Management System Development 18 Spring

PROJECT REPORT

TEAM 204

GROUP MEMBER:

KRRISH MITTAL

MINGCHAO WU

VARUN NANDU

HARSHA RAHUL BOGGARAM KRISHNAPRASAD

INSTRUCTOR:

PRO. JOSE ANNUNZIATO

Overview of the problem

Nowadays, the great development of the internet really helps professors and students to catch up the cutting edge technologies. At the same time, it is much easier for students accessing and using some resources that other people made without any thinking by themselves. Unfortunately, plagiarism has become a severe problem at many schools recently.

The topic of our project is making a plagiarism detection web app. After the agile meeting, we decided the specific problem where several students produce different versions of the same code: one version might be derived from another using copy-then-modify, or some students developed collaboratively without permission by the professor.

The main logic that we used in our application is written in Java. Our UI is web-based, by using HTML/ CSS/ JavaScript. For the integration environment, we chose AWS not only because it is widely used but also because it is the most stable platform.

Our system 'Plagiarism Detector' is designed to primarily reduce the load and automate the process of checking for Plagiarism which includes such functionality:
Receive and store assignments. Automatically run detection engine with the user base. And notify the necessary people when required with least user interaction. Display all the results in user-friendly Graph format for visualization.

The user can create their account and upload assignments for the courses they are registered. Our system accepts folder, zipped file of the entire project or even GitHub link. The user can submit their assignments multiple times but the server will not overwrite the previous submissions and they will be always tracked.

The Professor can see the notifications of the latest detections, he also can check every submission and run an on-demand run for plagiarism between different semester and courses. The results are displayed in the form of graph, the professor can click to get more detailed information.

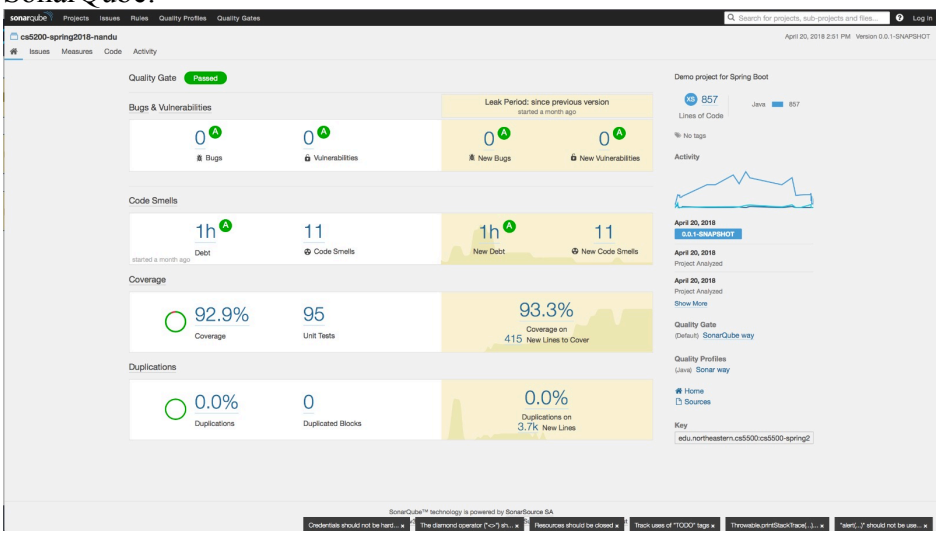
We used some existing libraries for basic infrastructure for parsing code and so on. After comparing with own plagiarism algorithm, we chose to use JPLAG finally. We always follow the development process which has been showed in the lectures.

Here are some required infrastructures we followed:
Our development environment includes both Eclipse and IntelliJ IDEA.
We chose JUnit for automated testing tools.
For version control and issue tracking, we used NEU GitHub.
The reason we prefer Jenkins is that it is always regarded as most efficient confluence continuous integration way.
Besides, we always message and communicate in Slack which is assigned by the professor.

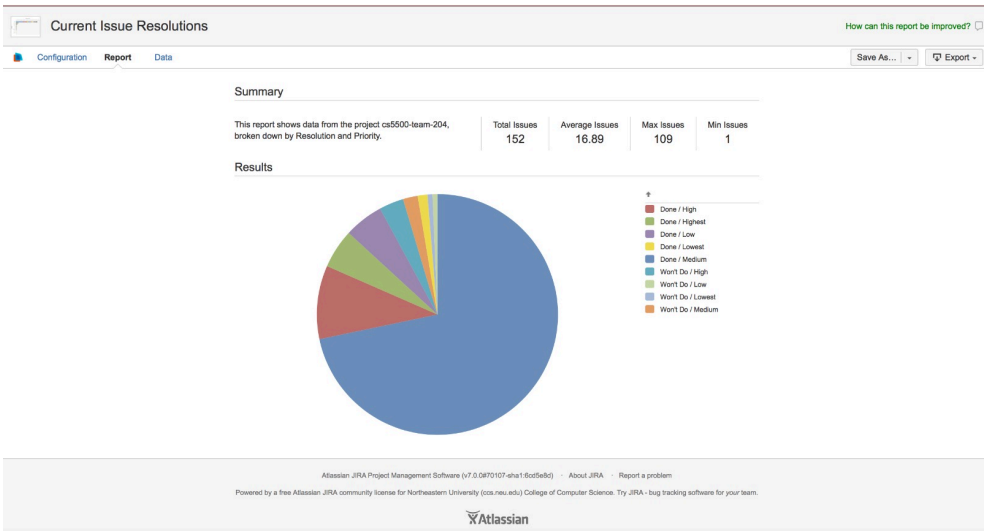
Overview of the result

We devoted lots of time and energy to our project and also accept some suggestions in backlog from students in other teams. They help us find some minus faults that are hard to detect by the developer. We try our best to bring the bugs, vulnerabilities to zero(A), and tests coverage to 93 finally.

SonarQube:



Here is the pie chart of backlogs based on priority and importance to functionality:
You can see we already finished all the functionality we plan and most of the stretches for the project which is indicated in the Jira graph.



Overview of the team's development process

SCHEDULE & DELIVERABLES		
▪ Phase A requirement		January 19-February 2
	Finish: ▪ Use cases ▪ Mock-up of UI	
▪ Phase B (design)		February2-February11
	Finish: ▪ UML diagrams ▪ Java interfaces	
▪ Phase C (implementation)		February 11- April 14
	Finish: ▪ Implementation ▪ Testing ▪ Documentation and user manual	
~Sprint1		February 11- March 2
	1. Made class and DAO for different object. 2. Made a demo for plagiarism diagram in html. 3. Link to the SQL database. 4. Log-in function work in web app. 5. Constructs an AST for a text file. 6. Employs relatively complex compare algorithm strategy. 7. UI as wireframes. 8. We pushed only branches to origin and used pull requests for all merge 9. Master on origin is protected so no direct merge to master is enforced. 10. The project uses maven. 11. Work is being managed in Jira. 12. Our web-app is deployed to a cloud environment on Amazon AWS.	
	1. Need more test for web service and algorithm. 2. Functions need to merge to master.	
~Sprint2		March 2 – March 23

	1. Used two compare strategy, one is complex algorithms and another is library. 2. Junit test case needed for self-defined algorithm. 3. Upload function and UI ready, also support upload zip file. 4. Made UI for different students, admin and professor. 5. Made a special Plagiarism graph, also highlight plagiarism code. 6. Performs comparisons for any new submission from two students. 7. Finish incremental comparisons.	
	1. Smart commit keeps having error. 2. Result page need to be improved. 3. Own Algorithm is separate. 4. Cannot use uploaded file to compare.	
~Sprint3		March 23 – April 6
	1.Faculty can login, select a semester, and CRUD courses and assignments. 2. Submissions are compared incrementally and also avoid long waits 3. Generalize being able to select from any course/assignments. 4. Students can submit an assignment multiple times. System maintains stats for every submission. 5. Any faculty be able to login and view multiple submissions from multiple students. 6. Student can login, select a semester, course, and assignment, and submit 7. System sends email when plagiarism has been detected. 8. Files can be viewed side by side with similar lines highlighted. 9. Implement complex plagiarism Graph	
	1. Not all branch merge to master, not on server during review. 2.Some UI need to be refined, have layout problem 3.Session management is not working. 4.Noresponse after successfully add assignment	

	5.Need notification for incorrect file format. 6.Upload function not working properly.	
Code Walkthrough		April 19
	1. Let system send plagiarism notice email includes link to report that shows students, files, and comparisons of interest. Link only available if user is logged in 2. Email can be cc to relevant students, and other faculty. Link only works if user is logged in. 3. Add queue to store selected submissions of different semesters to compare, faculty can choose which and when to compare. 4. System accepts both file/folder/zip and GitHub submission. 5. App support 4 languages: Python, Java, C,C# .	
▪ **Final Presentations**		April 20
	Finish: Videos, presentation slides, final report, use cases, UML diagrams	

The Retrospective of the project

Retrospectives are very useful and always are used frequently to give our teams the opportunity to pause and reflect on how things have been going and then, based on those reflections, identify the improvements we want to make. Conducting Retrospectives frequently and regularly supports us to continuously improve our future performance. Therefore, we retrospect the whole process in development after the final presentation. We find it really helps us a lot and we summary 4 points as follows.

1. What went well?

After 2 months of hard work, the good news is every function we imagined when we designed the system has been implemented properly. We really enjoy such sense of accomplishment when we use our own project which is good as other mainstream web apps.

2. What didn't go so well?

In the beginning, we just implemented function based on own understanding without general criteria. Then we encountered some problems and devoted lots of time to figure it out when we merge the whole system. Besides, we don't use some good design pattern, also need to change and add a great bond of things to implement new attributes.

3. What have I learned?

From this project, the most important thing that we learn is how to do real team development in the company. We found some bad habits that we had before (less comment, no rule for the naming of the parameter). And we learned using some technologies, for instance Jira for collaboration dev.

4. what needs to change?

Management system development is a really good course. I am really willing to recommend to my peer graduate students. Only one advisement it is giving students more material with real samples.

In performing our project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this project gives us much sense of pride. We would like to show our gratitude professor for giving us a good guideline for the project throughout numerous consultations.

We would also like to expand our deepest gratitude to all TAs who have directly and indirectly guided us in writing this project. Team members have made so many valuable comment suggestions which gave each other inspiration to improve our project.