace(".", "").replace('"', "'") + '"'                        for j in
```

```
range(len(list(search(query, tld=domain, num=10, stop=10,
pause=2)))):                    if i !=
j:                    continue                    hasil_plagiarism.append(inputa
n[i])                    hasil_link.append(list(search(query, tld=domain, num=10,
stop=10, pause=2))[j])                for i in
range(len(hasil_plagiarism)):                    for j in
range(len(hasil_link)):                    if i !=
j:                    continue                    while
True:                for k in
range(len(link_blocked)):                    if link_blocked[k] in
hasil_link[j]:                    break                else:
                hasil_plagiarism_final.append(hasil_plagiarism[i])
            hasil_link_final.append(hasil_link[j])                    break
                break                count = len(inputan)                count_hasil =
len(hasil_link_final)                hasil_persen += (count_hasil / count) *
100                for i in
range(len(hasil_link_final)):                    link_output.append(hasil_link_final[i])
    else:                pdfFileObj = open('uploads/{}'.format(name),
'rb')                pdfReader =
PyPDF2.PdfFileReader(pdfFileObj)                num_pages =
pdfReader.numPages                count = 0                text = ""                while
count < num_pages:                    pageObj =
pdfReader.getPage(count)                    count += 1                    text +=
pageObj.extractText()                inputan += text.replace("\n", " ").split(".

")                for i in range(len(inputan)):                    query = '"' +
inputan[i].strip().replace(".", "").replace('"', "'") + '"'                    for j in
range(len(list(search(query, tld=domain, num=10, stop=10,
pause=2)))):                    if i !=
j:                    continue                    hasil_plagiarism.append(inputa
n[i])                    hasil_link.append(list(search(query, tld=domain, num=10,
stop=10, pause=2))[j])                for i in
range(len(hasil_plagiarism)):                    for j in
range(len(hasil_link)):                    if i !=
j:                    continue                    while
True:                for k in
range(len(link_blocked)):                    if link_blocked[k] in
hasil_link[j]:                    break                else:
                hasil_plagiarism_final.append(hasil_plagiarism[i])
            hasil_link_final.append(hasil_link[j])                    break
```

```
                    break                count = len(inputan)                count_hasil =
len(hasil_link_final)                hasil_persen += (count_hasil / count) *
100                for i in
range(len(hasil_link_final)):                    link_output.append(hasil_link_final[i])
  return render_template("index.html", hasil_persen=hasil_persen,
data=inputan,                                hasil_plagiarism=hasil_plagiarism_final,
                        link_output=link_output, hasil_link=hasil_link_final) # To
generate report @app.route('/report', methods=['POST', 'GET']) def result():       if
request.method == 'POST':            result = request.form['text']            if len(result)
== 0:                resultf = request.files['myfile']                result = {'myfile':
resultf.read()}            test = returnTable(report(str(result)))            return
render_template('report.html', PWM_value=test) # Compare b/w two files code
below # Check root_text_from_html = 1   & root_file_from_html = 2 # Check
plag_text_from_html = 1@ & plag_file_from_html = 2@ @app.route('/f2f_kmp',
methods=['GET', 'POST']) def index1():       global rootText, rootFile, plagText,
plagFile       if request.method == 'POST':            # 1(available) &
2(unavailable)            if request.form['root_text_from_html'] != '' and
request.files['root_file_from_html'].filename == '':                # 1@(available) &
2@(unavailable)                if request.form['plag_text_from_html'] != '' and
request.files['plag_file_from_html'].filename == '':                    rootText =
request.form['root_text_from_html']                    plagText =
request.form['plag_text_from_html']                    return
redirect(url_for('onRunF2F'))                # 1@(unavailable) &
2@(available)                    elif request.form['plag_text_from_html'] == '' and
request.files['plag_file_from_html'].filename != '':                    rootText =
request.form['root_text_from_html']                    if 'plag_file_from_html' not in
request.files:                        flash('No file part')                        return
redirect(request.url)                    plagFile =
request.files['plag_file_from_html']                    # If the user does not select a
file, the browser submits an                    # empty file without a
filename.                    if plagFile.filename == '':                        flash('No
selected file')                        return redirect(request.url)                    if
plagFile and allowed_file(plagFile.filename):                        filename =
secure_filename(plagFile.filename)                        plagFile.save(os.path.join(ap
p.config['UPLOAD_FOLDER'], filename))                        return
redirect(url_for('onRunF2F'))                # BOTH 1@ &
2@(unavailable)                    else:                    flash('Please fill the
form')                return redirect(request.url)            # 1(unavailable) &
2(available)                    elif request.form['root_text_from_html'] == ''
and request.files['root_file_from_html'].filename != '':                # 1@(available) &
```

```
2@(unavailable)                 if request.form['plag_text_from_html'] != '' and
request.files['plag_file_from_html'].filename == '':                    plagText =
request.form['plag_text_from_html']                 if 'root_file_from_html' not in
request.files:                          flash('No file part')                       return
redirect(request.url)                   rootFile =
request.files['root_file_from_html']                    # If the user does not select a file,
the browser submits an                  # empty file without a
filename.                   if rootFile.filename == '':                         flash('No
selected file')                     return redirect(request.url)                    if
rootFile and allowed_file(rootFile.filename):                           filename =
secure_filename(rootFile.filename)                      rootFile.save(os.path.join(ap
p.config['UPLOAD_FOLDER'], filename))                   return
redirect(url_for('onRunF2F'))               # 1@(unavailable) & 2@(available) --->
Only both 2 & 2@ are availabe                   elif
request.form['plag_text_from_html'] == '' and
request.files['plag_file_from_html'].filename != '':                    # For 2 ->
(root_file_from_html)                   if 'root_file_from_html' not in
request.files:                          flash('No file part')                       return
redirect(request.url)                   rootFile =
request.files['root_file_from_html']                    # If the user does not select a file,
the browser submits an                  # empty file without a
filename.                   if rootFile.filename == '':                         flash('No
selected file')                     return redirect(request.url)                    if
rootFile and allowed_file(rootFile.filename):                           filename =
secure_filename(rootFile.filename)                      rootFile.save(os.path.join(ap
p.config['UPLOAD_FOLDER'], filename))                   # For 2 ->
(root_file_from_html)                   if 'plag_file_from_html' not in
request.files:                          flash('No file part')                       return
redirect(request.url)                   plagFile =
request.files['plag_file_from_html']                    # If the user does not select a
file, the browser submits an                    # empty file without a
filename.                   if plagFile.filename == '':                         flash('No
selected file')                     return redirect(request.url)                    if
plagFile and allowed_file(plagFile.filename):                           filename =
secure_filename(plagFile.filename)                      plagFile.save(os.path.join(ap
p.config['UPLOAD_FOLDER'], filename))                   return
redirect(url_for('onRunF2F'))               # BOTH 1@ &
2@(unavailable)                 else:                   flash('Please fill the
form')                  return redirect(request.url)                    # When nothing is
available (1, 2, 1@, ,2@)           else:                   flash('Please fill the
```

```
form')                return redirect(request.url)        return render_template("f2f.html")
def computeLPSArray(pat, M, lps):        len = 0        lps[0]        i = 1        while i <
M:            if pat[i] == pat[len]:                len += 1                lps[i] =
len                i += 1            else:                if len != 0:                    len =
lps[len - 1]                else:                lps[i] = 0                    i += 1 def
KMPSearch(pat, text, p):        M = len(pat)        N = len(text)        lps = [0] * M        j
= 0        computeLPSArray(pat, M, lps)        i = 0        while i < N:            if
pat[j].lower() == text[i].lower():                i += 1                j += 1            if j ==
M:                p += 1                j = lps[j - 1]                break            elif i < N
and pat[j].lower() != text[i].lower():                if j != 0:                    j = lps[j -
1]                else:                    i += 1        return p
@app.route('/f2f_kmp/onRunF2F', methods=['GET', 'POST']) def plag():        if
request.method == 'POST':            rootText =
request.form['root_text_from_html']            plagText =
request.form['plag_text_from_html']            plagFile =
request.files['plag_file_from_html']            rootFile =
request.files['root_file_from_html']            # result = request.form['text']            #
if len(result) == 0:            #      resultf = request.files['myfile']                text =
rootText            pattern = plagText            if len(text) == 0:                # resultf =
request.files['myfile']                text = {'root_file_from_html':
rootFile.read()}            if len(pattern) == 0:                # resultf =
request.files['myfile']                pattern = {'plag_file_from_html':
plagFile.read()}            sentences = re.split(r'[\.\?!\r\n]',
pattern)            counter_matched = 0            counter_total = 0            p =
0            for pattern in sentences:                pattern = pattern.strip()                if
len(pattern) > 0:                    counter_total +=
1                    counter_matched += KMPSearch(pattern, text, p)            rez =
counter_matched * 100 / counter_total            return render_template("f2f.html",
F2F_value=rez) if __name__ == "__main__":        app.run(debug=True,
host='127.0.0.1', port=5555)        app.run()        report('This is a pure test')
```

INTERNET SOURCES:

1% - https://note.com/satra/n/n2e14b37e2195
<1% - https://stackoverflow.com/questions/44589777/range-length-in-python
<1% - https://blog.miguelgrinberg.com/post/handling-file-uploads-with-flask
<1% - https://stackoverflow.com/questions/49121365/implementing-retry-for-requests-in-python

<1% -
https://www.chegg.com/homework-help/questions-and-answers/import-modules-import-pandas-pd-import-numpy-np-bs4-import-beautifulsoup-import-re-import--q74399606
<1% -
https://github.com/harirakul/Plagiarism-Detection/blob/master/websearch.py
<1% -
https://www.coursehero.com/tutors-problems/Python-Programming/20316867-I-need-help-figuring-out-why-this-code-keeps-giving-me-a-syntax-error/
1% - https://github.com/harirakul/Plagiarism-Detection/blob/master/similarity.py
<1% -
https://thispointer.com/sort-a-dictionary-by-value-in-python-in-descending-ascending-order/
<1% -
https://www.geeksforgeeks.org/how-to-create-dataframe-from-dictionary-in-python-pandas/
<1% -
https://stackoverflow.com/questions/42128484/flask-how-to-get-uploads-folder-path
<1% - https://github.com/pallets/flask/blob/main/docs/patterns/fileuploads.rst
<1% - https://it-undarmaa.github.io/python-backend/
<1% -
https://thewebdev.info/2020/10/08/python-web-development-with-flask%e2%80%8a-%e2%80%8arequest-and-response/
<1% -
https://stackoverflow.com/questions/34705969/how-to-read-a-text-file-and-write-it-word-by-word-into-another-file-in-python
<1% -
https://towardsdatascience.com/writing-a-multi-file-upload-python-web-app-with-user-authentication-8f75064b819a
<1% -
https://stackoverflow.com/questions/12309269/how-do-i-write-json-data-to-a-file
<1% - https://github.com/Rdg0/api_file_server_flask/blob/main/app.py
<1% - https://python.engineering/working-with-pdf-files-in-python/
<1% -
https://stackoverflow.com/questions/60215731/pypdf-to-read-each-pdf-in-a-folder

<1% -
https://peter-easter-do.medium.com/parsing-icd-codes-with-python-c478653a943a
<1% -
https://stackoverflow.com/questions/56679016/how-to-call-function-if-http-request-is-post-in-django-views-and-pass-new-submit
<1% -
https://stackoverflow.com/questions/62525450/possible-to-rewrite-for-i-in-range-loop-on-i-i1-into-for-i-in-loop-in-python
<1% -
https://stackoverflow.com/questions/53358753/why-we-use-rangelen-in-for-loop-in-python
<1% -
https://stackoverflow.com/questions/32930246/python-for-loops-for-i-in-range0-lenlist-vs-for-i-in-list
<1% - https://www.pythonanywhere.com/forums/topic/1695/
<1% -
https://stackoverflow.com/questions/62301945/how-to-open-pdf-file-using-pypdf2
<1% -
https://betterprogramming.pub/how-to-convert-pdfs-into-searchable-key-words-with-python-85aab86c544f
<1% - https://www.programcreek.com/python/example/51528/flask.request.files
<1% -
https://stackoverflow.com/questions/61534027/how-should-i-handle-duplicate-filenames-when-uploading-a-file-with-flask
<1% - https://github.com/tiangolo/fastapi/issues/426
<1% -
https://intellij-support.jetbrains.com/hc/en-us/community/posts/360009793540-Pycharm-keeps-asking-for-command-line-args-although-argparse-is-commented-out
<1% -
https://www.geeksforgeeks.org/python-program-for-kmp-algorithm-for-pattern-searching/
<1% -
https://stackoverflow.com/questions/69848780/if-request-method-post-and-sub-in-request-form-is-not-working-in-flask