

# Plagiarism Detection Tool

CS 5500 - Managing Software Development

**Team 206**

**Members:**

Asim Khan

Sanket Mathur

Jiaxin Yang

Jayanth Gangadhar

# Agenda

**System Functionality** - Mohd Asim Khan

**Job Quality** - Sanket Mathur

**Process and Teamwork** - Jiaxin Yang

**Technology Transfer** - Jayanth Gangadhar

# System Functionality: Requirements

## **Professor Weintraub**

- Build an application that detects plagiarism
- Include different strategies for comparison
- Weighted polynomial functional to incorporate all strategies
- Different types of submissions allowed (Folders/Files)
- Customised plagiarism threshold value which can be changed on the Users will
- Text highlighting to show similar code

## **Professor Annunziato**

- A portal for Students, Professor, Admin
- Get a taste of Student management system
- Automated report generation system with integrated notification Emails
- Good UX / Least user interaction
- Integrated logging for important events (Registration, Plagiarism detection, etc)

# System Functionality: Goals Achieved

We succeeded to build our project based on both the requirements published

- 3 portals to support Students, Professors and Admins
- Developed 3 Comparison Strategies using ASTs to check for Code Similarity
- A dedicated CRON Job to automatically run everyday to check for new submissions for comparison in an iterative manner
- A great UI which highlights code similarity with even minute details
- Automated Emails integrated with the application to send real time notifications
- Most advanced technologies used while development - AOP, Springs, MongoDB, Angular 4, AWS S3, Gmail API with OAuth2, Mockito, Sonar.

# Application Usability

- Intuitive User Interface
- Multiple functionalities available for the user
- Automated as well as Manual Support for Plagiarism Detection
- Automated Email notifications
- Admin portal for God access to all users
- Student portal for course enrollment and homework submission

Home Compare ▾ History

Log out

Plagiarism detected!

Threshold used: 40

Weighted average score: 98.75%

Metric type	Score
Zhang-Shasha Algorithm	100
Longest Common Subsequence Algorithm	100
Line Similarity Algorithm	93.75

File one: dice.py

```
1 # Script Name : dice.py
2 # Author : Craig Richards
3 # Created : 05th February 2017
4 # Last Modified :
5 # Version : 1.0
6
7 # Modifications :
8
9 # Description : This will randomly select two numbers, like throwing dice, you can change the sides of the dice.
10
11 import random
12 class Die(object):
13     # A die has a feature of number about how many sides it has when it's established, like 6.
14     def __init__(self):
15         self.sides=6
16
17     """because a dice contains at least 4 planes.
18     So use this method to give it a judgement when you need to change the instance attributes."""
19     def set_sides(self, sides_change):
20         if sides_change <= 6:
21             print("change sides from 6 to ",sides_change, "!")
22         else:
23             # added also clause for printing a message that sides set to 6
24             print("sides set to 6")
25         self.sides = sides_change
26     else:
27         print("wrong sides! sides set to 6")
28
29
30
```

File two: dice.py

```
1 # Script Name : dice.py
2 # Author : Craig Richards
3 # Created : 05th February 2017
4 # Last Modified :
5 # Version : 1.0
6
7 # Modifications :
8
9 # Description : This will randomly select two numbers, like throwing dice, you can change the sides of the dice.
10
11 import random
12 class Die(object):
13     # A die has a feature of number about how many sides it has when it's established, like 6.
14     def __init__(self):
15         self.sides=6
16
17     """because a dice contains at least 4 planes.
18     So use this method to give it a judgement when you need to change the instance attributes."""
19     def set_sides(self, sides_change):
20         if sides_change <= 6:
21             print("change sides from 6 to ",sides_change, "!")
22         else:
23             # added also clause for printing a message that sides set to 6
24             print("sides set to 6")
25         self.sides = sides_change
26     else:
27         print("wrong sides! sides set to 6")
28
29
30
```

Home

Course

History

Asim Professor

Email: asim.baw17790@gmail.com

Role: Professor

My courses

Course assignment

Profile

Courses

Managing Software Development

Algorithms

Assignments

Assignment Name	Edit
Homework 1 - Design Patterns	9
HW2 - Functional Testing	0

Create Assignment

Assignment: Homework 1 - Design Patterns

Deadline: 2018-04-28

Document link: <https://docs.google.com/document/d/1CHaKOf3TRJ3MAUCS9Wb3qz7eWwGd-69tBQ1Ys/edit>

Submission type: FILE

[View submissions](#)

First Name:

Asim

Last Name:

Professor

Plagiarism threshold: 30

Set how tolerant to plagiarism. Files whose similarity scores are above this threshold will be marked as plagiarism.

Save

Log out



Home
Compare
History
Log Out

# Plagiarism detected!

Threshold used: 40  
Weighted average score: 98.75%

Metric type	Score
Zhang-Shasha Algorithm	100
Longest Common Subsequence Algorithm	100
Line Similarity Algorithm	93.75

```

File two_dice.py
# Script Name : dice.py
# Author      : Craig Richards
# Created     : 08/01/2017
# Last Modified : 
# Version     : 1.0

'''
This script will generate random numbers, roll them and print the results.
'''

import random
class Roll(object):
    def __init__(self):
        self.sides = 6
        self.roll()

    def roll(self):
        """Roll the dice and return the result"""
        self.sides = 6
        self.roll()

    def __str__(self):
        return "The dice has been rolled %d times and the result was %d" % (self.roll(), self.result)

if __name__ == '__main__':
    r = Roll()
    print(r)
    while True:
        choice = input("Press enter to roll the dice or q to quit")
        if choice == 'q':
            break
        else:
            r.roll()
            print(r)

```

Date	User	Action
19-04-2018 03:37:15	User	
19-04-2018 03:36:46	User	
19-04-2018 01:17:05	System	
19-04-2018 01:03:09	System	
19-04-2018 01:03:04	System	
18-04-2018 23:36:23	User	
18-04-2018 23:32:55	User	
18-04-2018 23:32:14	User	

# Backlogs

- ❑ ⬆ CS206-93 Large file comparison failed
- ❑ ⬇ CS206-95 Rest APIs are public
- ❑ ⬆ CS206-130 Trace appears in UI
- ❑ ⬆ CS206-131 When someone logs out, new user when clicks on login, he can see previous user's credentials.
- ❑ ⬆ CS206-143 No tests for MailService class.
- ❑ ⬆ CS206-144 Only one test for FileServiceImpl, all the edge cases should be covered.
- ❑ ⬆ CS206-146 Even if user logs out, still he/she can access all the functionalities.
- ❑ ⬆ CS206-153 If user go back to the main page, user will automatically log out
- ❑ ⬆ CS206-154 user cannot change his password
- ❑ ⬆ CS206-156 Some error happen when refresh the page

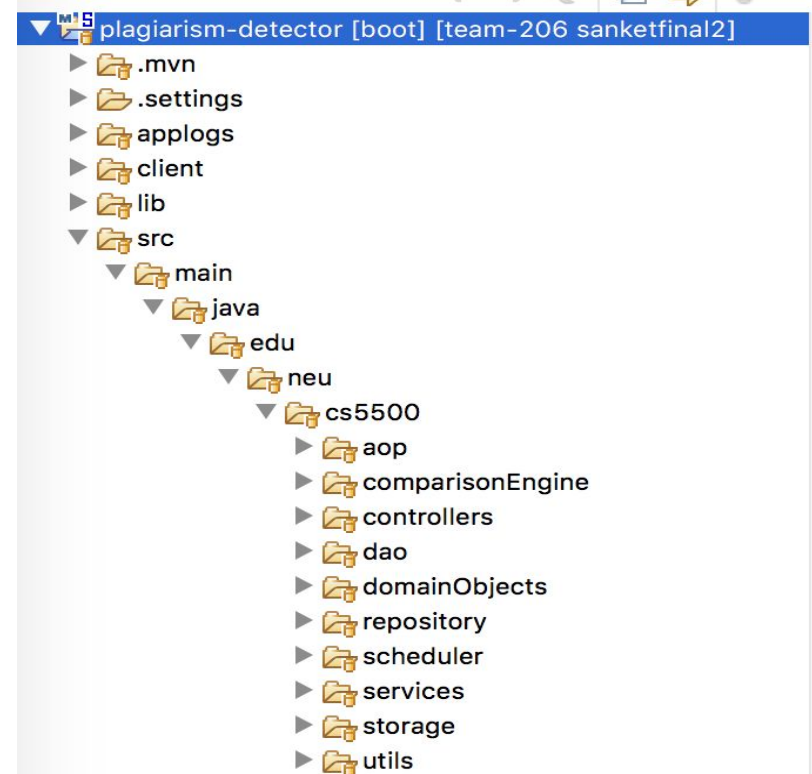
# Job Quality

- Followed Model View Controller architecture to provide proper structure to the backend system
- Followed quality standards by using SonarLint to maintain the code quality of the application
- Unit test coverage code has been set to at least 85% which includes instructions and branch covered
- Intuitive and simple user interface with color schemes to highlight plagiarism detected files
- Followed Spiral Model to build the UI. Earlier sprints provide very basic functionality and later sprints and final product have richer implementations.



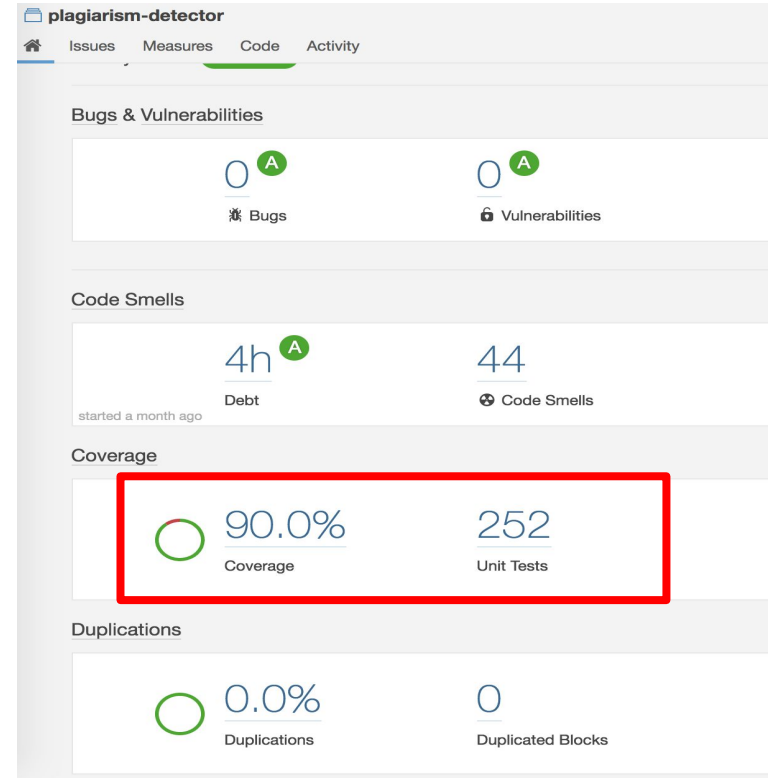
# Job Quality: Model View Controller Architecture

- All the controllers will be have the apis which will be used to interact with the front end
- The business logic is present in the service layer
- All the database related work tasks are present in the Data Access Objects (dao) layer



# Job Quality: Test coverage

- Lower limit of the test coverage has been set to 85% for the project
- Maintained to cover 90% of the code under test
- 252 total test written



# Job Quality: Intuitive UI

- Well organized and simple UI
- No prior knowledge of the system is required to complete their tasks
- Easy to identify the steps users need to complete their task
- Tasks are accessible based on the role of user: Admin, Professor or Student

The image displays two screenshots of a user interface, likely for a system where users can register and login. The top screenshot shows a 'Register' form with the following fields: 'First Name', 'Last Name', 'email', 'password', and 'verify password'. Below these fields is a 'Role' dropdown menu. At the bottom of the form are two buttons: a red 'Close' button and a blue 'Register' button. The bottom screenshot shows a 'Login' form with two fields: 'email' and 'password'. At the bottom of this form are two buttons: a red 'Close' button and a blue 'Login' button. Both forms have a close button (an 'x' icon) in the top right corner.

# Job Quality: Intuitive UI

## Compare Type: FOLDER

### Student List

Student 1	Student 2	Result
S1 Linda	Issac Newton	No Plagiarism
S1 Linda	Sheldon Cooper	No Plagiarism
Issac Newton	Sheldon Cooper	Plagiarism Found! <a href="#">Notify students</a>

### File List

Threshold used: 20

File 1	File 2	Average Score	View Detail
package1/File1.py	project2/package2/package3/File1.py	100	<a href="#">View Detail</a>
package1/File2.py	project2/package2/package3/File2.py	100	<a href="#">View Detail</a>
package1/File1.py	project2/package2/package3/File2.py	20.47	<a href="#">View Detail</a>
package1/File2.py	project2/package2/package3/File1.py	20.47	<a href="#">View Detail</a>
package1/File1.py	project2/dice.py	15.45	<a href="#">View Detail</a>
package1/File2.py	project2/diceV2_dynamic.py	15.15	<a href="#">View Detail</a>
package1/File2.py	project2/dice.py	13.08	<a href="#">View Detail</a>
package1/File1.py	project2/diceV2_dynamic.py	10.05	<a href="#">View Detail</a>

# Job Quality: Intuitive UI

Plagiarism detected!

Threshold used: 40

Weighted average score: 53.74%

Metric type	Score
Zhang-Shasha ALgorithm	47.73
Longest Common Subsequence Algorithm	42.19
Line Similarity Algorithm	83.33

File one: text1.py

```
1 class Solution:
2     def groupAnagrams(self, strs):
3         dict = {}
4         iterate(strs)
5
6     def iterate(strs):
7         for i in strs:
8             sort = ''.join(sorted(i))
9             check(dict,sort,i)
10        return list(dict.values())
11
12    def check(dict,sort,i):
13        if dict.get(sort) == None:
14            dict[sort] = [i]
15        else:
16            dict[sort].append(i)
17
18
```

File two: text2.py

```
1 class Solution:
2     def groupAnagrams(self, strs):
3         dict = {}
4         for i in strs:
5             sort = ''.join(sorted(i))
6             if dict.get(sort) == None:
7                 dict[sort] = [i]
8             else:
9                 dict[sort].append(i)
10        return list(dict.values())
11
```



# Job Quality: Team Performance

CS206 board

Cumulative Flow Diagram [Switch report](#)

Board



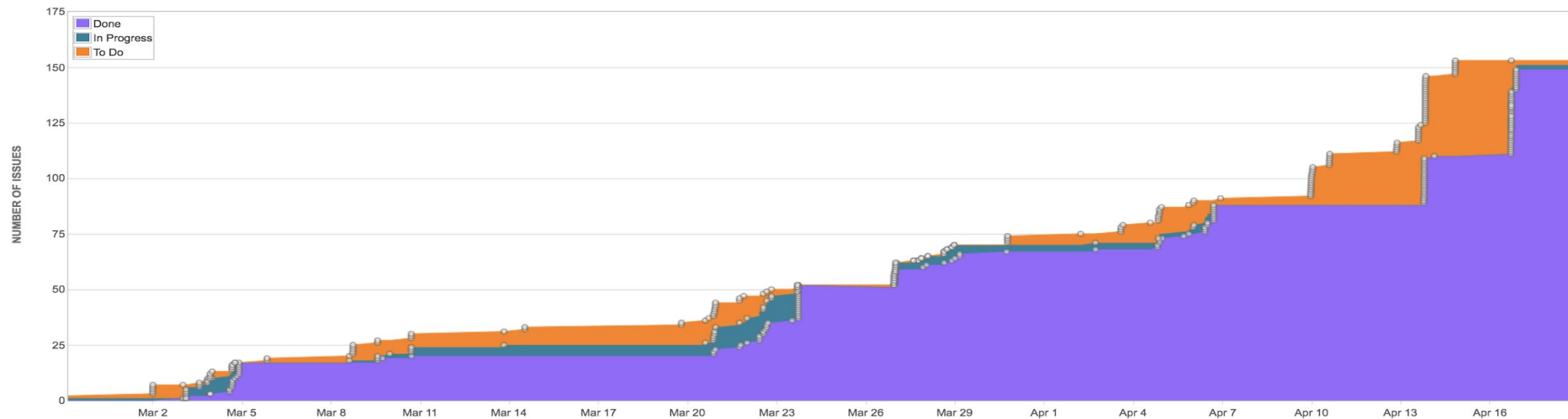
## How to read this chart

Shows the statuses of issues over time. This helps you identify potential bottlenecks that need to be investigated.

[Hide this information](#)

26/Feb/18 to 18/Apr/18 (All Time)

[Refine report](#)



# Process and Teamwork

## Three sprints

- Discuss, identify and prioritise use cases
- Create stories in Jira and assign tasks to each member
- Define apis and interfaces
- Communicate in person and via WhatsApp
- Integrate the project at the end of Sprint

## Automatic deployment

- Jenkins builds on every change
- Code quality is checked on every pull request
- Automatic deploy on every successful pull request

# Process and Teamwork

## Two challenges

- Different time frames for every teammate
  - Program to interface
  - Communicate effectively
- Frequently changed requirements
  - Be prepared for changes
  - Be selective to features



# Technology Transfer

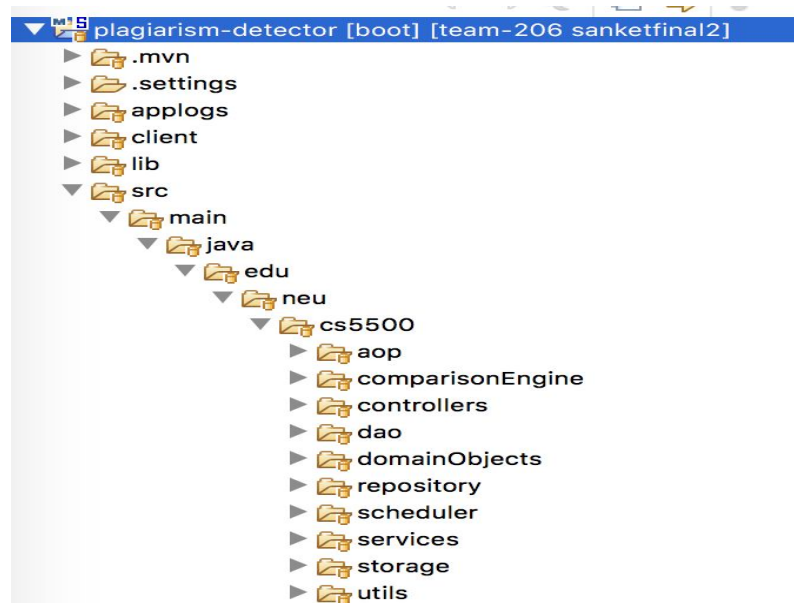
## Obstacles With Technology transfer

- Strict constraints of time
- Incomplete transfer of knowledge
- Lack of function recognition.
- Team members from a completed project are usually needed for the next project, and their new team leaders therefore must recruit them into new teams as soon as possible.

# Solution:

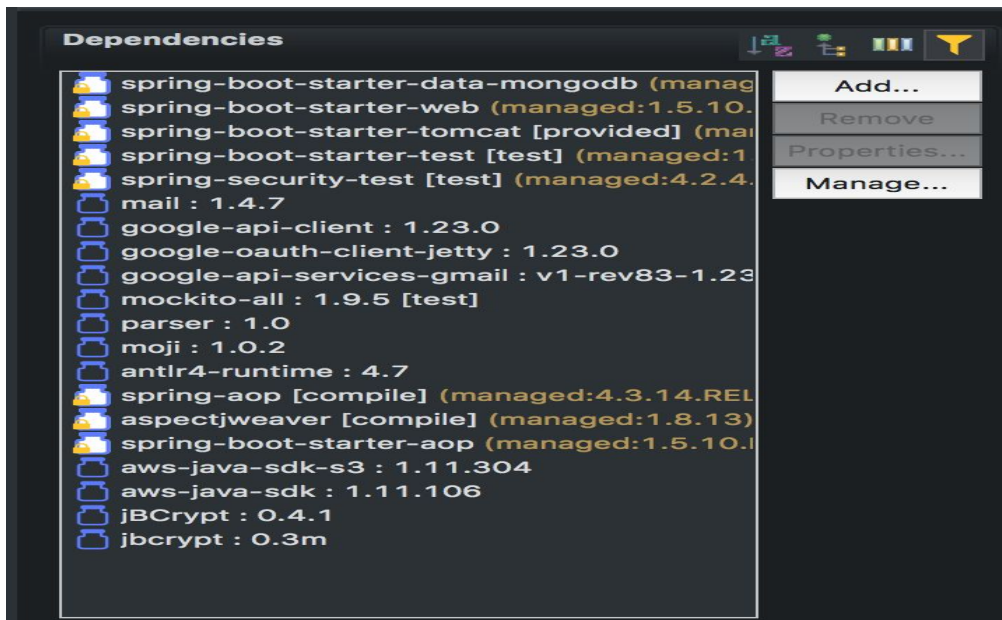
- Well documented code so it becomes easier for the future team.
- Followed MVC architecture, Hence proper organization of code and good readability.

```
2
3 import java.io.File;
4 /**
5  * Context is a class that uses Strategy
6  * @author JayanthGangadhar
7  *
8  */
9 public class Context {
10
11     private Strategy strategy;
12     //Constructor
13     public Context(Strategy strategy){
14         this.strategy = strategy;
15     }
16 /**
17  * execute Strategy demonstrates change in Context behavior based on strategy
18  * @param f1 first file
19  * @param f2 second file
20  * @return similarity of the two files
21  */
22     public double executeStrategy(File f1, File f2){
23         return strategy.compare(f1, f2);
24     }
25 /**
26  * To return the name of strategy
27  * @return strategy name
28  */
29     public String getStrategy() {
30         return strategy.getStrategy();
31     }
32 }
```



# Scalability

- Implemented Aspect Oriented Programming (AOP) which serves to provide cleaner code.
- System is a spring boot application. Adding of a new service is simplified due to dependency injection.



# Solution

- Followed strategy pattern and hence following open/closed principle.
- Makes it easy for future development to add add/update existing strategies.

```
2
3 import java.io.File;
4 /**
5  * Context is a class that uses Strategy
6  * @author JayanthGangadhar
7  *
8  */
9 public class Context {
10
11     private Strategy strategy;
12     //Constructor
13     public Context(Strategy strategy){
14         this.strategy = strategy;
15     }
16     /**
17     * execute Strategy demonstrates change in Context behavior based on strategy
18     * @param f1 first file
19     * @param f2 second file
20     * @return similarity of the two files
21     */
22     public double executeStrategy(File f1, File f2){
23         return strategy.compare(f1, f2);
24     }
25     /**
26     * To return the name of strategy
27     * @return strategy name
28     */
29     public String getStrategy() {
30         return strategy.getStrategy();
31     }
32 }
```

```
1 package edu.neu.cs5500.comparisonEngine.strategy;
2 import edu.neu.cs5500.comparisonEngine.ast.*;
3 /**
4  * Strategy1 corresponds to the Zhang Sasha Algorithm used to generate a similarity between ASTs
5  * @author JayanthGangadhar
6  */
7 public class Strategy1 implements Strategy {
8     //To store the name of the strategy
9     String strategy;
10    //To get similarity as per the strategy
11    double metric;
12    /**
13    * implements compare methods to compare similarity of two files
14    */
15
16    // Logger
17    private static Logger logger = Logger.getLogger(Strategy1.class.getName());
18
19    public double compare(File f1, File f2) {
20        //represents the similarity of the two files in term of '%'
21        metric=-1;
22        //To Parse the first file
23        ParserFacade parserFacade = new ParserFacade();
24        AstPrinter astPrinter = new AstPrinter();
25        //To Parse the second file
26        ParserFacade parserFacade2 = new ParserFacade();
27        AstPrinter astPrinter2 = new AstPrinter();
28    }
```

# Future Work/ Recommendation

- Application can be trained using some of the best currently available tools such as MOSS.
- Implement session management to better serve the user.
- Implement a single university sign on.
- Update system to compare submissions across different courses and also different semesters.
- Add more strategies to detect plagiarism.