Northeastern University
College of Computer and Information Science

CS5500: Managing Software Development

# Team-107 Project Report

Darshan Panse, Samanjate Sood
Shail Shah, Vaibhav Dave
https://github.ccs.neu.edu/cs5500/team-107

April 17, 2018

## Contents

## 1 Introduction

Northeastern University defines plagiarism as using words, ideas, data, code, or other original academic material of another without providing proper citation or attribution. Plagiarism can apply to any assignment, either final or drafted copies, and it can occur either accidentally or deliberately.

Plagiarism is a major concern in an academic environment. It affects the reputation of both the individuals involved and the institution. It is unethical to take part in such an activity. This report describes an overview of the problem, the result, the team's development process, and a brief retrospective.

## 2 The Problem

Currently, the instructors are dependent on the Teaching Assistants(TAs) to bring cases of plagiarism to light. The TAs are asked to keep an eye out for such instances while grading the homeworks. Then, the instructor manually analyzes the homeworks and has to meticulously file a report to University's Office of Student Conduct and Conflict Resolution (OSCCR). This is not always an optimal approach.

Firstly, the each TA is assigned a fraction of the total number of students enrolled in the course. It is possible that plagiarism goes undetected if students who committed this act are from different groups. It's not feasible for every TA to go through every homework assignment.

Secondly, while humans are good at analyzing patterns, it is highly possible that some instances of plagiarism are overlooked due to human error. Sometimes it is better to have a second or third eye.

Thirdly, manually checking for plagiarism is an arduous process. Usually, the class strength of CS 5550 is about 60 per section. It would take a really long time to properly check for plagiarism for all pairs of students.

Fourthly, after affirming a case of plagiarism, it takes considerable time for the instructor to file a report that describes the exact location and type of plagiarism.

Clearly, the current setup is flawed. We were hence tasked with designing, implementing, testing, and evaluating an application which helps automate the process of plagiarism detection and report filing. Since we were free to detect plagiarism cases of any programming language, we chose C, which enjoys popularity even after decades of inception.

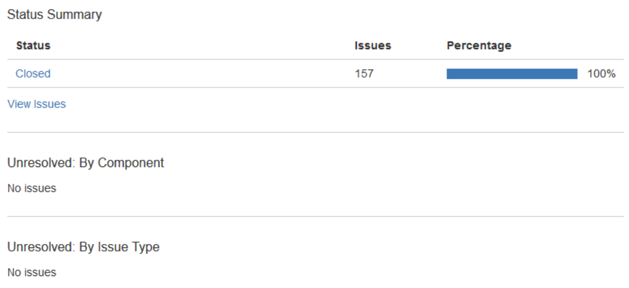
## 3 The Result

We built an application which accepts student repositories containing homework folders and that checks for plagiarism in the specified homework folders. Our application produces similarity scores and generates a summary of all probable plagiarism cases in the submissions. It categorizes student pairs into highly suspicious, moderately suspicious and clean lists. We have used multiple algorithms to detect plagiarism (LCS, Levenshtein distance and weighted average). The user can select any algorithm he/she wants. We even used machine learning to learn the weights by fitting a regression model over Moss dataset.

Using our application, a user can view similarity scores between homeworks of two students, view similarity scores of two files within two homework submissions and view similar snippets from the two C files of the two students under scrutiny. We have two different stand-alone versions of the application, one for the professors and the other for the teaching assistants. While the Teaching Assistants can view the summary & snippets, and generate a textual report which includes similar snippets from multiple code files of the two selected students, the professors can also view the names of the suspected students and email them.

The user can also view the global statistics showing the number of files that have been scanned and the number of plagiarism cases found by the application till date.

We accomplished everything we set out to do which is evident from our Jira board at https://cs5500-jira.ccs.neu.edu/projects/CS107.


Status Summary

| Status | Issues | Percentage |
|---|---|---|
| Closed | 157 | 100% |

Unresolved: By Component

No issues

Unresolved: By Issue Type

No issues

We took special care of the quality of the software that we built. To this end, we used Test Driven Design (TDD) and used SonarQube quality gate as a build step. We protected master branch and set up a workflow using Jenkins for continuous integration (CI). We made sure to have more than 80% test coverage so that our application is free of bugs. We were very particular about addressing issues raised by the testers from other teams and managed to close all issues in time. The end product is a beautiful piece of software which we are proud to present.

## 4 The Development Process

The project was divided into three phases, Phase A, B and C. Here is a brief description for each phase.

Firstly, Phase A involved understanding the problem, gathering requirements and coming up with mock-ups and use-cases. We set up a meeting with our instructor to get to know the problem and what he was looking for. We ran by our suggestions and in the end we had a solid understanding of how to move ahead. We also researched about existing systems and brainstormed ideas to implement in the next phases.

Secondly, Phase B was all about the system design. With the help of the use cases and mock-ups created in Phase A, we developed Sequence and Class Diagrams. Then, we translated them to interfaces which we would implement in the next interface.

Finally, Phase C involved testing, implementing, and documenting the system. We had three sprints, lasting two weeks each, to come up with a system.

Here's a brief overview of the process involved in each sprint:

1. At the beginning of each sprint, we added stories and/or bugs to our backlog and estimated the stories we would be taking in the sprint. We then added them to our sprint board.
2. Next, we built branches corresponding to the stories on the sprint board, and started working on them locally. We moved the stories we were working on to the 'In Progress' swimlane.
3. Once we thought that our work on the story is complete, we created a pull request on Github, asking others to test the changes. Continuous Integration via Jenkins made it possible to be sure that the master branch does not break and that all tests pass on a remote machine.
4. Once the PR was approved, we merged the changes from our branch to master. We then marked the story as closed on Jira.
5. At the end of the sprint, we had a sprint review with a TA to go over our progress in the sprint. It helped us get important feedback to do better next time.

Though we preferred face-to-face communication, we also used Slack, Teammates, and WhatsApp for communicating and co-ordinating.

## 5 Project Retrospective

We learned a lot on the way. There are a lot of things we got right and a few setbacks which are opportunities to improve. The course has made us confident enough to start working in the real world.

Here are some things we liked:

1. We were presented with a challenging problem to solve. It forced us to think about applying Design Patterns, and writing beautiful code.
2. We enjoyed the DevOps role we played in the project. We not only got to code, but we also got to build the pipeline and integrate tools that helped us in the development.
3. Code reviews in class were helpful to both the presenter and the audience. As presenters we got valuable feedback from different perspectives, and as an audience we got to see mutual bugs which we could later identify and fix.

Here are some things which can be improved for the subsequent semesters:

1. It would be great if individual feedback is provided, in a timely manner.
2. It would also be helpful for teams to work on different projects. It would have more impact on their resumes.
3. Starting the project a little earlier in the course, with a few more sprints would help the students build better software.

Overall, we had a great experience with developing the project. It has helped us write better tests and implementations, and has improved our inter- and intra-team communication.

## 6 Conclusion

We have built an application we are proud of. However, we plan to take it forward and add more features to it, and hope that instructors and TAs use our system in the semesters to come.