Northeastern University
College of Computer and Information Science

Plagiarism Detector

CS5500 Managing Software Development Report

Team-210

Shohit Bajaj, Hardik Shah, Willa Davis, Ruchitha Sundar

Problem Overview

Plagiarism in programming assignments can be difficult to detect. Projects are often long in length and dense in content. For the professors and teaching assistants grading the work, it can be very hard to detect plagiarism when relying on memory and eyesight alone. This is an important and relevant issue that runs rampant on college campuses.

In a classroom scenario, homework and project assignments will be similar by nature, due to students being assigned the same problems. This complicates the issue, as all students are learning from the same set of lectures, textbooks, and teaching style, and it follows that assignments are bound to have a degree of similarity. Additionally, students communicate with one another, and often seek help in their classmates.

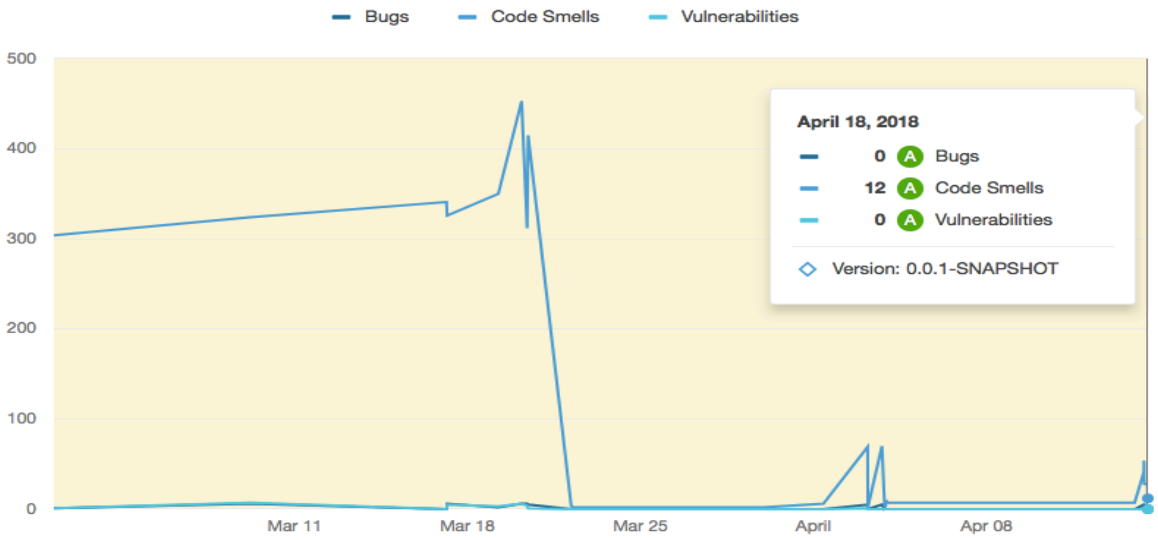
However, while a degree of similarity is expected, there definitely comes a point when work is clearly plagiarized. It can be difficult to decide exactly what that level is, which is why it can be useful for it to be decided on a case by case basis.

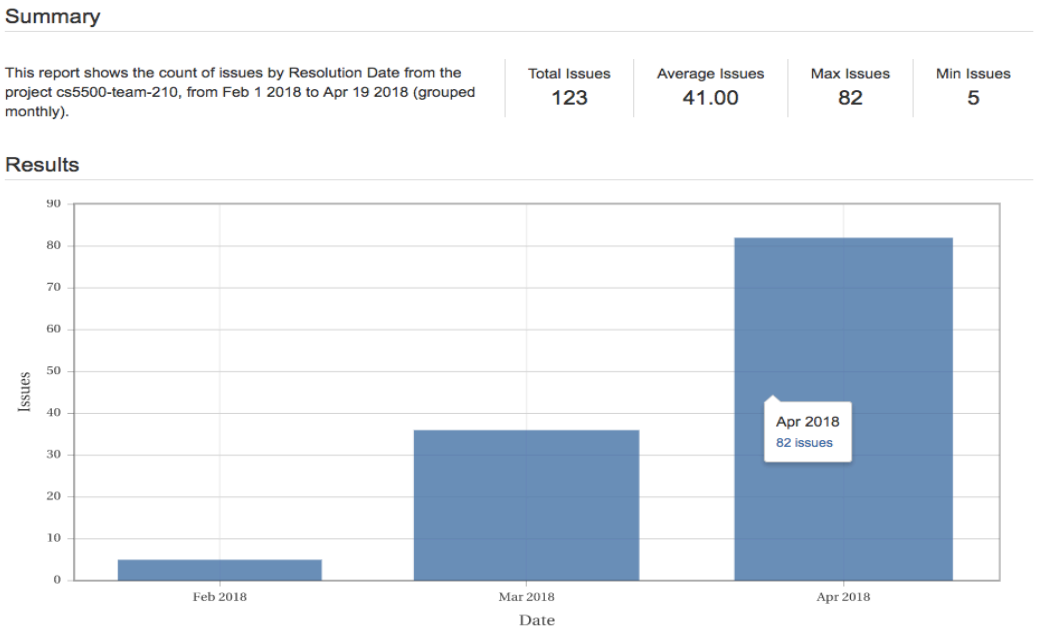
An additional consideration in detecting plagiarism is that there are many ways to commit it. Plagiarism can be two assignments being exactly the same, but more often than not, the situation will be more sophisticated. For example, a student might change method, class, object, or variable names. Alternatively, a student might reorganize the content, or split the content into a number of files. And, of course, a student might do all of these things in an effort to conceal this fact.

Clearly, there is a great deal of things to think about when trying to detect this problem through coding, and our team attempted to solve this problem by building a Web App that would assist the instructors to catch cases of plagiarism.

Result Overview

The Quality of the product was maintained by using Jenkins for Continuous Integration along with SonarQube Quality check running on all builds which was required to pass before any code can be merged to the Master branch. The team was successful in completing all stretch goals along with the base requirements in each sprint. Jira board was used to track issues in sprints. The first figure provides evidence on software quality while the second figure provides statistics about the issues resolved every month during the development process.





Development Process

Agile methodology was followed for software development where the development cycle was divided into 3 sprints. At the end of each sprint the functionality achieved along with code quality was reviewed with the Teaching assistants. The sprint reviews played the role of small milestones, which helped us keep track of our progress. The whole process of building and deploying our code on AWS EC2 instance was automated by using Jenkins, which also had a SonarQube quality check configured in it. The quality check would only allow code that passed certain quality criteria (85% test coverage) to get merged with the Master branch on Github.

The first step we took in our project was to research which algorithms would help us attain the most accurate percentage of similarity. Our thought process was that mostly completing the backend first allowed for a very sturdy foundation for our UI to interact with. We chose to implement three algorithms for a more wholesome approach; Levenshtein Distance, Longest Common Subsequence, and Edit distance for Abstract Syntax Tree comparison. In this stage, we also tested these algorithms. Another key piece of the backed was the classes that took the files and made them more digestible by removing white spaces and comments and condensing multiple files into one. On the User Interface side our goal was to make the application extremely easy for the end user to interact with which we believe we were able to achieve.

What worked for us:

- At least one member in the team with previous experience in UI development, Backend Development.
- Clear design goals at the start of the project, which helped in minimizing the design changes later.

What didn't:

- Workload of UI development not divided equally since not many people in the team were comfortable in developing the User Interface.
- No experience of using any testing tools for testing our API's.

Project Retrospective:

The Course title “Managing software Development” is apt in the sense that all of us learned how to manage the development process. We learnt why is it important to break down a big project into smaller tasks, following strict deadlines, constantly reviewing your code and writing tests, writing good code is more of a responsibility than a requirement since other members in the team might have to use and maintain our code. The whole project development process in the course was a fulfilling experience since we could start a project from scratch, overcame the hurdles, worked as a team and delivered a quality product to the client.

In the hindsight we feel that to improve the course experience proper and timely feed back could be given to teams so that they can improve the software and meet the expectations that the teaching team has. Sometimes we did not receive timely feedback after our sprint reviews, which put a lot of pressure on us to deliver some features at the last moment.

Finally, the course taught us that when you are working as a team it is important to support each other, have clear goals, have a good understanding of the requirements before starting to code, Communicate regularly with everyone in the team, manage tasks and reprioritize them and focus on developing a quality product rather than something that would meet the deadline but does not meet the client’s expectations.