**Analysis of Algorithms**

Spring 2020

**Members Details**

| Group ID | CS311S20PIDG02 |
| --- | --- |
| Registration Number of Group Members | 2018-CS-03 , 2018-CS-29 |
| Section | A |

**Project Details**

|  |  |
| --- | --- |
| ***Project*** |  |
| Project Title | PLAGIARISM CHECKERISTA |
| Executive Summary | Our plagiarism checker tool “**Plagiarism Checkerista**” works same as other plagiarism tools works but on a small scale. It is mainly built to show how plagiarism is checked among different files, it don’t have a large database. It takes one file or some piece of text from user as input, checking it with our small databases files one by one and bringing the result in the form of lines that have been plagiarized from all the files also giving percentage of how much content in user document is plagiarized. We have built our app in python using Django framework. Django is a very powerful tool for building APIs.it is a full stack web framework providing the facility of both embedded frontend and backend development making it easier for developer to make connections. In templates and static named folders all front end based files are placed. And in **views.py** file we have the real implementation of the tool. Connections are made in two **urls.py** files one is in the tool folder and the other is in checker folder.  Explaining the whole workflow of the app: **Checkerista>>.env**(Virtual environment for Django is created in this folder), **.idea**(also created in creation of virtual environment), **.vs**( Information for workplace for our project is stored in it), **checker**(Folder of user concern )**>>checker**(contains files like settings.py and urls.py but again not to be touched by the user), **Static/css >> main.css**(contains the css of 1st web page ) , **page2.css** (contains the css of the 2nd web page) ,**templates >> tool.html** (the first web page that asks user to enter data i.e. Text), **result.html** (the final page on which the result is shown), **tool** (this folder has the views.py file admin.py file models.py urls.py and some other files )**,**  **db.sqlite3** (it is the database which comes embedded with Django virtual environment and we are using dbsqlite3 to manage database of our project), **manage.py**(it has some built in code that is used for running the server. We can run our server if we are in the same folder in this case checker as manage.py file otherwise server will not be run on the system.) Procedure of running server is explained in **Readme.md** file associated with the project.  **Virtual environment:** Django framework needs a virtual environment to work in this is what everyone have to install on their own systems by running some commands explained in **Readme.md** file. |
| ***Business Case*** |  |
| Outline the business need for the project | There are already many plagiarism checker tools in the market. Some of very low functionality, medium and some of very high those mainly are paid apps. With this increasing demand of good quality work people are there that plagiarize content of other and present it as their own work. This is wrong in academia especially and we need to have tools to bring out plagiarized content of those people. So, a little contribution to high demanding community. |
| End user of the product | The User for “Plagiarism can be any instructor who want to know whether his pupil has plagiarized or not, any student who wants to know after plagiarizing how much he needs to change it to deceive his instructor (That’s a truly wrong use but unfortunately most of the users of these plagiarism checker tools are these students), any type of user. |
| Motivation for Project | It gave us drive to choose plagiarism checker tool among the other projects provided by our instructor that we want to know and implement how string matching is done and it seems interesting to us while we were developing it. |
| Description of the project objective(s) | The following are the objectives of  **Plagiarism Checkerista** :   * Checks the file or text document given by the user by sending data entered in textfield to the database. * On View Result result is displayed |
| State the level of impact expected should the project proceed and implications of not proceeding | The project is supposed to be giving plagiarized content after checking it and highlighting it. It may not be functioning properly and checking might not be done properly and not expected result might be diplayed. |
| Functional Requirements | Our tool performs following:   * Let user input in the textbox * On submit click it sends data for checking * On view Result it displays the next page containing result and percentage of plagiarized content. |
| ***Benefits*** |  |
| What benefits are expected/ anticipated? | Academically and Industrially it is beneficial as it:   * Purely written in Python a language that says: “Program development using python is 5-10 faster than using C/C++ and 3-5 times faster than using Java. * The “batteries included philosophy” means it comes with a lot of benefits like free authentication, admin panel, ORM, semi-automatic DB migrations from python code, lightweight development web server, and many more. * The point in using Django is that when you want to quickly develop your product. |
| ***Implementation Details*** |  |
| Link to Github Repository | <https://github.com/LybaFatimaNasir/CS311S20PID02> |
| Total Number of commits in repository before 5th August 2020 |  |
| Exact contribution of each member |  |
| ***Commits in github repository by each member*** | |
| |  |  | | --- | --- | | **Member Registration No.** | **Total Commits** | | 2018-CS-03 |  | | 2018-CS-29 |  | |  |  | | |
| **Details of commits** | |
| |  |  |  |  | | --- | --- | --- | --- | | **Sr. No.** | **Details of commit** | **Date** | **Member Reg No.** | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| Have you used built in algorithms or you have implemented yourself? | We have used LCS( longest Common subsequence ) algorithm in our project. |
| Formats of input | The format of input should be text document containing English words or simple English words. |
| Validations | We have validated that input should be only English words. |
| Format of output | The format of output is also text file displayed in the box with highlighted content/lines and plagiarism percentage displayed. |
| Deployment | *Have you deployed your project in any format? If yes, provide the details* |
| ***Details of algorithms*** | |
| Algorithm that will be used to implement this tool will be the Longest Common Sub-sequence (LCS). LCS can be implemented using dynamic programming or can be recursively but we will be following dynamic programming problem solving technique to find optimal solution. It works by comparing two strings in our two files (One that user entered on frontend with all the files that are at our backend) and displaying content that is matched.  **Pseudocode**  function Checker(C[0...m,0...n],x[1...m],Y[1...n],i,j)  if i<0 and j<0  return  elseif i<0  Checker(C,X,Y,i,j-1)  print "+" + Y[j]  elseif j<0  Checker(C,X,Y,i-1,j)  print "-" + X[i]  elseif X[i] == Y[j]  Checker(C,X,Y,i-1,j-1)  print " " + X[i]  elseif C[i][j-1] >= C[i-1][j]  Checker(C,X,Y,i,j-1)  print "+" + Y[j]  elseif C[i][j-1] < C[i-1][j]  Checker(C,X,Y,i-1,j)  print "-" + X[i]  function Check(X,Y)  C = lcslen(X,Y)  return Checker(C,X,Y,len(X)-1,len(Y)-1)  function backtrack(C[0..m,0..n], X[1..m], Y[1..n], i, j)  if i = 0 or j = 0  return ""  if X[i] == Y[j]  return backtrack(C, X, Y, i-1, j-1) + X[i]  if C[i,j-1] > C[i-1,j]  return backtrack(C, X, Y, i, j-1)  return backtrack(C, X, Y, i-1, j)  function lcs(X[1..m], Y[1..n])  C=lcs(X, Y)  return backtrack(C, X, Y,len(X-1),len(X,-1))    function lcslen(X[1..m], Y[1..n])  C = [[for i =0 to (len(Y) + 1)] for j= 0 to (len(X+ 1)]  for i, X[i] to enumerate(X)  for j, Y[i] to enumerate(Y)  if X[I] == Y[j]  C[i][j] = 1 + C[i-1][j-1]  else  C[i][j] = max(C[i][j-1], C[i-1][j])  return C  **Correctness**  **checker and lcs**  **inductive hypothesis:**  The algorithm computes C[X ,Y] correctly for all (x, y) < (i, j). (C[X, Y] is computed before C[i, j].) FOR LCS: it gives the maximum length  **base case:**  C[0, j] = C[i, 0] = 0. Correct because empty sequence has no LCS.  **inductive step:**  Assume IH is true. When computing C[i, j], the transition considers 3 cases.  If x[i] = y[j], then c[k] = x[i] = y[j] and C[k]−1 is an LCS of X[i−1] and Y[j−1].  If x[i] != y[j], then c[k] != x[i] implies that C is an LCS of X[i−1] and Y.  If x[i] != y[j], then C[k] != y[j] implies that C is an LCS of X and Y[j−1].  By IH, C[i − 1, j − 1], C[i − 1, j], m[i, j − 1] are computed correctly.  Hence, the algorithm makes the correct decision and C[X, Y] is computed correctly  **Complexity Analysis**  function lcslen(X[1..m], Y[1..n])  C = [[for i =0 to (len(Y) + 1)] for j= 0 to (len(X+ 1)]--nm  for i, X[i] to enumerate(X)-----------------n+1  for j, Y[i] to enumerate(Y) ----------------n(m+1)  if X[I] == Y[j] ---------------------------nm  C[i][j] = 1 + C[i-1][j-1]------------------  else  C[i][j] = max(C[i][j-1], C[i-1][j]) -----------  return C------------------------------1  T(n)=(n+1)+n(m+1)+nm+1+nm  T(n)=n+1+nm+n+nm+1+nm  T(n)=2n+3nm+2  function backtrack(C[0..m,0..n], X[1..m], Y[1..n], i, j)  if i = 0 or j = 0 --------------------1  return ""  if X[i] == Y[j] ----------------------1  return backtrack(C, X, Y, i-1, j-1) + X[i]  if C[i,j-1] > C[i-1,j]----------------1  return backtrack(C, X, Y, i, j-1)  return backtrack(C, X, Y, i-1, j)---------1  T(n)=T(n)+4    function lcs(X[1..m], Y[1..n])  C=lcs(X, Y) -----------------------1  return backtrack(C, X, Y,len(X-1),len(X,-1)) ------------1  T(n)=2  function Checker(C[0...m,0...n],x[1...m],Y[1...n],i,j)  if i<0 and j<0--------------n  return-------------------1  elseif i<0-----------------n  Checker(C,X,Y,i,j-1)-------1  print "+" + Y[j]  elseif j<0-----------------n  Checker(C,X,Y,i-1,j) -------1  print "-" + X[i]  elseif X[i] == Y[j]----------n  Checker(C,X,Y,i-1,j-1) -------1  print " " + X[i]  elseif C[i][j-1] >= C[i-1][j]---------n  Checker(C,X,Y,i,j-1) -------1  print "+" + Y[j]  elseif C[i][j-1] < C[i-1][j]----------n  Checker(C,X,Y,i-1,j) -------1  print "-" + X[i]  T(n)=6n+6    function Check(X,Y)  C = lcslen(X,Y)-------------1  return Checker(C,X,Y,len(X)-1,len(Y)-1)---------------1  T(n)=2  **Worst case time complexity: O(nm)\***  **Average case time complexity: O(nm)\***  **Best case time complexity: O(nm)\***  Since we are using two for loops for both the strings , therefore the time complexity of finding the longest common subsequence using dynamic programming approach is O(n \* m) where n and m are the lengths of the strings. Since this implementation involves only n rows and m columns so complexity becomes O(n \* m). | |
| ***Interfaces for your project*** | |
| This the first page where user is supposed to enter data in the box and click submit will send data to the database and then clicking ViewResult you will be able to see the result page which is as follow:    The result is shown here this is the on click function when clicked result is already displayed user can only see the result. | |
| ***Integration*** | |
| *What type of difficulties were faced by you while integration of UI and algorithms. What was your strategy in this regard.* | |
| ***Change Requests*** | |
| *Do you have any changes in proposed algorithm(in the previous deadlines) or changes in interface. Mention all the changes here* | |
| ***Testing*** | |
| *In this section, you are required to mention the issues report and solution proposed.* | |
| ***Technology*** |  |
| Programming Language | Python3 |
| Platform | Web Application |