Overview of the problem

Plagiarism Detection:

High Level Objective: The goal of the project is to build an interactive tool for detecting plagiarism in programming assignments.

Detailed Objective: The goal of this project is to design, implement, test, and evaluate an application to help instructors detect situations where two or more students submit similar solutions to an assignment, in which one version derives from another version through one or more behavior-preserving transformations. Examples of such transformations are moving functions or methods to another location in the same file, renaming variables, classes, and methods, extracting sequences of statements into methods, etc. In general, the nature of these transformations depends highly on the programming language in which the assignments are written. Therefore, different techniques will be required in plagiarism detectors for different programming languages.

Few Problem Scenarios:

1. Student copies entire homework from some other student with comments like @author present after copying.
2. Student copies entire homework with variable names changed.
3. Student copies homework but swap function positions.
4. Student copies partially.

Pre-requisites for plagiarism detection:

1.Students submit their homework in git-hub accounts for their course.

2. Students submit their homework in python programming language.

3. The course structure in github follows the current structure for CS5500 course.

CS5500---parent directory

Student-1--- sub-directory

Hw1 --- homework in student directory

Requirements :

1. The software designed should be able to detect plagiarism, plagiarism can be reported as percentage as in MOSS.
2. The software should be able to segregate plagiarism cases according to severity.
3. The software should provide significant proof of plagiarism in two files. This can be either highlighted portions of common code in plagiarised files, code snippets which are similar.
4. The software should be able to find plagiarism in students' homeworks containing multiple files.
5. The application should be interactive and should provide some options to download or get plagiarised case content to be used by professor for examining severe plagiarised cases.

Retrospective of the project

- ✓ Best part about the project is that there is a lot to learn – we were given an opportunity to develop a full stack application from start to end – defining our own development path.

- The only drawback is that it is time consuming – it was a difficult to balance the requirements of the project with other courses.

Learnings from the project:
The project gives entire overview of Software Development Life Cycle using Scrum methodology.

Scrum Methodology:
The team worked in sprints which consist of work units that are required to achieve a requirement defined in the backlog. Sprint allows us to work in short-term but stable environment.

Project Management:
The project gives us knowledge of working with Jira which is a project management, bug tracking and issue tracking tool.

Code Quality Management and Continuous Integration:
The project made use of continuous integration tools like Jenkins and Code quality management tools like Sonarqube. Sonarqube helped us in continuously improving the quality of our code by detecting bugs, code smells and security vulnerabilities.

Apart from the above-mentioned tools and methodologies, the project provided us opportunities to learn new technologies during the sprint requirements.

1. Design Patterns: The project helped us in exploring various object-oriented concepts using Design Patterns like Strategy design patter, factory design pattern etc.
2. Antlr : The project helped us in gaining good knowledge in using antlr jar for language parsing and converting into abstract syntax trees.
3. jsPDF and html2canvas: We can create pdf on the fly using jsPDF and capture screenshot using html2canvas.
4. MOSS: Accessing MOSS using socket client.
5. Jsoup: To read any html and get elements from html.
6. Singular Value Decomposition: For weight training of algorithms like LCS, EditDistance and AST strategy using MOSS data.
7. JGit: To extract contents from git hub.
8. AWS: The project is hosted on AWS instance.
9. MySQL, java, jsp, html, jquery, css are some of the technologies used in the project.

Suggestions in course to support a great experience:

1. Dev-ops configuration aspects of the project can be reduced if possible and focus on implementation can be enhanced – enjoyed the challenge of HW3

2. Make the bug hunt more interactive and interesting just a bug hunt – the present setup caused a number of irrelevant issues to be posted by testers since there was no clear demarcation/rules to define a bug/suggestion

An Overview of the team's development process

TEAM DEVELOPMENT PROCESS(Sprints - AGILE):

1. Log story tickets for tasks related to sprint requirements in Jira. We made sure that each team member is assigned equal number of tickets if possible.
2. Logged extra tasks in Jira as and when needed – splitting a large task into smaller ones or adding new functionality.
3. We used to discuss current task status of sprint on WhatsApp, Slack and held face to face meetings as and when required (Scrum)
4. We used smart commits while commiting code to a branch – thus linking Jira tasks to code commits
5. We used to help each other if we faced roadblocks through pair programming.
6. Wrote unit tests of all the possible scenarios of the functionality that we implemented to made sure functionality is working fine.
7. We pushed code to a branch created off master and confirmed through the checks enforced on Github that the build passed on Jenkins and the code quality check on Sonarqube passed as well .
8. Team member used to review code after all checks passed before merging changes to master - a developer's code on the branch could only be merged into master if he/she received at least one approval from the other team members.
9. The issues reported as bugs were triaged as professor suggested and were assigned to team members who had worked on the functionality in question or they wished to take it up out of interest

What worked?

- ✓ As a team we were able to complete all the stretches and Sprint requirements
- ✓ Since we were a three member team, we faced lesser communication difficulties – we were able to coordinate timings and assign tasks with each person's approval
- ✓ This was a fantastic opportunity to get an understanding of the software engineering process and we were able to adapt to the code quality and functional requirements efficiently
- ✓ Other than knowledge in front end technology – each of us had a good grasp over the Java programming language and were able to communicate with each over issues faced in the code

We thought of few extra functionalities apart from Sprint requirements like sending mail to OSCCR, view reports of previous test runs, that we are not able to complete, so we marked them as future enhancements.

What didn’t work?

We faced a lot of issues during the development process. Below is a list of the issues faced:

1. We struggled with configuring Sonarqube to our system. We initially had a different maven project structure due to which tests were not getting detected in Sonarqube. We fixed the issue later after recognizing the rootcause. Debugging and configuring the issue took up a lot of our development time
2. None of the team members is proficient in sound front-end technology so we had to stick to JSPs – we would have loved to take some more time and learn, implement Angular – but we focused on the algorithms, strategies and functionality over the look
3. We took some time to implement test driven development and often faced sonarqube quality errors

   For example - Using JSP and javascript, we used controllers, it was not possible to test controllers and quality check on sonarqube was getting failed, so after discussion with Prof. Jose we moved the logic in controllers to services and tested services
4. We were unable to visualize/foresee a couple of important design decisions that should have been made early in the development process – thus causing a frantic sprint towards the end of the two weeks to meet requirements

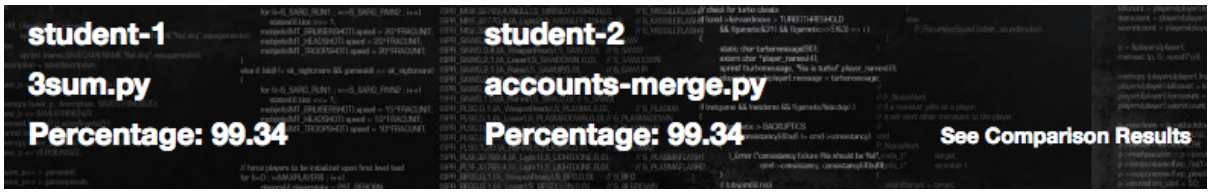
An overview of the result (1 page), with completion claims supported by backlog statistics and quality claims supported by testing data

This application is aimed towards detecting and reporting plagiarism between student submissions for a particular homework in a course. Let's go through the primary focus behind the application and the result obtained from it – with the help of its users, this application works towards processing the submissions made by students for a homework and displays the results in a report that lists the plagiarism cases detected and statistics that support the same.

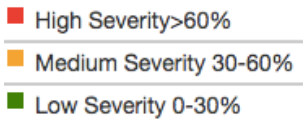
The report is a visualization of the instances(cases) of plagiarism detected by the system sorted by severity. Each instance is a pair of file submissions with the following metadata

- The ids of the students who have submitted the files in contention
- The file names of students that have been detected as similar/plagiarized
- The plagiarism percentage for each file (Each file is evaluated against the other and the percent reported is subject to the percent of the file that is plagiarized – thus two percentages – one for each file

An example of a plagiarism case reported by the system with its metadata is given below -

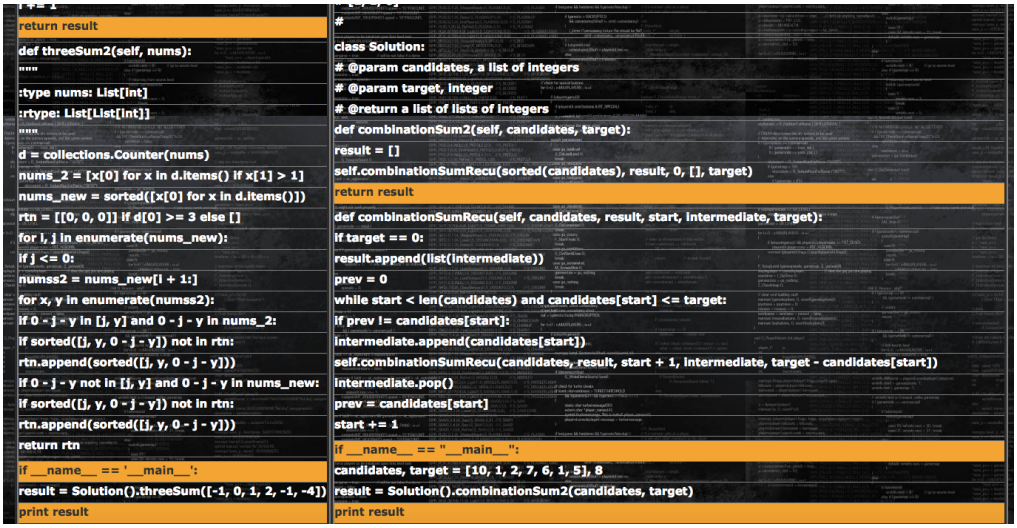


The report generated by the system is a list of cases like the one above sorted by the level of plagiarism detected in the files. This arrangement would assist a professor or teaching assistant to take a look first at the most serious cases of similarity between two students' files for the assignment. We have configured the severity levels as the following ranges in percentages:



Thus, the report is a display of lists of cases in each category/level of severity in the order of high, medium followed by low. The color associated with each list gives the indication of whether the issue must be addressed and reported immediately.

The system also provides a side by side comparison of the contentious files within each case and can be accessed using the 'See Comparison Results' button right aligned with the meta data of each case reported. This is the display of the common lines of code between the files thus providing evidence to the percentages reported on the results screen earlier. This gives the instructors a look into the file submissions by two students and gives them an understanding of the level of similarity within the codes. The system highlights the common sections – functions, variables, logic control structures from both the files.



```
return result
def threeSum2(self, nums):
"""
:type nums: List[int]
:rtype: List[List[int]]
"""
d = collections.Counter(nums)
nums_2 = [x[0] for x in d.items() if x[1] > 1]
nums_new = sorted([x[0] for x in d.items()])
rtn = [[0, 0, 0]] if d[0] >= 3 else []
for i, j in enumerate(nums_new):
if j <= 0:
numss2 = nums_new[i + 1:]
for x, y in enumerate(numss2):
if 0 - j - y in [j, y] and 0 - j - y in nums_2:
if sorted([j, y, 0 - j - y]) not in rtn:
rtn.append(sorted([j, y, 0 - j - y]))
if 0 - j - y not in [j, y] and 0 - j - y in nums_new:
if sorted([j, y, 0 - j - y]) not in rtn:
rtn.append(sorted([j, y, 0 - j - y]))
return rtn
if __name__ == '__main__':
result = Solution().threeSum([-1, 0, 1, 2, -1, -4])
print result
```

```
#
class Solution:
# @param candidates, a list of integers
# @param target, integer
# @return a list of lists of integers
def combinationSum2(self, candidates, target):
result = []
self.combinationSumRecu(sorted(candidates), result, 0, [], target)
return result
def combinationSumRecu(self, candidates, result, start, intermediate, target):
if target == 0:
result.append(list(intermediate))
prev = 0
while start < len(candidates) and candidates[start] <= target:
if prev != candidates[start]:
intermediate.append(candidates[start])
self.combinationSumRecu(candidates, result, start + 1, intermediate, target - candidates[start])
intermediate.pop()
prev = candidates[start]
start += 1
if __name__ == "__main__":
candidates, target = [10, 1, 2, 7, 6, 1, 5], 8
result = Solution().combinationSum2(candidates, target)
print result
```

This case can also be downloaded as a pdf file by the instructor to his/her system to keep a record of the same for future reference or as an assistance for evidence to be forwarded to higher authorities such as OSCCR.

We worked on this project through a division of time into three sprints. We addressed the requirements of each sprint in a meeting and discussed the issues to be worked on with a sense of priority. Each issue to be worked on the sprint was added to backlog and as and when an individual would complete a story/issue – he/she would pull in the next task to be addressed immediately. As and when a developer felt that an additional issue would need to be created or a big task would have to be divided into a number of smaller tasks – he/she will create the tickets related to the task in the bigger picture and link the related tickets to each other.

SPRINT 1

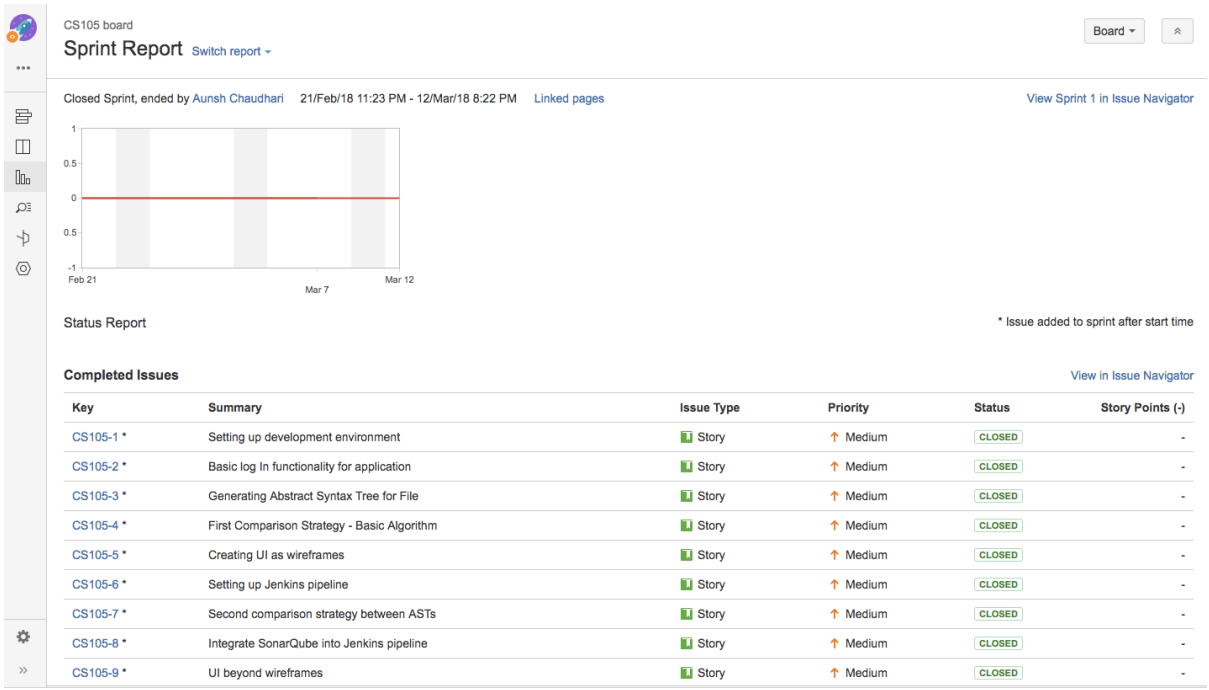
The first sprint was focused on the following requirements:

- Login functionality
- AST generation for a code file
- Implementation of a comparison strategy
- UI as wireframes
- Project environment setup (Maven, Github, JIRA, AWS)

We also set our targets on achieving the stretches as suggested –

- Employing another comparison strategy
- Setting up Jenkins and Sonarqube
- UI beyond wireframes

Our sprint report at the end of the report with the issue completed gives a visualization of the progress we made through the sprint –



CS105 board

Sprint Report Switch report

Board

Closed Sprint, ended by Aunsh Chaudhari 21/Feb/18 11:23 PM - 12/Mar/18 8:22 PM Linked pages

View Sprint 1 in Issue Navigator

Status Report

* Issue added to sprint after start time

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
CS105-1 *	Setting up development environment	Story	Medium	CLOSED	-
CS105-2 *	Basic log In functionality for application	Story	Medium	CLOSED	-
CS105-3 *	Generating Abstract Syntax Tree for File	Story	Medium	CLOSED	-
CS105-4 *	First Comparison Strategy - Basic Algorithm	Story	Medium	CLOSED	-
CS105-5 *	Creating UI as wireframes	Story	Medium	CLOSED	-
CS105-6 *	Setting up Jenkins pipeline	Story	Medium	CLOSED	-
CS105-7 *	Second comparison strategy between ASTs	Story	Medium	CLOSED	-
CS105-8 *	Integrate SonarQube into Jenkins pipeline	Story	Medium	CLOSED	-
CS105-9 *	UI beyond wireframes	Story	Medium	CLOSED	-

We were able to achieve the goals set at the start of the sprint and made sure that we completed each task added to the backlog throughout the period of two weeks.

SPRINT 2

This sprint was focused on the following requirements, stretches and we added them to our backlog:

- Employing a sophisticated strategy.
- Using the Strategy design pattern to accommodate more than one strategy and provide them to the user on demand.
- Performing the comparison against multiple submissions.
- Fully functional UI for base expectations.
- Logging of activities and errors to the system as well as via email.

Notes from the sprint:

- ✓ Additional Strategies: Since we had worked on a sophisticated comparison strategy between the ASTs of the code files in the first sprint, we decided to implement a couple of string comparison strategies like LCS, Edit Distance as well as a strategy to check the author of a file
- ✓ Github Integration: In this sprint, we took up the task of figuring out a way to provide the system an input of file submissions for evaluation. We felt that cloning/pulling the repositories from the Github would be closest to how our client (Professor) would prefer it, thus we went ahead and implemented this approach

Completed Issues View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (12)
CS105-10	Employ a sophisticated comparison strategy	Story	Medium	CLOSED	1
CS105-11	Use other comparison strategies	Story	Medium	CLOSED	2
CS105-12	Report results of each comparison strategy	Task	Medium	CLOSED	3
CS105-14	Ensure system logs activity	Story	Medium	CLOSED	1
CS105-16	Perform the comparison against multiple files used in two submissions	Story	Medium	CLOSED	1
CS105-17	Performs the comparison against multiple submissions	Story	Medium	CLOSED	2
CS105-18	UI for login	Story	Medium	CLOSED	1
CS105-21	Strategy pattern to allow strategy on demand	Task	Medium	CLOSED	1
CS105-24 *	Cloning Git repositories	Task	Medium	CLOSED	-
CS105-25 *	Pulling changes to created Git repository	Task	Medium	CLOSED	-
CS105-26 *	Longest Common Subsequence	Task	Medium	CLOSED	-
CS105-27 *	Comparison Strategy - Using metadata of file	Story	Medium	CLOSED	-
CS105-28 *	Make Sonarqube analysis and code private to team	Task	Medium	CLOSED	-
CS105-30 *	Edit Distance String Comparison	Task	Medium	CLOSED	-
CS105-31 *	System sends an email on exception	Story	Medium	CLOSED	-
CS105-32 *	Backend - Test and Case classes	Task	Medium	CLOSED	-
CS105-33 *	Sequence Detection - improvement of Basic Algorithm	Task	Medium	CLOSED	-

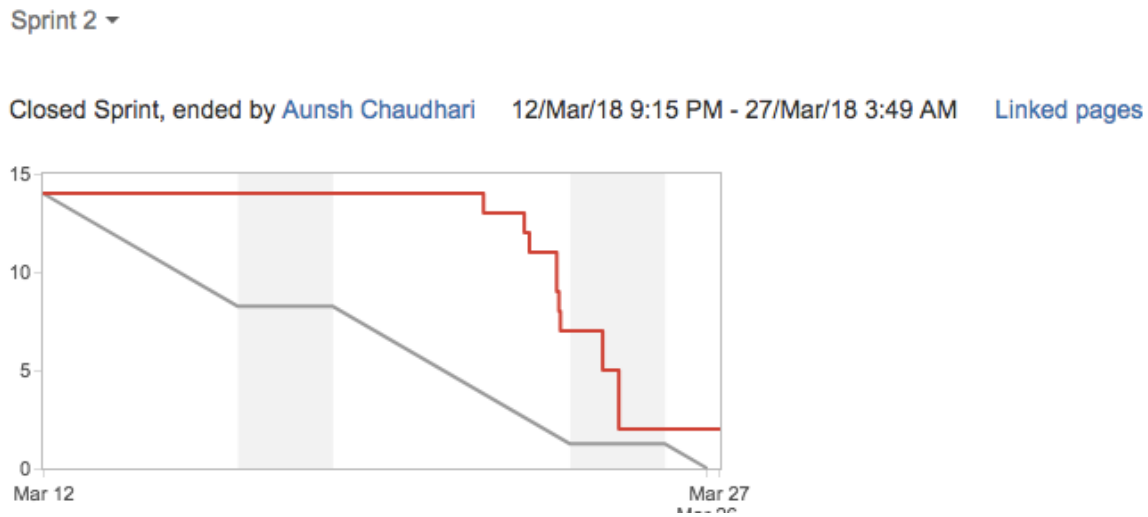
Backlog:

- We worked towards completing the goals set at the start of the sprint. We did overcommit to complete a couple of stretches during the sprint - thus we had following issues as part of our backlog at the end of the sprint. We carried these issues forward to Sprint 3

Issues Not Completed View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (2)
CS105-13	Fully functional UI	Epic	Medium	IN DEVELOPMENT	-
CS105-19	UI for showing similar lines between two files	Task	Medium	TESTS REVIEW	1
CS105-20	UI for test results - comparison strategies	Story	Medium	IN DEVELOPMENT	1
CS105-22 *	Compute overall score using weighted polynomial function	Story	Medium	IN DEVELOPMENT	-

Sprint Report:



SPRINT 3

This sprint was focused on the following requirements, stretches and we added them to our backlog:

- Creating six (6) sets of test projects, each one featuring a particular example of plagiarism, and at least one example that is not plagiarism
- System should provide usage statistics – the number of cases run and system status
- Reporting results of each comparison strategy

Since we were unable to complete the stretches defined by the requirements and added by us in Sprint 2, we pulled them into this sprint and worked towards them:

- Computing an overall score(percent) from the strategies implemented with the help of a weighted polynomial function
- Training the above function using MOSS as a performance standard
- Highlighting the common lines of code between two file submission within a Plagiarism Case
- Making UI fully functional for all expectations

Completed Issues View in Issue Navigator

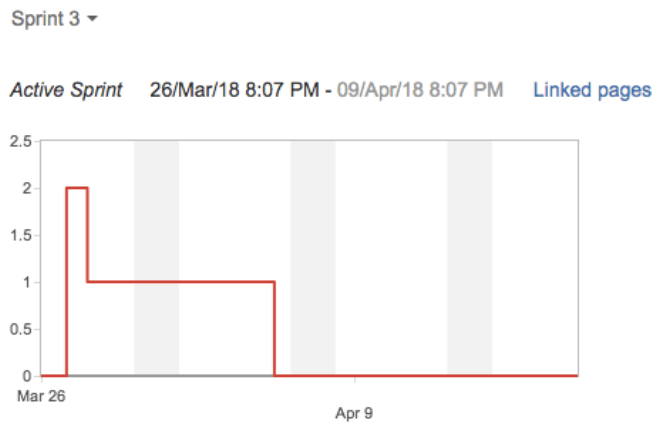
Key	Summary	Issue Type	Priority	Status	Story Points (2)
CS105-15 *	Provide usage statistics	Story	Medium	CLOSED	-
CS105-19 *	UI for showing similar lines between two files	Task	Medium	CLOSED	1
CS105-20 *	UI for test results - comparison strategies	Story	Medium	CLOSED	1
CS105-22 *	Compute overall score using weighted polynomial function	Story	Medium	CLOSED	-
CS105-34 *	Training Observed Results of Strategies according to MOSS data	Story	Medium	CLOSED	-
CS105-35 *	Refactor, integrate strategies and tests	Task	Medium	CLOSED	-
CS105-36 *	Create candidate branch for HW5 with packaged version	Story	Medium	CLOSED	-
CS105-39 *	Test issue- dummy	Bug	Medium	CLOSED	-
CS105-40 *	Test Issue	Story	Medium	CLOSED	-
CS105-41 *	Include line numbers in plagiarism result	Story	Medium	CLOSED	-
CS105-43 *	Testcases giving errors for pythonpasrsers.Python3Parsers testcases, 1 for Login	Bug	Medium	CLOSED	-
CS105-45 *	Javadoc errors in ASTParser, TreeTraverse, Instructor, PlagiarismResult, ASTStrategy, EditDistance, LCS	Task	Medium	CLOSED	-
CS105-46 *	refactoring Antlr code	Story	Medium	CLOSED	-
CS105-47 *	Updating references to show appropriate plagiarism result objects on page redirect	Task	Medium	CLOSED	-
CS105-49 *	Connect to MOSS	Task	Medium	CLOSED	-
CS105-50 *	create six sets of test projects	Story	Medium	CLOSED	-
CS105-51 *	Test set for author strategy	Story	Medium	CLOSED	-
CS105-52 *	Store files on AWS from Git	Task	Medium	CLOSED	-
CS105-53 *	Configure AWS Elasticbeanstalk instances for storing files and directories	Task	Medium	CLOSED	-
CS105-54 *	Integrate Moss with weighted polynomial, new tests and folder structure	Task	Medium	CLOSED	-
CS105-55 *	UI for usage statistics	Task	Medium	CLOSED	-

Additional Notes:

- ✓ We also worked on improving the quality of our code through this sprint. We added a number of unit tests to improve code coverage and refactored code in classes as suggested through code review

- ✓ As part of the requirements for HW5, we prepared our system for the bug hunt and added a number of validations and error checks that improved error handling and the UI for each functionality
- ✓ We tackled a bug reported by a user of our system in HW5 and began working on a couple of others

Sprint Report:



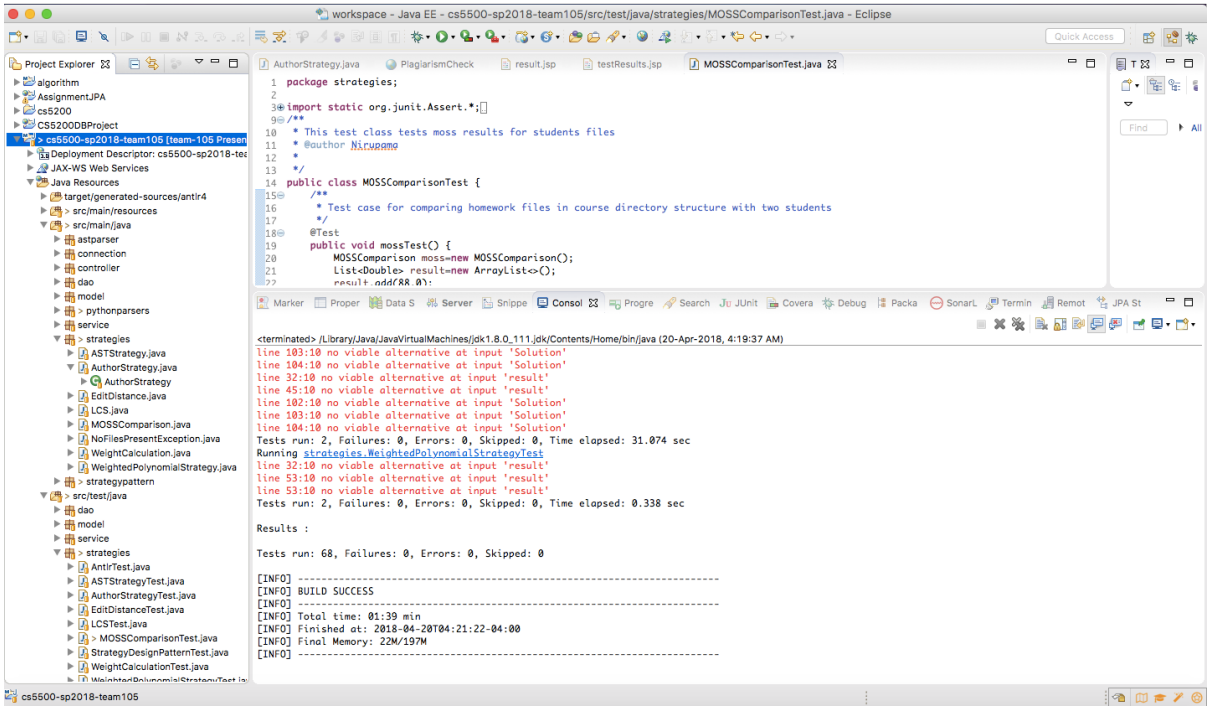
Quality claims supported by testing data

We followed the steps and processes below to ensure code quality –

- Unit and functional testing
- Peer reviews and approvals
- Jenkins build
- SonarQube quality report

Unit Tests

- We have written 68 unit tests for the application and currently have 85.9% code coverage on SonarQube
- We have given it our best shot to test as much of the functionality possible by removing code out of the controllers into services
- Running mvn test locally on the Eclipse IDE, we can see that all the tests pass –



Functional Tests

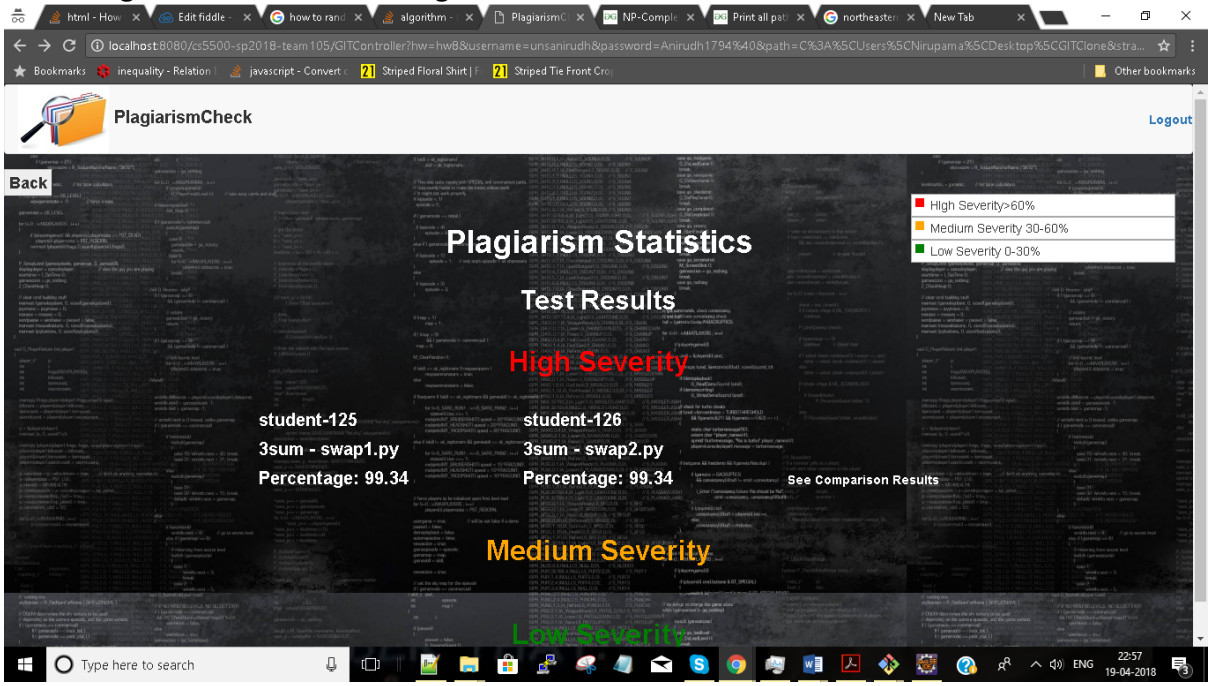
- We have worked towards testing each functionality of the application step by step – initially through unit tests for each function, further through regression and integration testing

- Further, for testing the accuracy of the plagiarism detection, we created six sets of projects for the sprint 3 requirements that validate the working of the detector against a number of scenarios. Below is the report and plagiarism case obtained for each scenario -

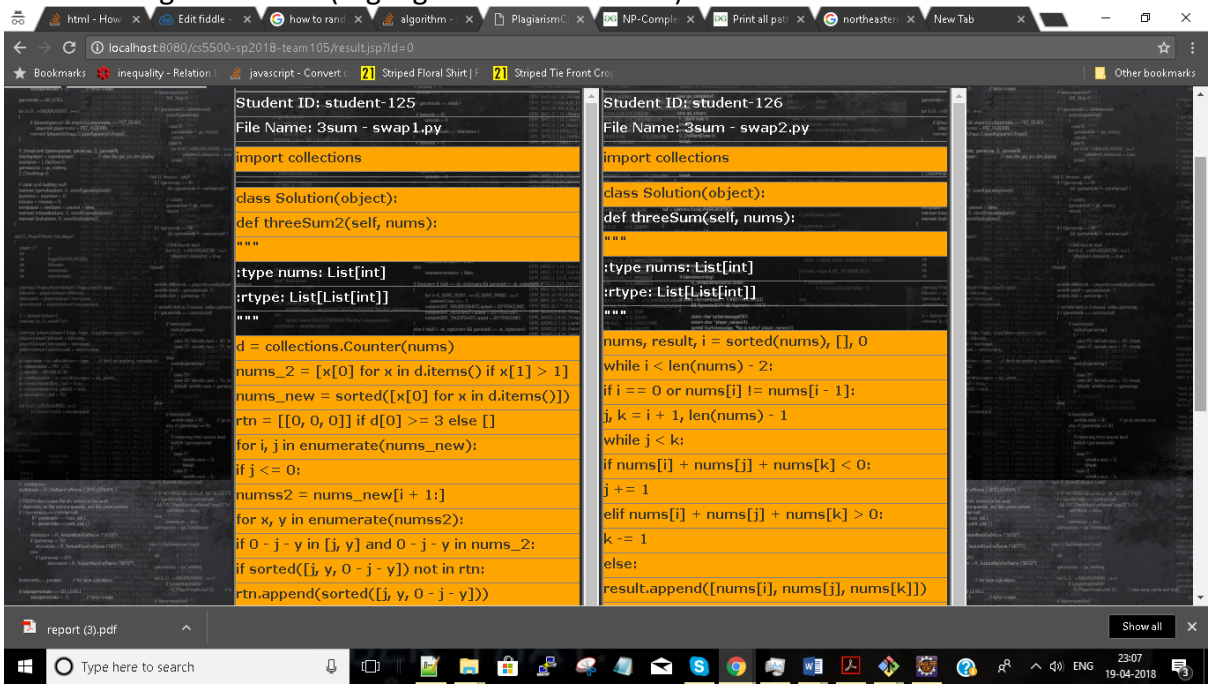
Test set 1:
Function swapping

Student1 Id: Student-125 File submission of Student -125: 3sum-swap1.py
Student2 Id: Student-126 File submission of Student-126: 3sum-swap2.py

Percentages and Metadata of Plagiarism Case



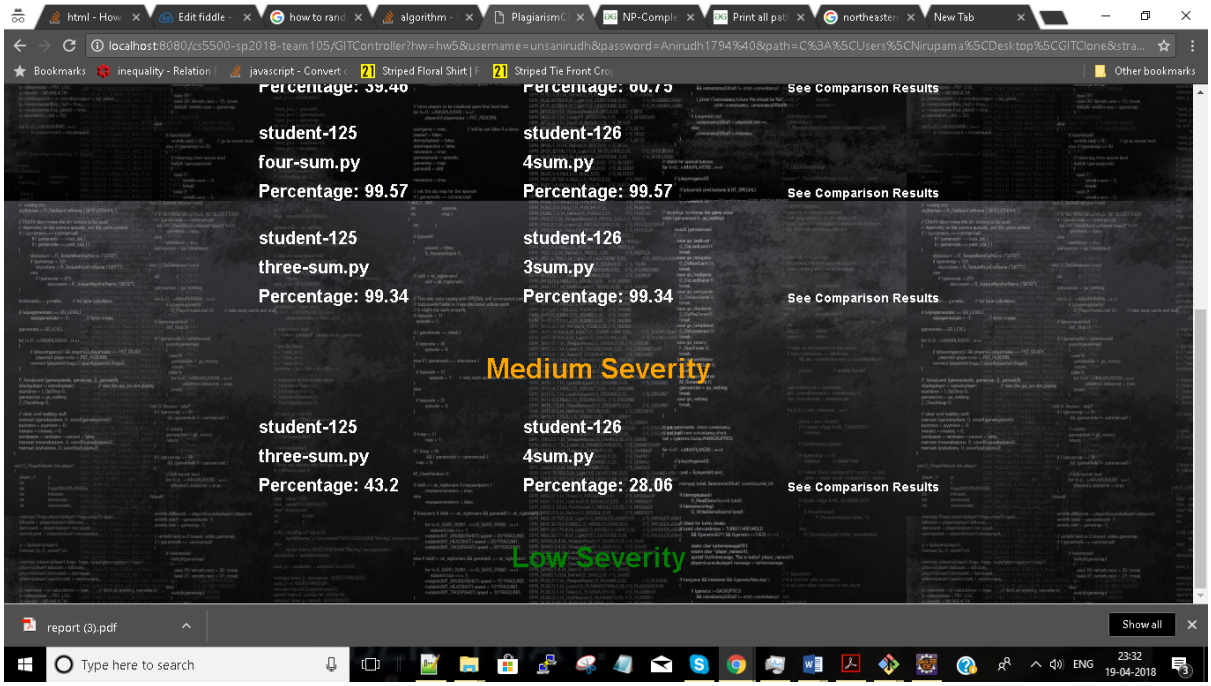
Detailed Plagiarism Case (Highlighted Code Content)



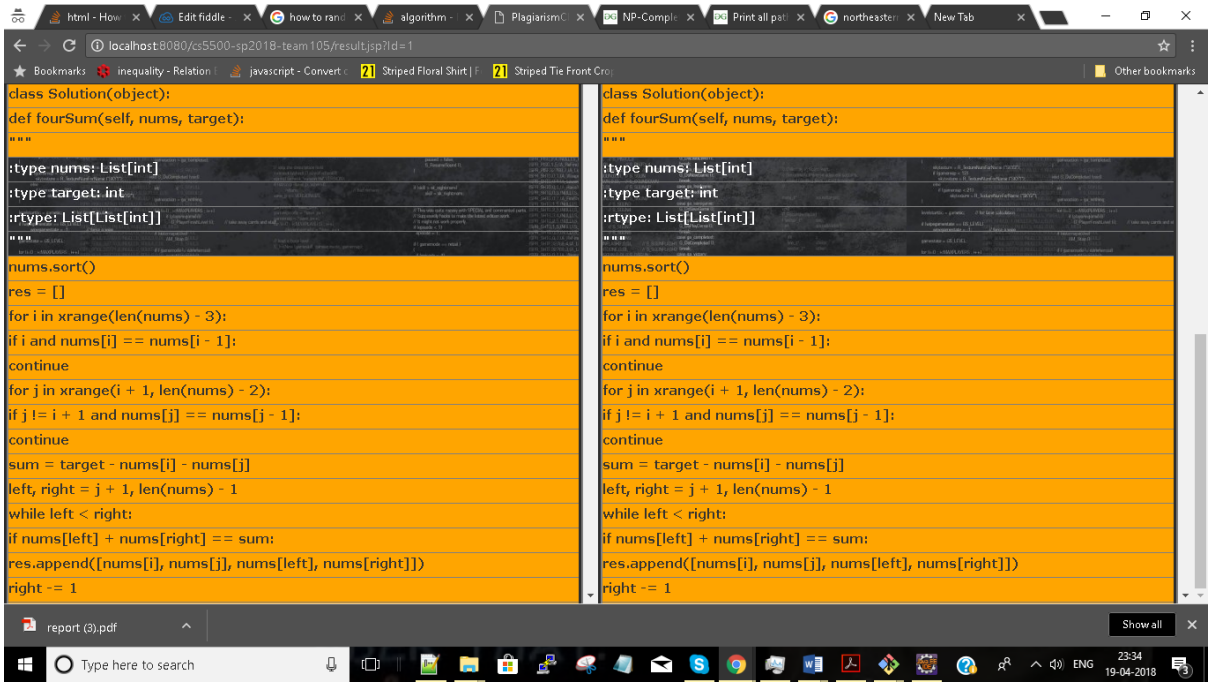
Test set2: Exact match between both files

Student1 Id: Student-125 File submission of Student -125: three-sum.py
Student2 Id: Student-126 File submission of Student-126 3sum.py

Percentages and Metadata of Plagiarism Case:



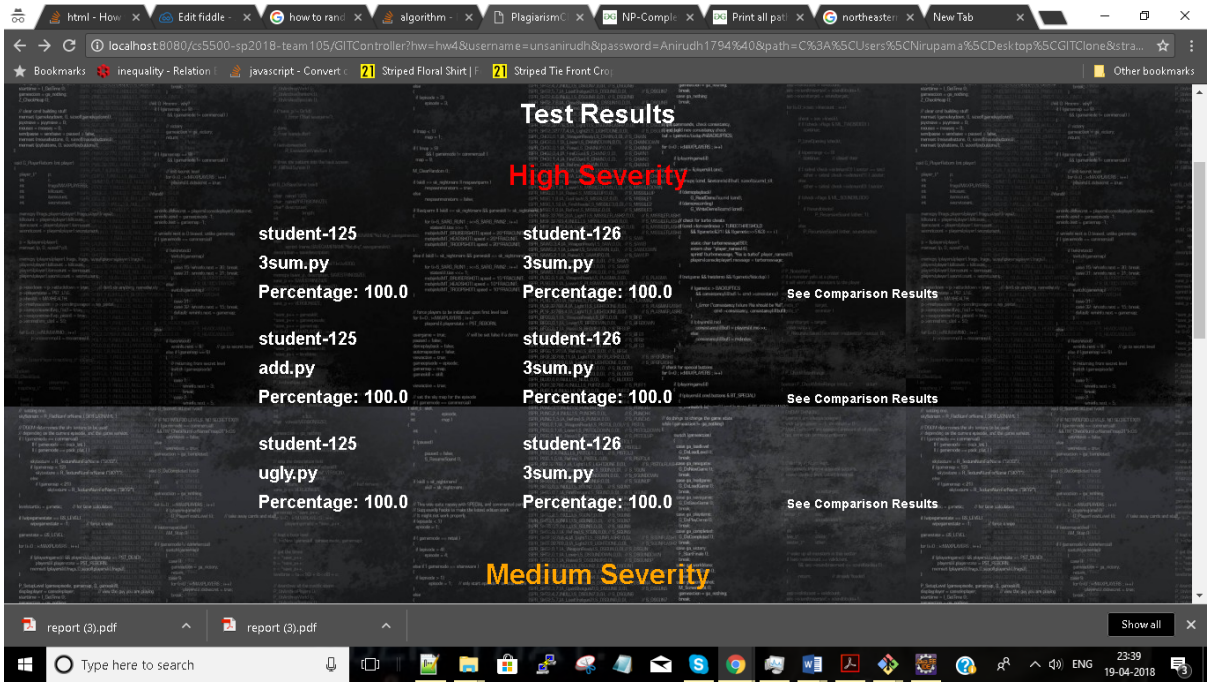
Detailed Plagiarism Case (Highlighted Code Content):



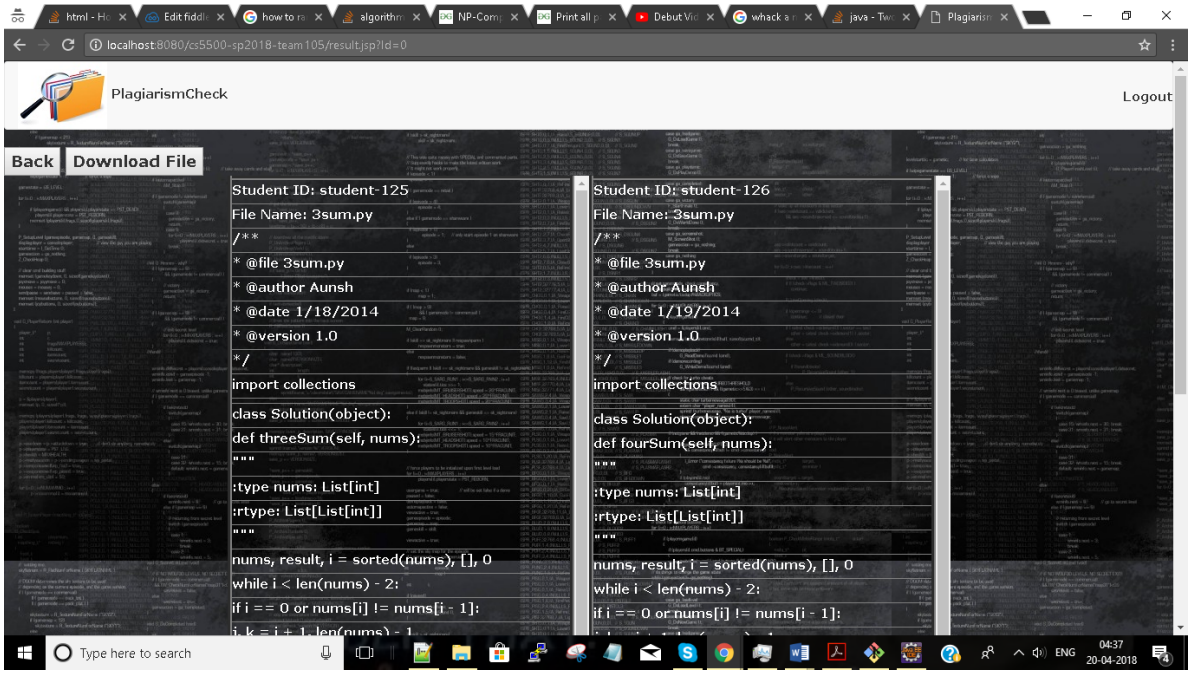
Test Set3: Identical Author of file detected

Student1 Id: Student-125 File submission of Student -125: 3sum.py
Student2 Id: Student-126 File submission of Student-126 3sum.py

Percentages and Metadata of Plagiarism Case:



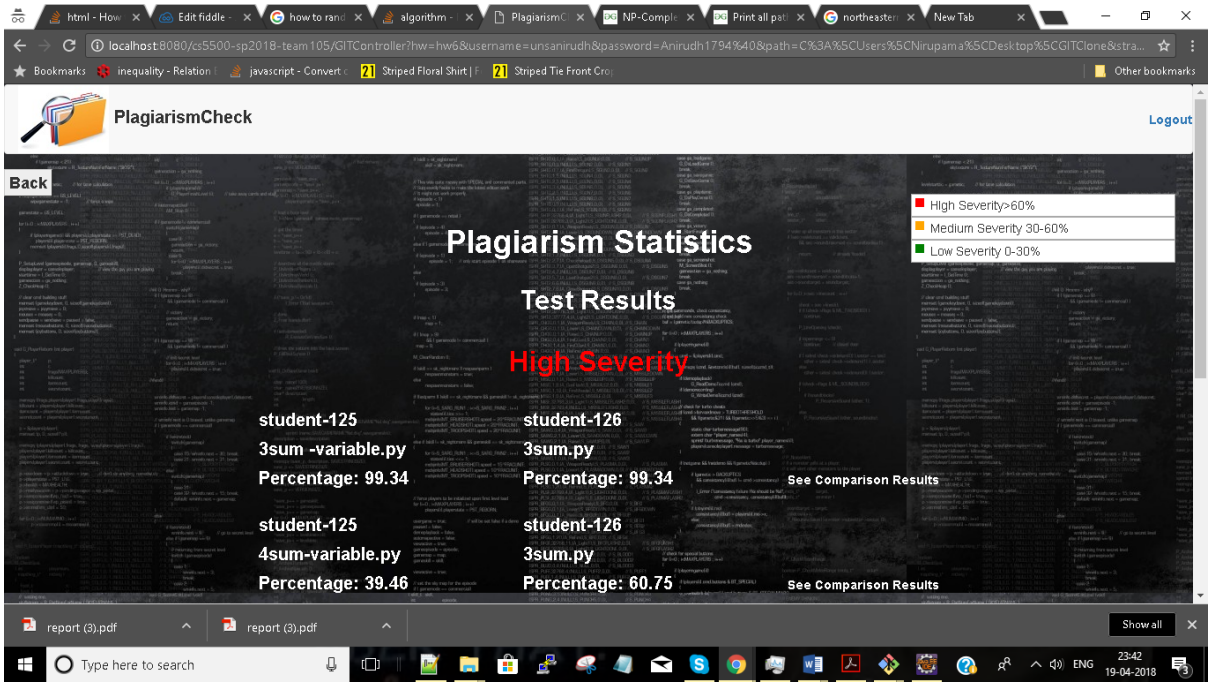
Detailed Plagiarism Case:



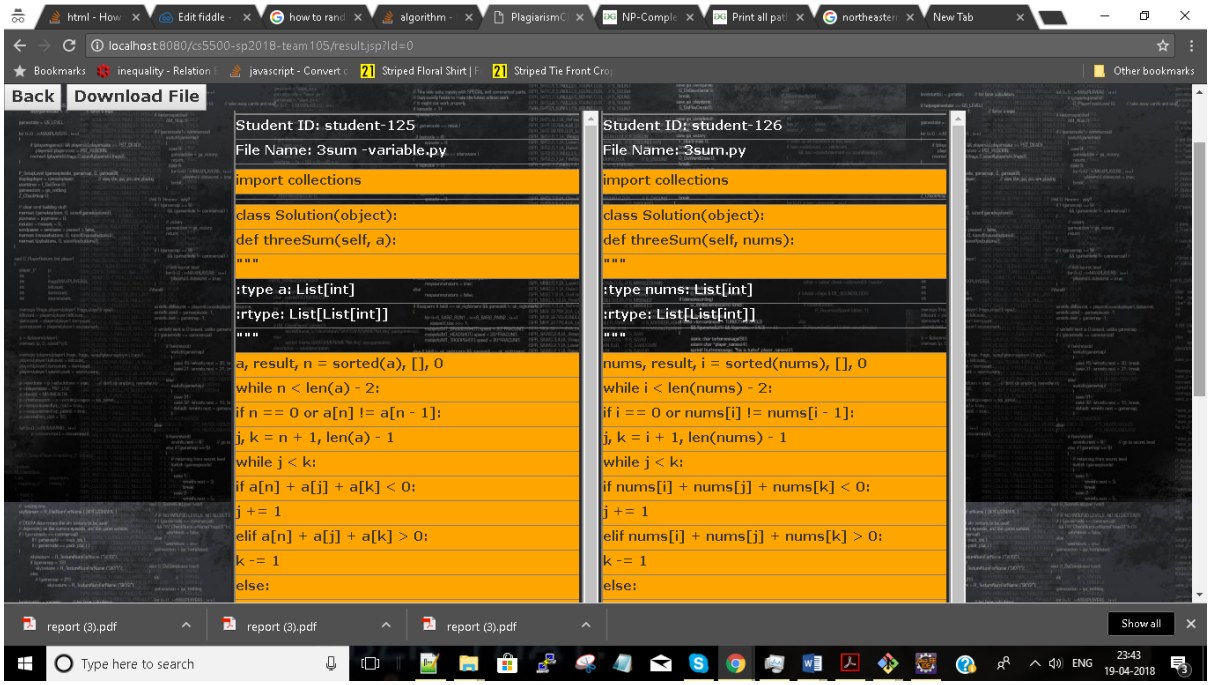
Test Set4: Variable change between two file submissions

Student1 Id: Student-125 File submission of Student -125: 3sum-variable.py
Student2 Id: Student-126 File submission of Student-126 3sum.py

Percentages and Metadata of Plagiarism Case:



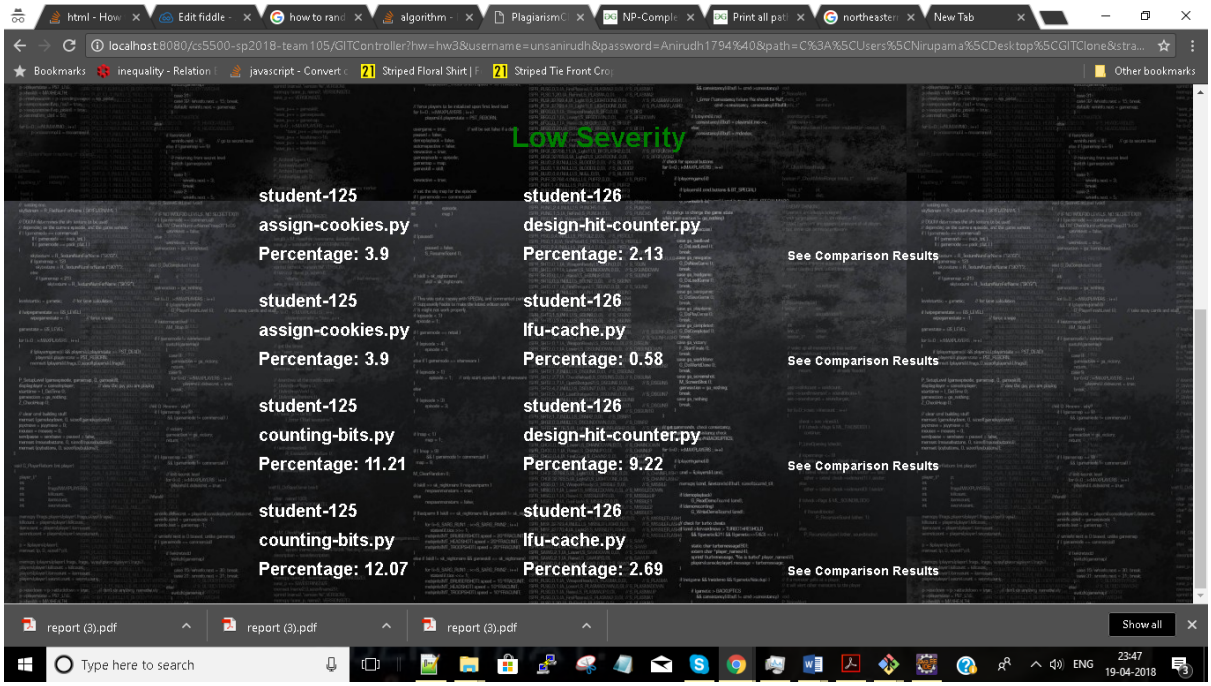
Detailed Plagiarism Case (Highlighted Code Content):



TestSet5: Low severity(plagiarism percentage) test case:

Student1 Id: Student-125 File submission of Student -125: assign-cookies.py
Student2 Id: Student-126 File submission of Student-126 lfu-cache.py

Percentages and Metadata of Plagiarism Case:

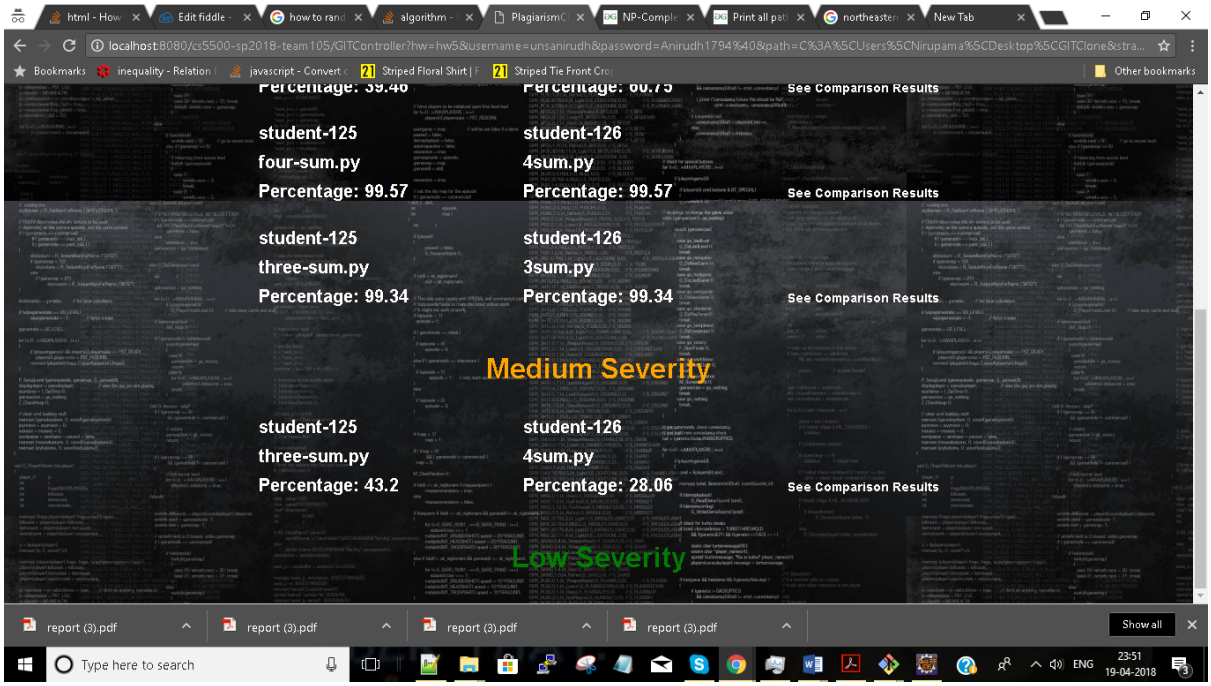


Detailed Plagiarism Case (Highlighted Code Content):

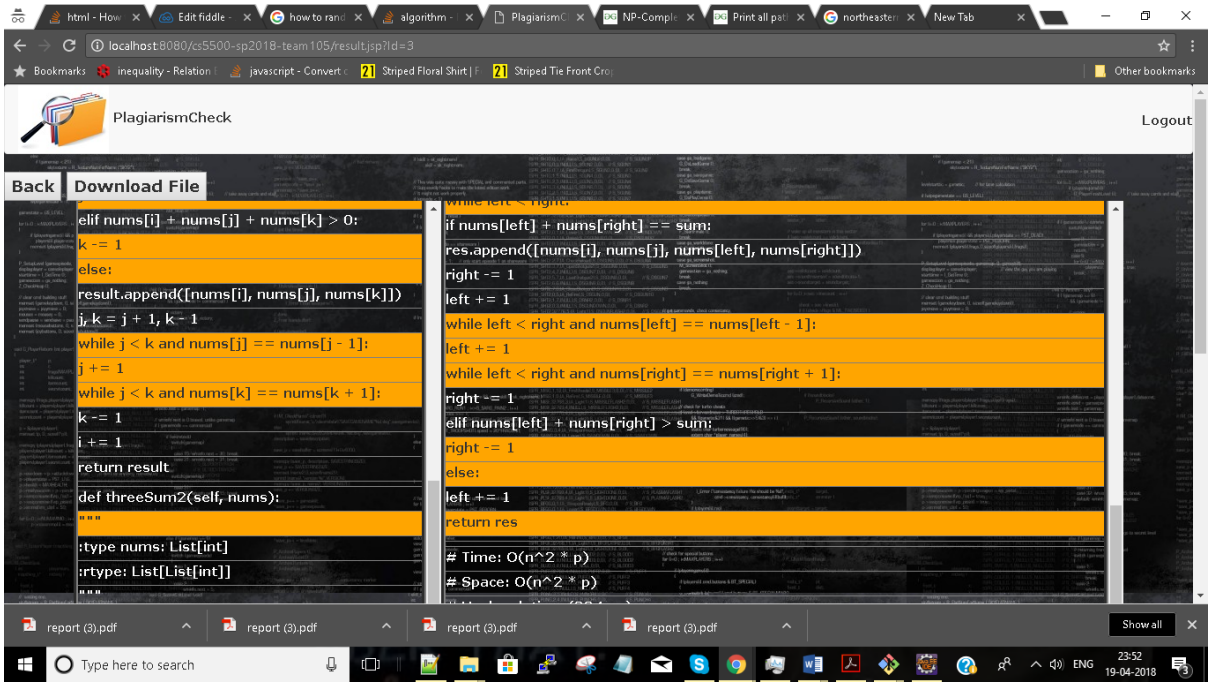
TestSet6: Medium severity case(Partially plagiarized)

Student1 Id: Student-125 File submission of Student -125: three-sum.py
Student2 Id: Student-126 File submission of Student-126 4sum.py

Percentages and Metadata of Plagiarism Case:

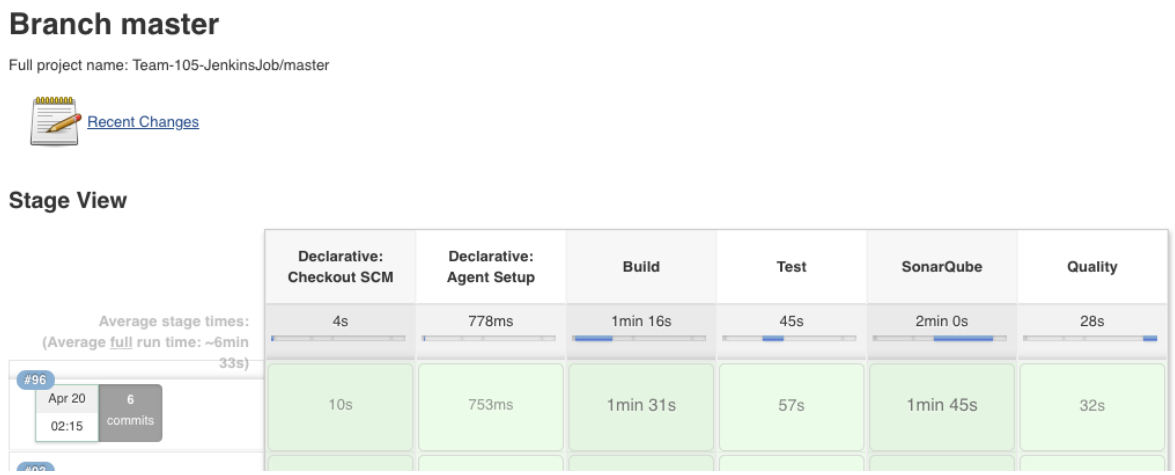


Detailed Plagiarism Case (Highlighted Code Content):



Jenkins

- We can only merge a pull request once all three checks on Jenkins have passed – branch, head and merge
- The run of the most recent build of the application on the master branch is seen below:



SonarQube:

- We can only merge a pull request if the stages of SonarQube and Quality pass on the build
- The SonarQube report of the most recent build of the application on the master branch is seen below: