An overview of the problem

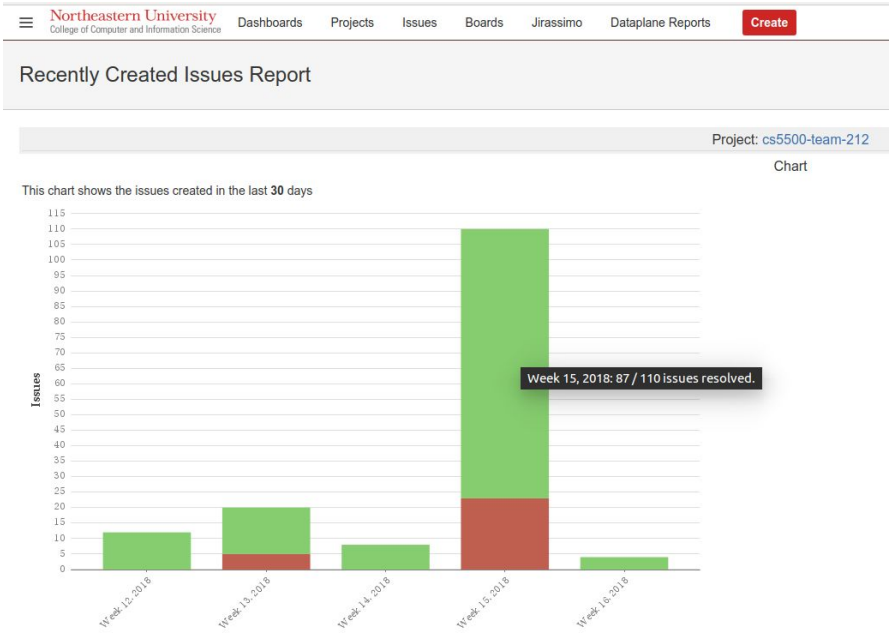
Often times, although unpleasant, we have students not putting in the work into an assignment, giving into pressure and writing someone else's work off as their own. Plagiarism is a common (and often misunderstood) problem that is often the result of a lack of knowledge and skills. We wanted to write a web application to support the education community with a set of resources to make sure students submit their assignments with integrity.

We wanted to make one single platform where a student can submit their work for an assignment created by some faculty and the faculty can check the submitted works of the students for instances of plagiarism if any.

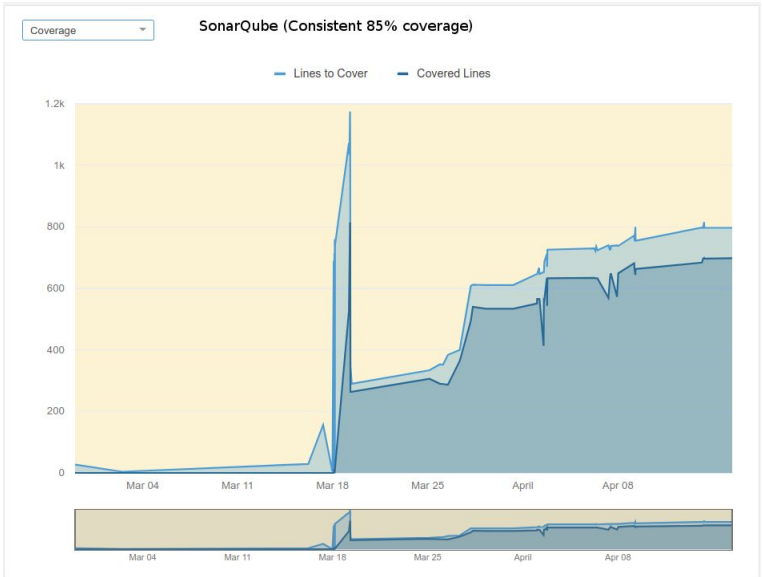
Additionally, we wanted the application to be such that the faculty need not manually check for plagiarism for every single submission made by every student that has taken the course. The application should save the faculty's time by notifying the professor only when cases of plagiarism occur. Also, the application should be efficient enough to save resources by incrementally checking every new submission by the existing submissions for one specific assignment.

Overview Of the Result

- We have resolved about 80 % of the total bugs that were reported by our testing team. The figure below substantiates this claim.



- March 15 onwards, it was expected from the project that the code coverage had to be more than 85%. Since then we have managed to consistently achieve this target and it can be seen in the coverage statistics:



- Additionally, we have used a library that does not merely compare bytes of text, but is aware of programming language syntax and program structure and hence is robust against many kinds of attempts to disguise similarities between plagiarized files. It also helps our web application to detect plagiarism in source code in Python, Java, Scheme, C/C++ and natural language text.

Overview of Team Development Process

During our sprint planning session, as a team, we collectively used the Agile estimation technique of story points to better estimate the work needed to complete a task. This means, the higher the number of story points, the more the number of hours a developer spends on it. This in turn helped us plan to split our tasks more evenly and fairly across all the developers.

The team also consistently reviewed another team members code on github in the form of CodeOwners merging it thereby giving a new perspective to the code which led to an increase in the code quality.Over the last few sprints, our development process had many positives and negatives.The positives being that we consistently had 85% code coverage, checked jenkins tests before merging patches and used smart commits in JIRA so as to track progress in JIRA.

However, we faced a few hiccups where two team members would work on the same module and face a ton of merge conflicts. This in turn increased the latency to finish a certain task. We rectified this later by planning our sprint better where we wouldn't step into each others toe's.

We automated the build process by automatically deploying to Amazon EC2/S3/RDS as soon as a merge was made. We run further Shell scripts via Jenkins to automate the extra changes to run AngularJS and Java in the deployment instance.

When it comes to testing, We use mocking techniques to mock all our access to the system or network to create unit test cases that tested a piece of code. We further had functional tests that ran real code without mocking in our system to maximise the extent to which we tested our system.

We promoted process by adhering to all the required guidelines and added some of our own processes. We enforced a Squash and Merge technique in Github where a Pull Request will correspond to strictly one commit in the repository and each commit will be a smart commit to modify the status of the JIRA Task ID. This helped with readability of the commits and better history tracking in git.

When it comes to keeping track of team progress, we created Epics and Sprints and added tasks to them to closely tracked our Burndown chart to know how much at any time of the sprint we were accomplishing tasks as a team and whether we needed to accelerate at any point to accomplish our sprint goals. We always made it a point to overestimate story points in sprints intentionally so as to help push ourselves to our respective limits.

We had a few shortcomings too. At first, we would autonomously pick bugs and work by ourselves but soon we realised that the lack of communication and the gap in knowledge and/or skills hindered our progress. We later switched to programming in pairs that helped increase overall progress and productivity.Furthermore, we hit a roadblock to merging pull requests because unit tests would fail because they interacted with an external system. We later fixed this by mocking all the tests and by using more test frameworks like mockmvc andpowermock.In the end, we realised that for our team, it was better off to work in teams than to work alone as we each complimented each others skills.

Retrospective of the project

The whole team is excited to deliver a functional web-based application following scrum development process with thorough quality controls. We are able to deliver all the designed functionalities as well as most of the stretch tasks with high quality.
Our UI design has followed the priority pyramid of product / service design: reliability -> speed -> order and structure -> interaction -> aesthetics. Many thanks to professor Annunciato's Web Development class, we were able to quickly built a functional and agile front end with Angular4 framework. This gives us more time in back-end development, integration and testing as well as fast implementation of new customer requirements.
Our back end is based on Java, Maven and interacts with MySql via JDBC. We've been following the test-driven model for all development. We created detailed testing specs before we wrote class methods to make sure we have a good understanding of the requirements. So the development can be deployed with fewer mistakes and easily maintained in the future.
Third thing we are glad with is we have a fast system setup/deploy process. It only takes three simple steps to setup local development environment and the deployment is totally automated since Sprint 1. This helped us to increase development efficiency, improve product quality and customer satisfaction.

The first thing we feel we would have done better is change the JDBC API to JPA. This will save us much time in adding new features and testing. We chose JDBC because three of us are also taking database course and we were learning JDBC at that time. The functionality of the project requires all kinds of complex CRUD. With JDBC we have to write similar queries time to time in different DAOs. By using JPA we can transfer states between our RDBMS and JAVA in a complete transparent way without SQL. With the time saving from this we could have implemented more stretches and enhance the project quality.
The second thing is we would improve is we would spend more time working together from the beginning of the project. We had some problems when we integrated our frontend and backend before Sprint 2 as everyone was working on own part. Although we have daily scrum reviews, we didn't really realize this could be an issue until we tried to combine our work. Later we had whole-team extreme programing working session twice a week to make sure everyone is on the same page.

The most important thing we've learned through the project is the process of developing a software with agile model and using different kinds of development management tools. We were able to apply course knowledge into the project development. As most team members are new to these concepts and tools, learning and practicing them is very helpful to our future academic projects and professional careers.
The second thing we've learned is how the vague and changing customer requirements can affect the development process and what to do to make the project efficient, dynamic, reliable, resilient and maintainable. We know the real-world projects will have much bigger scale and more complicated situations, this project gave us a very good chance to practice formal management of software development.

Taking CS5500 this semester is a fantastic experience. Here we would like to take the chance to thank everyone who helped us through the project the and the semester. We would like to thank Prof. Jose Annunziato for the insightful teaching and the quick response time when asked for help. We would also like to thank our TAs for all the support. Thank you very much for this opportunity.