Final Team Report

1. Introduction

1.1 Background

Students especially in computer science department regularly submit code as part of their assignment. The department now has around 240 undergraduate students and it is difficult for academics to detect plagiarism in code when marking this amount of work. Checking code similarity by teachers will cost a lot of time and cause low efficiency.

1.2 Purpose

We implement the code similarity detection service to check the assignments (or quizzes), which can avoid the unfair means events from happening. If teachers use this code similarity detection service, it can not only reducing the workload of teachers, but also making it easier for teachers to grade the assignment.

1.3 Main Work

Code similarity detection service is a tool based on web service to detect code clone of submitted assignments between classmates. We have mainly carried out two parts of work. In the first part, we use PHP to build our web service. Then we wrote the user interface, in which we will create the function for users to upload assignments or files. Meanwhile, the second part, code similarity detection is carried out in the same time. We chose an algorithm for this function. Then implement it according to the algorithm, which can deal with every assignment or file uploaded, and highlight the three different types of code similarities by using different colours, and indicate it is similar with which classmates and the degree of similarity. Finally generate documents.

2. Project scope and objectives

2.1 Project scope

(1) Operating system: Windows and Linux
(2) The coding language being detected: Java
(3) Coding language: PHP and python
(4) The file format of uploading files: zip files
(5) The maximun size of uploading files：20480000kB
(6) The file format of outputs：HTML and ZIP files containing PDF files
(7) Just for Directly copied code and Code that has been copied and then had variable names changed. Not for potentially advanced: code where small changes have been made, like swapping for-loops for while loops, or switch statements for ifs.

2.2 Project objectives

The aim of this team project is to implement the code similarity detection service. This project has three main functions :

(1) Uploading zip files:

- Implement the upload function, including uploading .zip files.
- After receiving the zip files, we have a function to unzip them, and then store the paths of unzip files to an array.
- Transfer the path array to Code similarity detection module.

(2) Detecting the clone code:

- The first part is preprocessing. Preprocessing is to delete the useless context in the code file including spaces, some meaningless punctuations and comments. Then preprocessing will divide the original code file into individual words.
- The second part is token extraction, the preprocessed line sequence is tokenized and the variables, keywords and symbols is encoded separately by assigning ascii number to each functor. In the end of this step, it will produce a string representing the whole file.
- The third part is to use the algorithm based on suffix-array to detect the repeated substring in the whole string. Then, output the clone code zone for each Doc and calculate the similarity rate for each Document.

(3) Presenting the result:

- One method is to show the report online.
- Another method is to turn the online report to PDF files, then zip those files as one result. Then users can download the .zip result.

3. Team name and list of members

(1) Team name: Punchline
(2) List of members: Zixuan Zhang, Chenxi Liu, Lu Wang, Sheng Huang, Chen Luziyang, Ruocheng Ma.

4. User stories

Our user stories are as follows (shown in Table 4.1).

Table 4.1: User stories

No.	User story	Task assigned
1	Submit and save assignment submissions as Java code.	1. Identify style of files uploaded, ignore wrong style files 2. Save files and paths of them in the database
2	Identify potential plagiarism by scanning each submission for similarity between all submissions. Identify directly copied code or the code that has been copied and then had variable names changed.	1.preprocessing:delete the useless context in the code file 2.Token extraction:produce a string representing the whole file 3.Clone code detection:implement algorithm based on suffix-array
3	Can detect more categories of coding language.	1. Research on how to extend the code 2. Implement the code to detect Python
4	Generate a rate of similarity for each code.	1. Develop an algorithm to generate a rate of similarity 2. Find a standard of rate to judge whether a code is plagiarism or not
5	Generate a report for each submission and highlight all possible plagiaristic code.	1. Show users the html report on the web page 2. Convert the html files into PDF files for users to download. 3. Highlight plagiarisms parts.
6	Need a web service where academics can upload zip files to and download result reports.	1.Use html to complete the upload function. 2.Use php to complete the download function.

5. Analysis & Design

5.1 System Architecture

This is our system architecture map(shown in Figure 5.1), and we divide our project into four modules, the following is the description :
(1) User interface:
In this project, we use HTML, CSS, and PHP to do interface design, and by request and response methods to connect with server.
(2) Server:
In this project, we chose the server in PHP as our web server, and merge the user interface, database, and clone code detection.
(3) Database:
In this project, we use MySQL as our database, and we combine the database and server by PDO.
(4) Clone code detection:
In this project, we use Python to complete the algorithm based on Token comparison to detect the clone code.

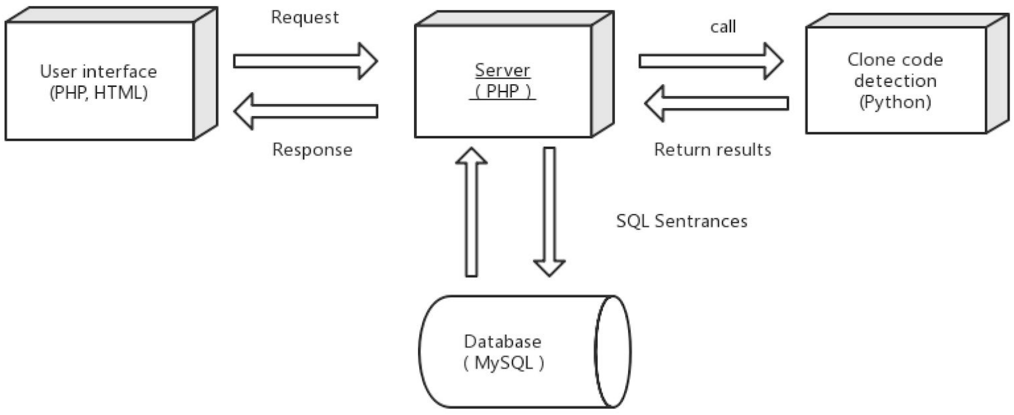


Figure 5.1: System Architecture Map

5.2 Language

(1) JavaServer Pages
As Tyson (2019) defined, JavaServer Pages is the technology that can help us build the front end interface, it can be used together with HTML and CSS, they can create dynamic application.
(2) Python
Sodhi (2019) declared that python is the most important programming language which will be used in artificial intelligent applications, python itself is very simple, but it has a rich library of modules and packages which can help users to solve most of the problems, such as real life or research problems.
(3) PHP
PHP is designed for creating dynamic Web applications. Zeev Suraski (2002) demonstrated that using php can create very powerful applications at a short time. Meanwhile, Machlis (2002) declared that php is a free and easy programming language because it is an open-source language. So, there are many materials about it and it is also very easy to learn. Therefore, php is a very suitable language for the web-based programs.

5.3 Algorithm

The clone detection technique we used is proposed by Kamiya(2002) which consists of the transformation of input source text and a token-by-token comparison. The advantage of this algorithm is that it is not very complicated but can detect both directly copied code and code that has been copied and then had variable names changed.

The code similarity detection part is written in python and is consists of four steps: preprocessing, token extraction, token comparison and report production.

(1)Preprocessing

Preprocessing is to use the regular expression to delete the useless context in the code file including spaces, some meaningless punctuations and set up a dictionary to restore the these punctuations. The algorithm is to use the dictionary to split the original code file into individual words and store them into an array.

(2)Token Extraction

The preprocessed line sequence is tokenized in this section. As Figure 5.2 manifested, it encodes the variables, keywords and symbols separately by assigning ascii number to each functor. In the end of this section, it will produce a string representing the whole file. Table5.1 summarized the lexical function of key word and symbols in java.

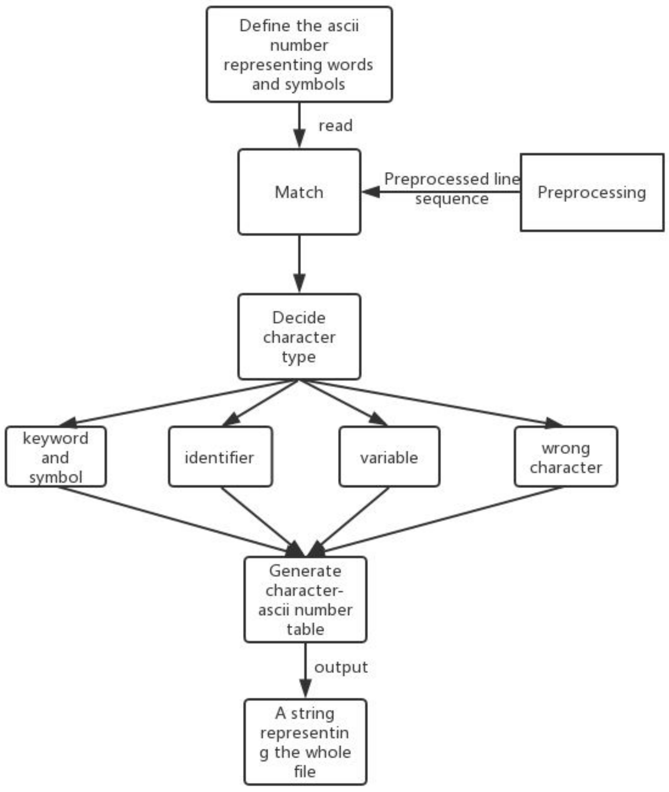
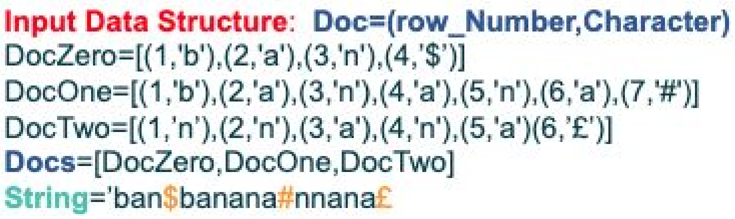

Define the ascii number representing words and symbols
read
Match
Preprocessed line sequence
Preprocessing
Decide character type
keyword and symbol
identifier
variable
wrong character
Generate character-ascii number table
output
A string representin g the whole file

Figure 5.2: Data Flow Chart of Token extraction

Table 5.1: The lexical function of key word and symbols in java

Delimiter	"[","]","(",")","{","}"
Operators	"+","*","-","/","%","="
Relation Operators	">","<","<=",">=","<>","=="
Logical operators	"||","&&","!"
Access control	"private","protected","public"
Classes, methods, and variable modifiers	"abstract","final","class","extends","implements", "interface","native","new","static","strictfp","synch ronized","volatile","transient"
Exception handling	"try","catch","finally","throw","throws"
Variable reference	"super","this","void"
Basic type	"boolean","byte","char","double","float","int","long ","short","null","true","false","String"
Program control	"continue","return","do","while","if","else","for","in stanceof","switch","case","default"
Reserve keyword	"goto","const"

(3)Search repeated token strings and detect code clone
As you can see Figure 5.3, in order to find the copy code among all the documents, we merge the context of each Doc to one string. Next, we get the border of each Doc in the whole string and use the algorithm based on suffix-array to generate a pair of locations of two clone code. Thirdly we merge overlapping zones of copy code. Finally: we compute the similarity rate of each Doc (See Figure 5.4).



```
Input Data Structure:  Doc=(row_Number,Character)
DocZero=[(1,'b'),(2,'a'),(3,'n'),(4,'$')]
DocOne=[(1,'b'),(2,'a'),(3,'n'),(4,'a'),(5,'n'),(6,'a'),(7,'#')]
DocTwo=[(1,'n'),(2,'n'),(3,'a'),(4,'n'),(5,'a')(6,'£')]
Docs=[DocZero,DocOne,DocTwo]
String='ban$banana#nnana£
```

Figure 5.3: Input Data Example

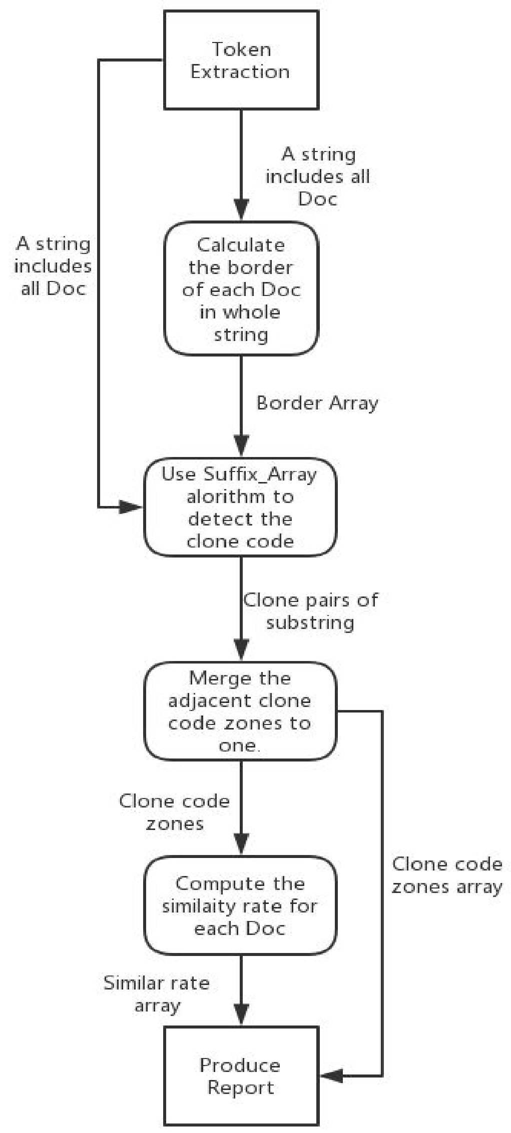
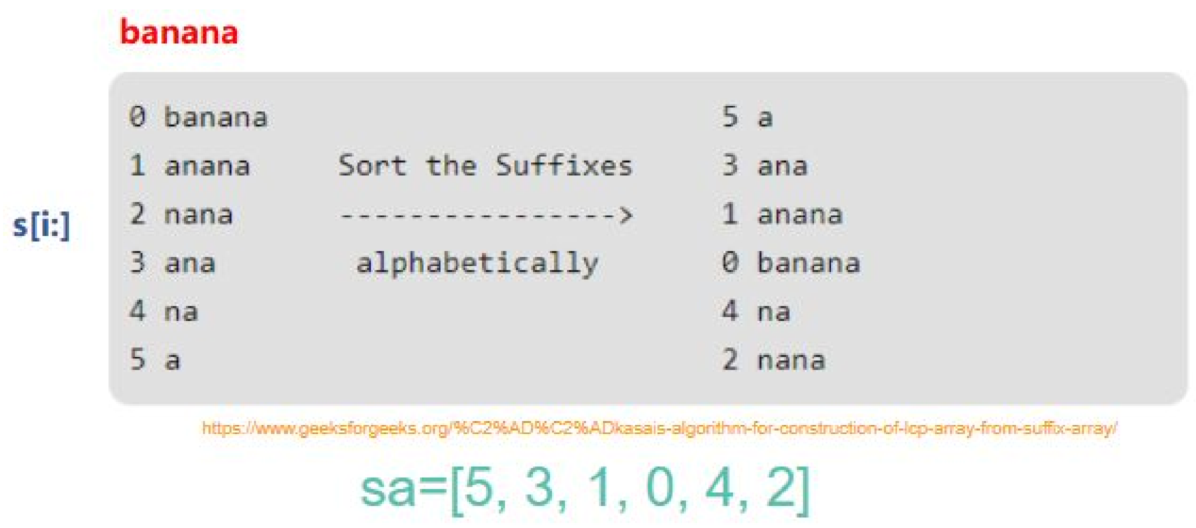

Token Extraction
A string includes all Doc
A string includes all Doc
Calculate the border of each Doc in whole string
Border Array
Use Suffix_Array alorithm to detect the clone code
Clone pairs of substring
Merge the adjacent clone code zones to one.
Clone code zones
Clone code zones array
Compute the similaity rate for each Doc
Similar rate array
Produce Report

Figure 5.4: Data Flow Chart of Clone Code Detection

For the details of algorithm based on suffix array, we use a string of ‘banana’ to illustrate the process. Suffix array is the result of lexicographic order of the suffix at index i (S[i:]).
The left side of Figure 5.5 is suffixes and the right side is the result after alphabetically sorting.



banana

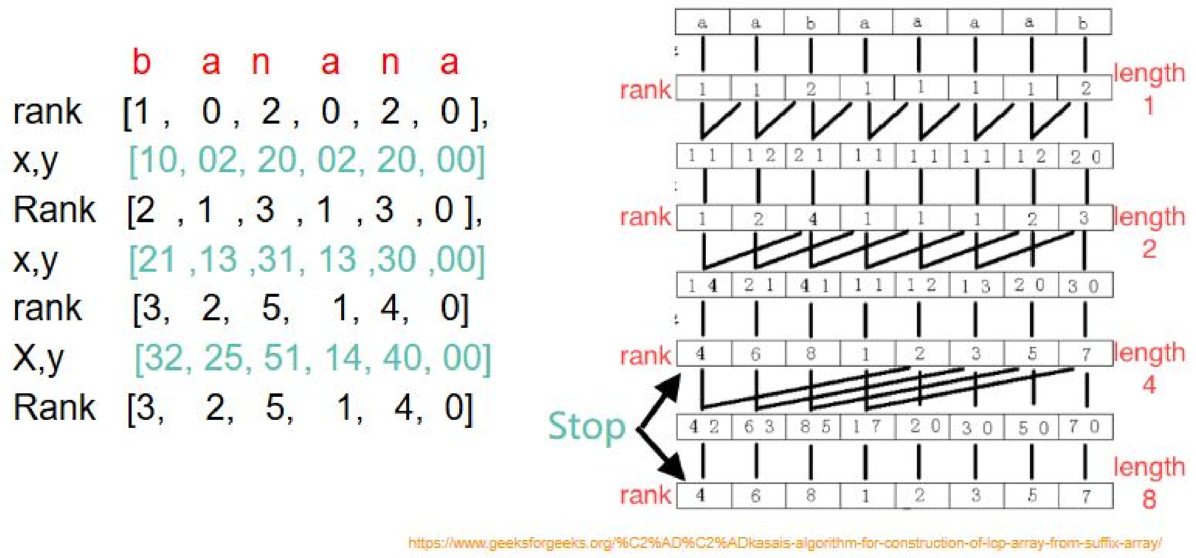
```
     0 banana                          5 a
     1 anana    Sort the Suffixes      3 ana
s[i:] 2 nana    ----------------->     1 anana
     3 ana       alphabetically        0 banana
     4 na                              4 na
     5 a                               2 nana
```

https://www.geeksforgeeks.org/%C2%AD%C2%ADkasais-algorithm-for-construction-of-lcp-array-from-suffix-array/

sa=[5, 3, 1, 0, 4, 2]

Figure 5.5: Lexicographic Order of the Suffixes

To sort those suffixes, we use prefix doubling algorithm. As the right side of the Figure 5.6 shown, firstly, we use integer number to represent the character according to the lexicographic order. Then combine the adjacent ones to new one and use integer number to represent them according to the number size. Repeat above steps until the result does not change.
The left side is an example of string ‘banana’. The last rank array is the result of sorting suffixes of ‘banana’.

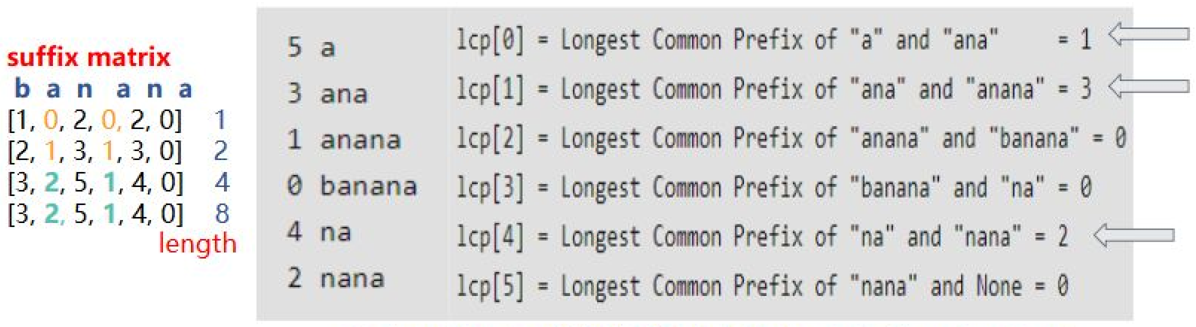


```
         b   a   n   a   n   a
rank   [1 ,  0 , 2 , 0 , 2 , 0 ],
x,y    [10, 02, 20, 02, 20, 00]
Rank   [2 , 1 , 3 , 1 , 3 , 0 ],
x,y    [21 ,13 ,31, 13 ,30 ,00]
rank   [3,  2,  5,  1,  4,  0]
X,y    [32, 25, 51, 14, 40, 00]
Rank   [3,  2,  5,  1,  4,  0]
```

https://www.geeksforgeeks.org/%C2%AD%C2%ADkasais-algorithm-for-construction-of-lcp-array-from-suffix-array/

Figure 5.6: Prefix Doubling Algorithm

After suffixes of string ‘banana’ having been sorted, we need to calculate the longest common prefix between the adjacent ones. According to whether the rank of each pair of suffixes is same in each step (see Figure 5.7, left), we calculate the length of longest common prefix between substrings.


suffix matrix

b a n a n a

[1, 0, 2, 0, 2, 0] 1

[2, 1, 3, 1, 3, 0] 2

[3, 2, 5, 1, 4, 0] 4

[3, 2, 5, 1, 4, 0] 8

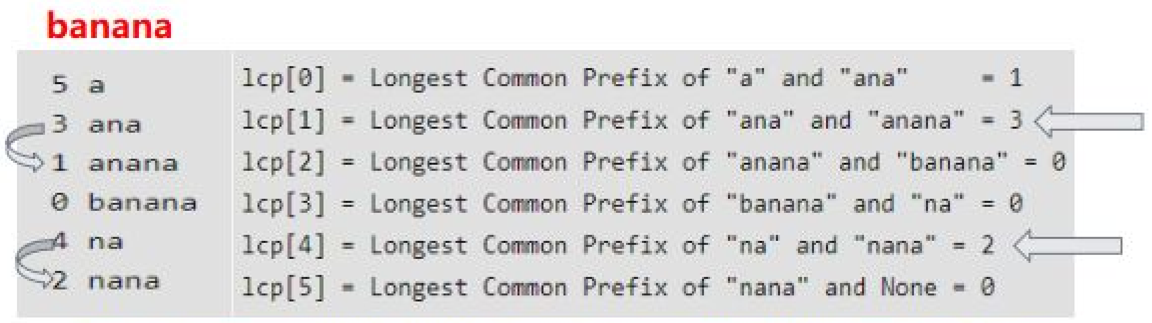
length

```
5 a        lcp[0] = Longest Common Prefix of "a" and "ana"      = 1
3 ana      lcp[1] = Longest Common Prefix of "ana" and "anana" = 3
1 anana    lcp[2] = Longest Common Prefix of "anana" and "banana" = 0
0 banana   lcp[3] = Longest Common Prefix of "banana" and "na" = 0
4 na       lcp[4] = Longest Common Prefix of "na" and "nana" = 2
2 nana     lcp[5] = Longest Common Prefix of "nana" and None = 0
```

https://www.geeksforgeeks.org/%C2%AD%C2%ADkasais-algorithm-for-construction-of-lcp-array-from-suffix-array/

Figure 5.7: Longest Common Prefix

After getting the longest common prefix array, we set the minimum length of common prefix and select the candidates. E.g. set 2 as minimum length. Then get the locations of each pair of clone code; you can see the group [3,1]and another group[4,2], whose length is bigger or equal to 2 (see Figure 5.8).


banana

```
5 a        lcp[0] = Longest Common Prefix of "a" and "ana"      = 1
3 ana      lcp[1] = Longest Common Prefix of "ana" and "anana" = 3
1 anana    lcp[2] = Longest Common Prefix of "anana" and "banana" = 0
0 banana   lcp[3] = Longest Common Prefix of "banana" and "na" = 0
4 na       lcp[4] = Longest Common Prefix of "na" and "nana" = 2
2 nana     lcp[5] = Longest Common Prefix of "nana" and None = 0
```

https://www.geeksforgeeks.org/%C2%AD%C2%ADkasais-algorithm-for-construction-of-lcp-array-from-suffix-array/

Figure 5.8: Select Candidates

For the location pair, each location add the result of minimum length minus 1 to get the terminal location. Then we get the clone code zone.

Common Prefix Location pair [3,1]-->(Each Add 2-1=1)

clone zone 1: [3,4] clone zone 2: [1,2]

Common Prefix Location pair [4,2]-->

clone zone 1: [4,5] clone zone 2: [2,3]

we combine overlapping clone zone to one. For string banana: the substring from 3 to 5 and substring from 1 to 3 has common prefix bigger than 2.

Combine overlapping clone zone to one:

[3,4] [4,5] →[3,5]

[1,2] [2,3] →[1,3]

banana: The substring pairs whose Longest Common prefix is bigger than 2.→[3,5][1,3]

(banana, banana)

5.4 Database design

Part 1 : database implementation
(1) We chose the PDO(PHP Data Objects) to connect MySQL database.
(2) Then we use CREATE DATABASE SQL sentences to create a database for our project.
(3) Then we use CREATE TABLE SQL sentences to create two data tables: one is to store upload .zip files, which are the assignments files; the other is to store the .zip results files which are PDF files.
(4) We use INSERT SQL sentences to insert files to data tables.
Part 2 : data tables introduction
The following shows the tables of the database:
(1) the files data table:
Id is the identity for each upload files, it is the primary key and foreign key, it is same in the results data table, and it begins from 1 with automatic increase. The file_name is the names of upload files. Time is the upload time for each upload files (shown in Table 5.2).

Table 5.2 The instruction of files data table

Field name	Field type	Primary key	Foreign key	empty	defaults
id	int(11)	yes	yes	no	
file_name	varchar(32)			yes	NULL
time	varchar(32)			yes	NULL

(2) the results data table:
Id is the identity for each results files, it is the primary key and foreign key, it is same in the files data table. and it begins from 1 with automatic increase. The file_name is the names of results files. Time is the upload time for each upload files (shown in Table 5.3).

Table 5.3 The instruction of results data table

Field name	Field type	Primary key	Foreign key	empty	defaults
id	int(11)	yes	yes	no	
file_name	varchar(32)			yes	NULL
time	varchar(32)			yes	NULL

5.5 User interface design

5.5.1 upload function

Part 1: upload function implement

(1) Create a data table in index.html file, which has a label for file, and a submit button.
(2) Then combine the data table with upload_file.php file. After click the submit button, the upload file will be post to upload_file.php file.
(3) In the upload_file.php file, it will get the filesize of upload files, and then judge the type and size of the files. If they are fit the standard of .zip file, then they will be continued process. The upload file will be unzipped, and get the locations of them.
(4) Convert the locations to Python part.
Part 2: upload function interface design
(1) In the main page, users can chose files from their computer, and then submit (shown in Figure 5.9).

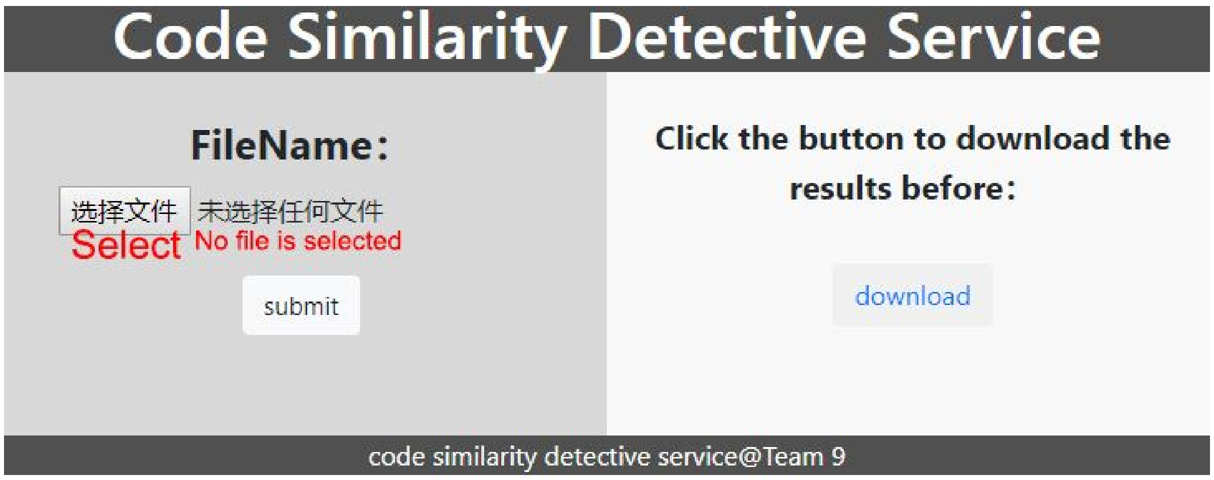
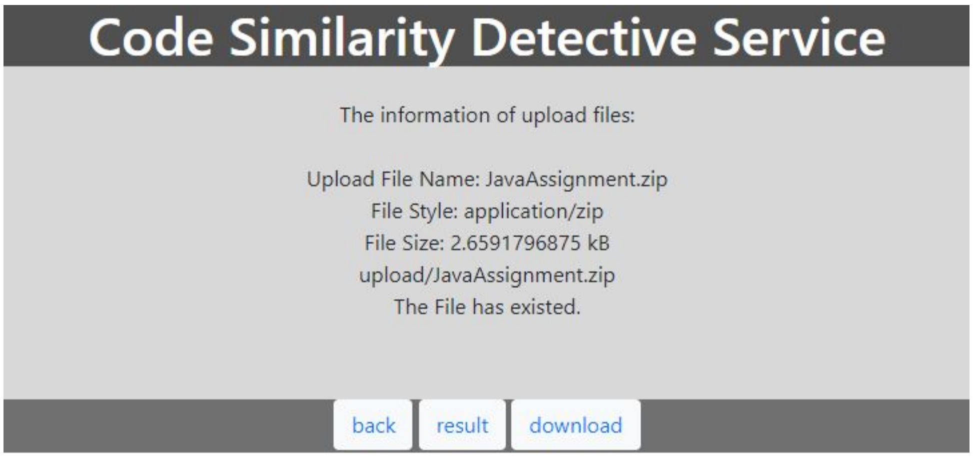

Code Similarity Detective Service
FileName:
选择文件
未选择任何文件
Select
No file is selected
submit
Click the button to download the results before:
download
code similarity detective service@Team 9

Figure 5.9: Main page

(2) Then it will jump to the next page, shown with the properties of files, and three buttons on the bottom. The back button is for going back to the home page. The result button is for jumping to the result page to read online result files. The download button is for showing the .zip result files (shown in Figure 5.10).



The information of upload files:

Upload File Name: JavaAssignment.zip
File Style: application/zip
File Size: 2.6591796875 kB
upload/JavaAssignment.zip
The File has existed.

Figure 5.10: Information page

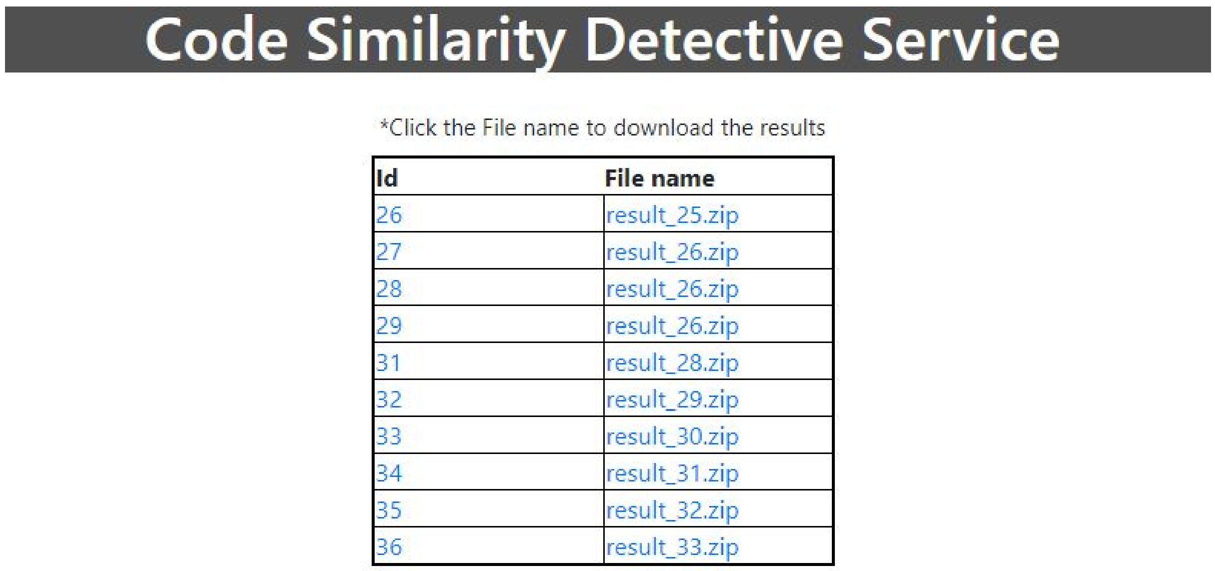
5.5.2 download function

Part 1: download function implement

(1) After getting the HTML result files, using PHP to turn them to PDF files, and zip them in a .zip file.
(2) After getting the result .zip file, using insert SQL sentences to store the result to results data table.
(3) In the download.php, we use the preprocessing syntax, to get the id, file_name fields from results data table, and put them into a HTML table.
(4) Click the file_name, then can download .zip result files.

Part 2: download function interface design

(1) The download interface is a table with id, file_name fields (shown in Figure 5.11).
(2) In the home page, users can click the download button to see the history results to download. And users can also download results from the download button on the properties of files page.



Code Similarity Detective Service

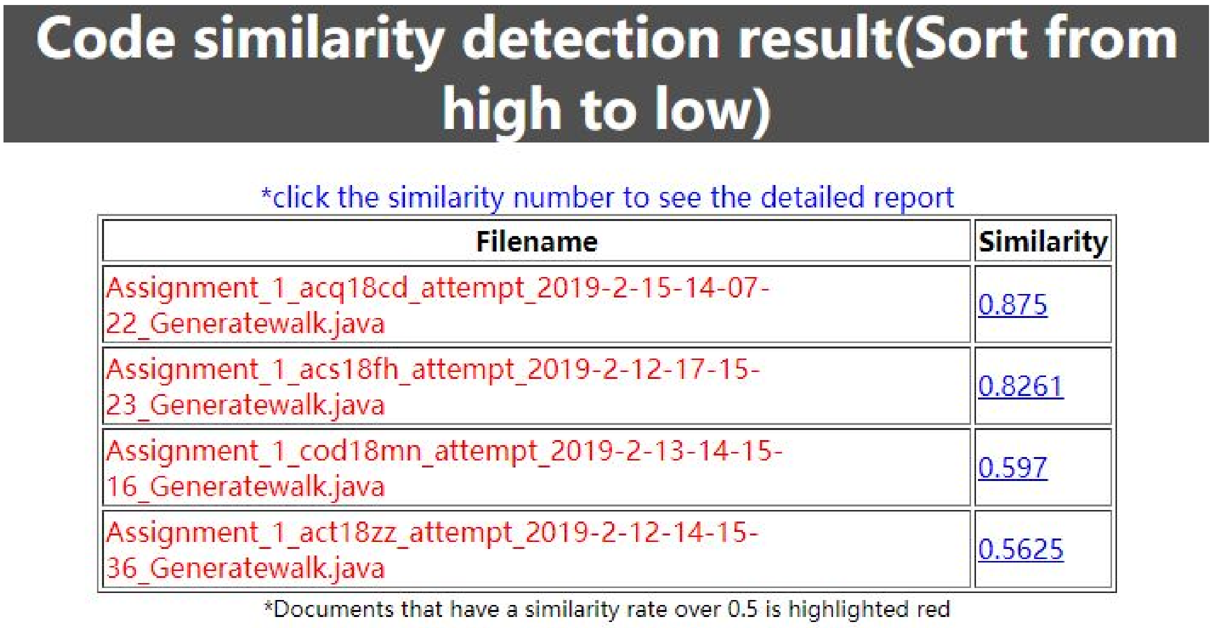
*Click the File name to download the results

Id	File name
26	result_25.zip
27	result_26.zip
28	result_26.zip
29	result_26.zip
31	result_28.zip
32	result_29.zip
33	result_30.zip
34	result_31.zip
35	result_32.zip
36	result_33.zip

Figure 5.11: Download information page

5.5.3 Results showing function(shown in figure 5.12 and figure 5.13)

(1)Result generated before is used to produce the html report in python.
(2)First, the similarity rate dictionary is sorted from big to small and then iterated to get the id of every file.
(3)The compare result can be found in the result list the according to file id. Among all the similarity rate results, rates over 0.5 will be highlighted red by adding colour attribute in html string.
(4)In order to show users both the similarity of all the files and the detailed report of each file online, the result report is designed to jump to the url of detailed report by clicking the rate number.
(5)After this, through merging the front end and back end by calling python in php, the html report will be shown for users online.The report can also be converted to pdf and zipped if users want to download it.



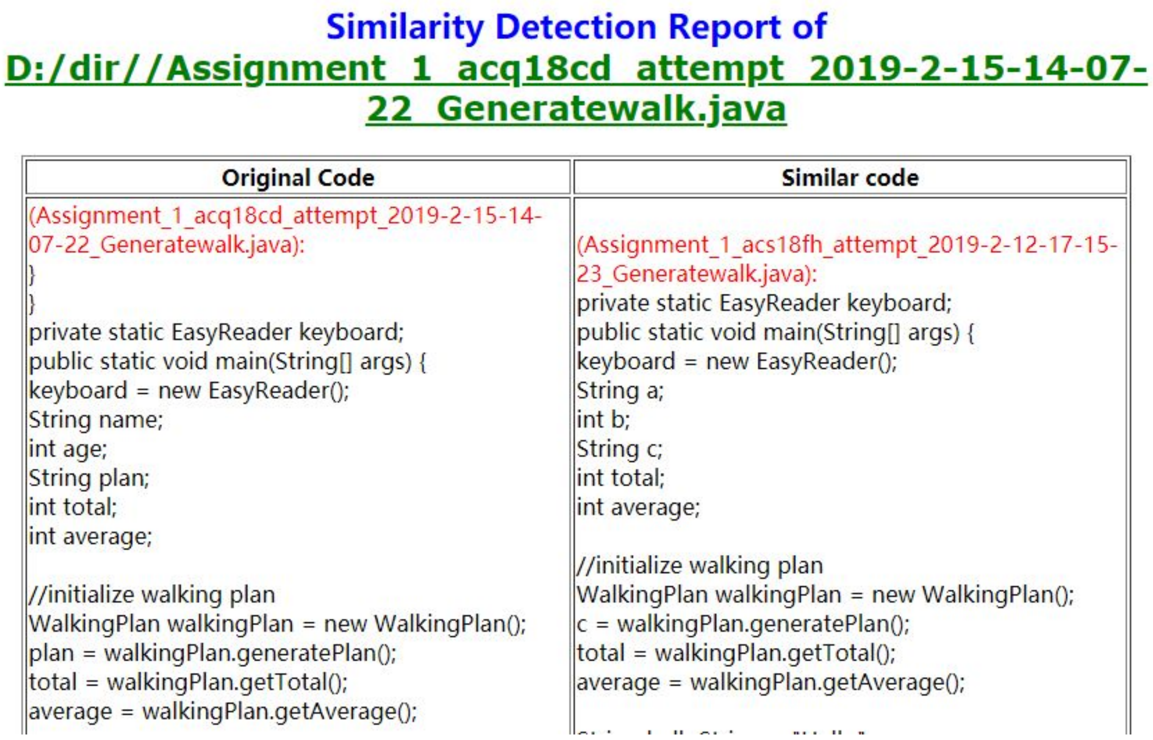
Code similarity detection result(Sort from high to low)

*click the similarity number to see the detailed report

Filename	Similarity
Assignment_1_acq18cd_attempt_2019-2-15-14-07-22_Generatewalk.java	0.875
Assignment_1_acs18fh_attempt_2019-2-12-17-15-23_Generatewalk.java	0.8261
Assignment_1_cod18mn_attempt_2019-2-13-14-15-16_Generatewalk.java	0.597
Assignment_1_act18zz_attempt_2019-2-12-14-15-36_Generatewalk.java	0.5625

*Documents that have a similarity rate over 0.5 is highlighted red

Figure 5.12: Similarity rate



Similarity Detection Report of D:/dir//Assignment_1_acq18cd_attempt_2019-2-15-14-07-22_Generatewalk.java

Original Code	Similar code

(Assignment_1_acq18cd_attempt_2019-2-15-14-07-22_Generatewalk.java):
}
}
private static EasyReader keyboard;
public static void main(String[] args) {
keyboard = new EasyReader();
String name;
int age;
String plan;
int total;
int average;
 	
//initialize walking plan
WalkingPlan walkingPlan = new WalkingPlan();
plan = walkingPlan.generatePlan();
total = walkingPlan.getTotal();
average = walkingPlan.getAverage();
	(Assignment_1_acs18fh_attempt_2019-2-12-17-15-23_Generatewalk.java):
	private static EasyReader keyboard;
	public static void main(String[] args) {
	keyboard = new EasyReader();
	String a;
	int b;
	String c;
	int total;
	int average;
	
	//initialize walking plan
	WalkingPlan walkingPlan = new WalkingPlan();
	c = walkingPlan.generatePlan();
	total = walkingPlan.getTotal();
	average = walkingPlan.getAverage();

Figure 5.13: Detailed report

6. Test plan

The process of test plan can be divided into three parts: algorithm testing, database testing and testing of the whole programme.

1. Algorithm testing(shown in table 6.1).

Table 6.1: Algorithm testing

Function	Precondition	Operation	Test cases	Expected results
Detecting similar parts between two code	Successfully submitted code	Use Token Comparison algorithm to check code	Two code exactly the same as the other	The algorithm will find two code are same
Detecting similar parts between two code	Successfully submitted code	Use Token Comparison algorithm to check code	Two completely different code	It cannot detect any available difference
Detecting modified plagiaristic code	Successfully submitted code	Use Token Comparison algorithm to check code	An original code and the other code modified structure of code	It can still detect plagiaristic code
Detecting modified plagiaristic code	Successfully submitted code	Use Token Comparison algorithm to check code	An original code and the other code rename all variables	It can still detect plagiaristic code
Detecting modified plagiaristic code	Successfully submitted code	Use Token Comparison algorithm to check code	An original code and the other code adjusted sequence of words and format	It can still detect plagiaristic code
Generating a rate of similarity	Successfully detected plagiaristic code	Use Token Comparison algorithm to generate similarity	Two code exactly the same as the other	The rate of similarity is 100%
Generating a rate of similarity	Successfully detected plagiaristic code	Use Token Comparison algorithm to generate similarity	Two completely different code	The rate of similarity is so low that may be 0%
Highlighting rate over 50% in red	Generated a rate of similarity	Distinguish rate whether it is more than a half or not	A code of rate of similarity less than 50%	The number uses default colour
Highlighting rate over 50% in red	Generated a rate of similarity	Distinguish rate whether it is more than a half or not	A code of rate of similarity more than 50%	The number will be red

2. Database testing(shown in table 6.2).

Table 6.2: Database testing

Function	Precondition	Operation	Test cases	Expected results
Submitting a .zip file	Database works well	Submit a .zip file in homepage	A normal .zip file which size is several KB	Store code in server and save path of code in database
Submitting a large .zip file	Database works well	Submit a .zip file in homepage	A large .zip which is about 1MB and includes many code	Store code in server and save path of code in database
Displaying a warning when submit a wrong type file	Submitting function works well	Submit some files	Some other types files that not .zip and a .zip file does not include code	The program will not store the file and give a warning
Judge a code if it is in database or not	Submitting function works well	Submit a .zip file	A .zip file that has been uploaded before	It will not be stored in database
Download reports	Generated .pdf reports and are packaged in a .zip file	Download reports in history	Some different sizes of code in .zip packages	All reports that represent every submits can be downloaded

3. Integration testing(shown in table 6.3).

Table 6.3: Integration testing

Function	Precondition	Operation	Test cases	Expected results
Transmission parameters	Code has been stored in server and the database	Submit several .zip files in homepage	Some different sizes of code in .zip packages	All code are transmitted to background to process
System runnable environment	The front application and the background works well	Run the program in different browsers	Chrome and FireFox (In Windows)	It can be run in different browsers

7. Team management & communication

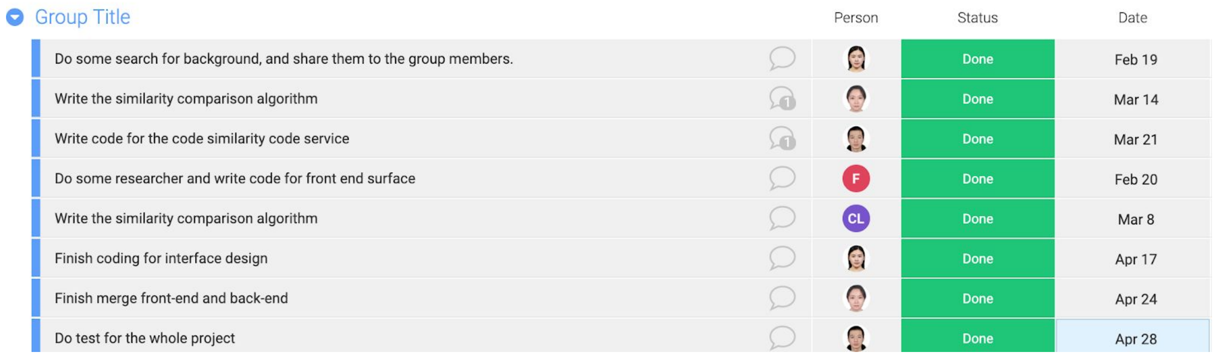
1.Team management

(1) Division of work: we divided our team members into 2 parts. Team 1 mainly focus on database design and front end interface design. Team 2 mainly focus on implementing code similarity detection algorithm(shown in Table 7.1 Division of work).

Table 7.1 Division of work

Division	Team member	Task
Part 1	Zixuan Zhang	Database implement, user interface implement, and merge python and PHP.
	Ruocheng Ma	Design the architecture of user interface and databases, build API and do software testing
	Sheng Huang	Read articles, learn relative knowledge and do some prepare work. (Left group since Week 6)
Part 2	Chenxi Liu	Design and implement the token extraction part and the report producing part of the algorithm; merge the different part of the algorithm and connected it to front end.
	Lu Wang	Design and implement an algorithm based on suffix-array for detecting the clone code.
	Luziyang Chen	Design and implement the preprocessing part and record the minutes of the meeting.

(2) Project management tool: we use the Monday tool to manage our project (shown in Figure 7.1).



Group Title

	Person	Status	Date
Do some search for background, and share them to the group members.		Done	Feb 19
Write the similarity comparison algorithm		Done	Mar 14
Write code for the code similarity code service		Done	Mar 21
Do some researcher and write code for front end surface	F	Done	Feb 20
Write the similarity comparison algorithm	CL	Done	Mar 8
Finish coding for interface design		Done	Apr 17
Finish merge front-end and back-end		Done	Apr 24
Do test for the whole project		Done	Apr 28

Figure 7.1: Project manegement tool

(3) Coding share: Git Lab, CodePile.

(4) Material share: Google Drive.

2.Team communicate

(1) Communicate tool: we use wechat to do normal chatting and discussing.

(2) Group meeting: we meet twice every week, and discussing the whole project.

3.Challenges we met

(1) Database design

At first, we chose JAVA as our coding language, and when we tried to build database and connected it to the whole project, there was a mistake, and we worked for 3 days, still can not solve this. So, we decided to use PHP to begin our design, and we did this successfully.

(2) Algorithm implement

We compared token comparison algorithm with two other code similarity detection algorithms before we started. One is called textual comparison and the other is called Comparison of Abstract syntax tree. Finally, we chose the token comparison algorithm out of the consideration of the complexity of algorithm and the completion of the function. However, it was quite hard to understand. Therefore, the team members tried to learn it by reading lots of professional articles and journals before coding. After many times of debugging, we finally succeeded to implement code similarity detection.

(3)Find the number of lines of similar code in source file

After we finished our system, we found it was hard for users to find out where are the similar code comes from, so we decided to present the lines of similar code in source file. At first we were confused. Then we tried to record the number of lines in preprocessing, it successfully solved this problem.

8. Completed Features

The features that our team have implemented are listed in the following table(Table 8.1).

Table 8.1: Completed features

Completed features	Description
Identity directly copied and code that change the variable names	The directly copied code, code that change the variable names, code that change the order and structure of the original code can be detected.
Web service for users to upload zip files	Users can click the ‘choose file’ button to select the files from users’ computer and submit them to the system.
Produces reports for each submission	Html reports are generated by python. The report can also be converted to pdf and zipped if users want to download it.
Create Database and two data tables	We use the two data form to store upload files and results files.
User interface	Create the user interface to guide users to use the functions of the whole system
Download PDF results	Users can download PDF results by click the

9. Uncompleted feature

The uncompleted features and the reason why our team did not finish them are all listed in the following table(Table 9.1).

Table 9.1: uncompleted features and reasons

Uncompleted features	Reasons
Expand the algorithm to detect different languages	lack of time
Identity potential advanced copied code	lack of time

10. Conclusion

The project finished all requirements. It has a concise interface which developed by PHP. Users choose and upload files through the home page, then download the result in the other page. Results will be provided in .pdf files in a .zip package. The back-end processing for the web application is handled with Python. It uses Token Comparison algorithm which works with Java. A special API is used to convey data from front application to the background to process.

The programme has several distinct features. The interface is not complex but it is believed both attractive and practical. All results can be viewed online. Also, they are saved in the server and all file paths of results are stored in the database, so users can always download all old results in another page. There is another webpage using for contrasting two code, which is used for showing plagiaristic parts between two code. Even a cheater changes the whole structure of code and renames all variables, plagiarism can still be detected. For each code the programme can give a rate of similarity. It is defined a rate more than 0.5 is probably plagiarized from others, which will be highlighted in red in the page.

However, there is room for improvement in the programme. At present, the application just can work with Java. We wish it can work with more languages like Python and Rude, which is the next step of work.

We acquired some experience thorough the project. For the one thing, it is the experience of programming. For example, when we tried to save codes and results in database, although just less than 2 MB, it is found that is still too large for a database. In this way, after a discussion, we decided to just save the paths of those files in database, then call them by paths. It is a useful experience for similar problems in the future. For the other thing, we can handle the skill of teamwork now. An event is good to mention is that each student has his or her own work. Nevertheless, not all of us are suitable for their works. At the beginning of the mission, all group members said their areas of expertise. After a long talk, everyone got their division of labour. However, as students, we had basic theories rather than enough practice, which led several team members could not do their contributions in plan and caused delay of planned work. It demonstrated the lack of true practice. Despite of that, we adjusted the plan as soon as possible. Although one team member left in March, other students still successfully took over his work as soon as possible. At last, although we lost one person, the team finished the basic work much earlier than most other teams.

Bibliography

1. Q,SHI. L,ZHANG. L,YIN. D,LIU. Clone Detection Based on Suffix Array[J].*Computer Engineering*, 2013,(39)123-130.
2. S. Ducasse, M. Rieger, S. Demeyer, "A Language Independent Approach for Detecting Duplicated Code", *Proc. Int'l Conf. Software Maintenance (ICSM '99)*, 1999.e
3. B.S. Baker, "A Program for Identifying Duplicated Code", *Proc. 24th Symp. Interface*, pp. 49-57, Mar. 1992.
4. T. Kamiya, S. Kusumoto, K. Inoue, "CCFinder: A Multi-Linguistic Token-Based Code Clone Detection System for Large Scale Source Code", *IEEE Trans. Software Eng.*, vol. 28, no. 7, pp. 654-670, July 2002.
5. E. McCreight, "A Space-Economical Suffix Tree Construction Algorithm", *J. ACM*, vol. 32, no. 2, pp. 262-272, 1976.
6. Y. Higo, Y. Ueda, T. Kamiya, S. Kusumoto, K. Inoue, "On Software Maintenance Process Improvement Based on Code Clone Analysis", *Proc. Int'l Conf. Product Focused Software Process Improvement*, pp. 185-197, 2002.
7. J. Mayrand, C. Leblanc, E.M. Merlo, "Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics", *Proc. Int'l Conf. Software Maintenance*, pp. 244-254, Nov. 1996.
8. Yongchang Ren, Deyi Jiang, Tao Xing, Ping Zhu, "Research on software development platform based on SSH framework structure", *Procedia Engineering*,Volume 15, 2011, Pages 3078-3082
9. What is JSP? Introduction to JavaServer Pages: Get an overview of JavaServer Pages, then write your first JSP page that connects with a Java servlet and deploys on Tomcat, Tyson, Matthew.JavaWorld; San Francisco (Jan 29, 2019).
10. Sodhi, Pinky and Awasthi, Naman and Sharma, Vishal, Introduction to Machine Learning and Its Basic Application in Python (January 6, 2019). Available at SSRN: https://ssrn.com/abstract=3323796 or http://dx.doi.org/10.2139/ssrn.3323796
11. Machlis, PHP, *Computerworld.*, Vol. 36 Issue 6, p46. 1p, April 2002