CodeSniffer - The Plagiarism Detector
CS 5500 - Section 2 (Prof. Jose) - Team 202
Managing Software Development

Presented by:

Sidharth Thapar
thapar.si@husky.neu.edu

Fibin Francis Assissi
francisassissi.f@husky.neu.edu

Anubhuti Vyas
vyas.an@husky.neu.edu

Hao Wu
wu.hao2@husky.neu.edu

CONTENT

Problem Overview:

Detecting plagiarisms from homeworks' submissions has always been a hard task for professors and TAs in universities. Students may use different techniques to cover their behaviour, thus, a simple tool such as "diff" does not work well in such case. For example, a cheating student may split code into different files, renaming function names and adding comments to copied code.

Therefore, our team has come to decide to develop a better tool for faculties. The aim is to build an interactive web application that could detect plagiarism from two computer programs' source code and visualize the result. This tool can be used by 3 user types

- Admin
    - Generic user type. Able to manage users and semesters.
- Faculty
    - As an end-user, faculty can add courses, assignments, and compare student's submissions.
- Student
    - Students has lowest permissions to the system, they can only upload their homeworks.

The tool provides the following functionalities:

1. The UI for the application should be designed for 3 different user roles, depending on who logs in correct UI should be displayed. Faculty use a faculty UI, Students use a student UI, Admin use an admin UI.

2. Admin have the authority to CRUD the user details, semesters and the courses.

3. Faculties and students are able to select semesters. Students can enroll in existing courses. Faculties can create courses or enroll in the existing courses.

4. Faculties are able to create assignments for a particular course in a particular semester, select language for the assignment and set a threshold.

5. Professor can set the number of previous semesters to be compared with, for each of the assignment while creating it.

6. Students can submit their assignments for a registered course in the current semester. Submissions could be a zipped file containing or a GitHub URL.

7. An automatic comparison with existing submissions for an assignment should be triggered when a student submits.

8. The application is able to perform the comparison against multiple submissions for a single assignment.

9. The application is able to perform comparisons for any new submission from same two students, e.g., incremental comparisons.

10. System will send an email (containing link to report) to the concerned instructor when plagiarism is detected.

11. Assignments with plagiarised submissions are highlighted in the professor dashboard.

12. Faculties are able to update the threshold for an assignment.

13. Professor is able to view detailed comparison reports generated by the automated plagiarism detection system. The report display the submissions of two students with similar code sections highlighted.

14. Professor is able to to share a comparison report with students caught in plagiarism.

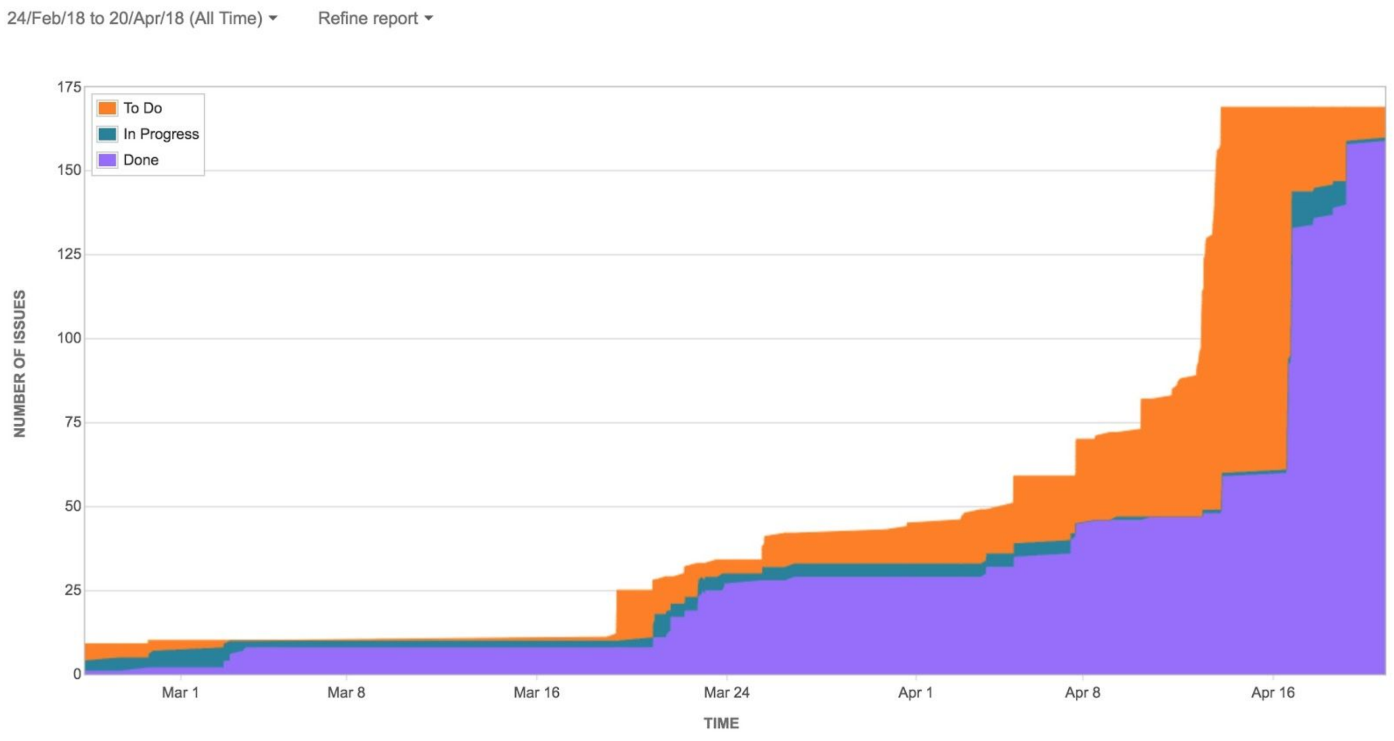
Result Overview:

COMPLETION CLAIMS:

1. By the end of each sprint the team was able to meet all the base expectations and necessary stretches.

2. During the sprint review, based on the suggestions by TAs, if the team figured out a task or feature which should contribute to the project goal, it was added to the backlog and was completed in the next sprint.

3. The team updated the backlog throughout the project duration, as the team gained knowledge about which features should be considered necessary and which should not.

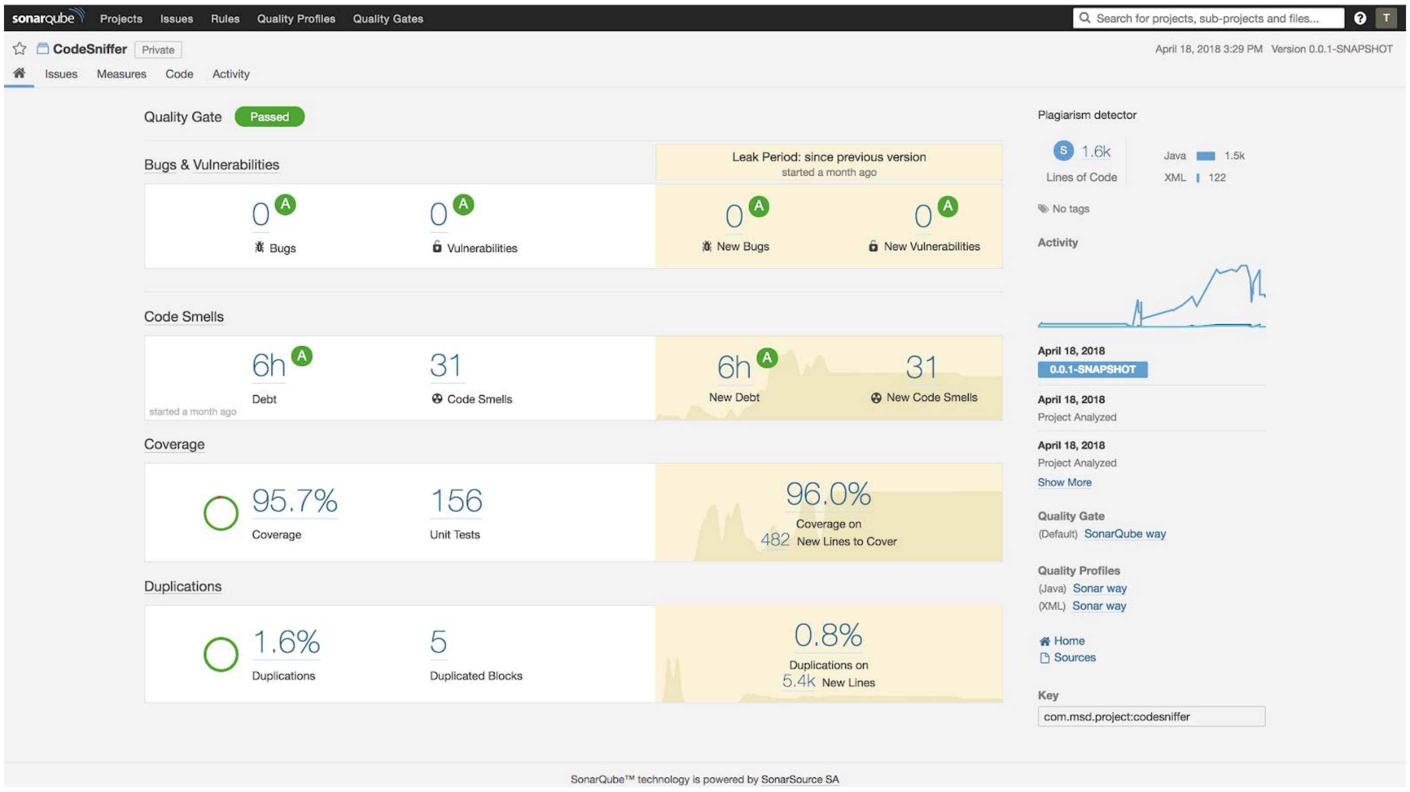


QUALITY CLAIMS:

1. The code is well tested using Mockito.

2. SonarQube quality gate ensured all quality metrics are achieved before the code is pushed to the git, at all times.

3. There was a significant improvement in the quality of the code for the final version.



CodeSniffer Private

April 18, 2018 3:29 PM Version 0.0.1-SNAPSHOT

Quality Gate Passed

Bugs & Vulnerabilities

Leak Period: since previous version
started a month ago

0 A Bugs
0 A Vulnerabilities
0 A New Bugs
0 A New Vulnerabilities

Code Smells

6h A Debt
31 Code Smells
6h A New Debt
31 New Code Smells
started a month ago

Coverage

95.7% Coverage
156 Unit Tests
96.0% Coverage on 482 New Lines to Cover

Duplications

1.6% Duplications
5 Duplicated Blocks
0.8% Duplications on 5.4k New Lines

Plagiarism detector

S 1.6k Lines of Code
Java 1.5k
XML 122

No tags

Activity

April 18, 2018
0.0.1-SNAPSHOT

April 18, 2018
Project Analyzed

April 18, 2018
Project Analyzed

Show More

Quality Gate
(Default) SonarQube way

Quality Profiles
(Java) Sonar way
(XML) Sonar way

Home
Sources

Key
com.msd.project:codesniffer

SonarQube™ technology is powered by SonarSource SA

Development Process:

What worked

Communication:

Overall, we had good experiences communicating as a team. We have used various channels (Slack, texting, phone calls) to discuss problems and solutions over the semester. Everyone in our team is responsive and willing to help on each other.

Responsibility:

We are a group of responsible individuals. Requirements were broken down into small tasks and divided amongst us. Our focus was to ensure we complete on our allotted tasks, and also help each other if struggling. We can recall numerous instances where we used the power of peer-programming to tackle challenging tasks. Everybody actively took part in brainstorming over decision making and building implementation strategies throughout.

What didn't work

Tasks Distribution:

There were few fumbles here and there with unequal distributions due to variable workload for each of the team members in a particular week. Being a group of responsible individuals we managed to tackle those pending tasks.

Efficiency:

We were pretty efficient throughout. Members with upcoming exams/deadlines were given a little flexibility as the other members took charge in that duration. We managed the workload balance by rotating the Scrum Master responsibility, bi-weekly. Even after adopting all these steps, few instances of mis-coordination occured, but we recovered well from such situations.

Retrospect

What did the team like best.

We like learning new development technologies. None of us have used Spring framework before, and some of us have never been a front-end developer. Overall, everyone in the team get to do something new during the project.

From design to deploy, we got to setup our own Jenkins server, doing code quality tests, setup AWS servers, etc. In most cases, the industry requires us to work as more specific roles like Front-end Developers, Back-end Developer, Database Admins, and DevOps Engineers. This project offered us an opportunity to observe and act as many different roles, and take different types of responsibilities.

We get to participate a full SDLC. For some of us, this project is the first time to work in a Agile style management team. Even though it is different than the real working environment, for example, we could not physically meet everyday; this project still prepared us to work in a fast-paced team environment.

What did the team like worst.

The project requirements was not very clearly defined in the second sprint. We chose to use a 3rd-party library for plagiarism detection, and we got the requirements very close to the sprint review date. So we did not have enough time to implement our project to meet all the sprint’s expectations. This did not happen in the later sprints.

What did you learn

We have learned to plan and design a project from scratch, taking risks into consideration. We got a chance to explore a lot in Spring Boot, which helped us understand dependency injection and other Spring concepts. We learnt to test controllers and services using mockito and mockmvc

The whole devOps experience, from setup of Jenkins and Sonarqube to solving any bugs in the pipeline, turned out to be a great learning curve for use. Integration of backend with the MySql database using JPA was an amazing learning experience.

Working in a team definitely helped us learn different approaches for solving a problem.

Throughout the course, we learnt to develop an end-to-end application from scratch in an AGILE manner. The best thing about the course is TEAMMATES, the review system.

What needs to change in the course to support a great experience

Only suggestion I would give is involvement of TA, as a lead during the sprint planning phase also. This ensures equal distribution of work and everyone will be more accountable, which can avoid a lot of chaos within the team. This helps grading also easier because TA can now grade a person individually based on his work rather than grading the entire team.

Future Scope

If we had one more sprint there are few things we would like to implement.

1. The system could extend to cross-language comparison.

2. It would be useful if the system could compare submissions implemented in different languages, since there are courses that flexible with the use of language in their homeworks.

3. Better UI design.
    a. Our website's navigation might not be very straight forward to the users.

4. Forgot your password?
    a. Reset user's password if he/she forgot.

5. Improve the robustness in terms of security
    a. Using hash values on users' passwords