



Date: 04/20/2018

CS5500 - SEC 02
Managing Software Development
Team - 206

Plagiarism Detector

Developed by:

Mohd. Asim Khan
Sanket Mathur
Jiaxin Yang
Jayant Gangadhar

Guided by:

Prof. Jose Annunziato
Prof. Michael Weintraub

Problem Statement:

Plagiarism is a statement that someone copied code deliberately without attribution. In academia, programming assignments are used to evaluate students in programming courses. Therefore, it is a necessity to ensure a student's submission is novel.

Plagiarism is further more than copying submission from one student to another. It is a legal problem. It can be illegal in certain circumstances to plagiarise copyrighted material without the consent of the copyright owner.

Impacts of plagiarism:

- **Academic Integrity** - Academia is built on the tradition of a concept of intellectual honesty - academic integrity. The trust that our world has in education and research is founded on this concept. However, the occurrence of plagiarism risks the breakdown of this trust and subsequently constitutes a basic threat to academic integrity.
- **Knowledge** - The fact that cheating occurs is a direct threat to the quality of education and research. An academic degree should mirror the knowledge and skills of the person being examined; qualities that cannot be acquired through cheating.
- **Fairness** - The fact that plagiarism occurs contributes to the creation of unfair conditions between students. This comes to a head, for example, in competition for internships or employment following completion of the students' education.

It is common knowledge among the higher education community that a number of students are engaged in some forms of academic dishonesty. Plagiarism in programming assignments is a conventional form of a breach of academic integrity and it constitutes a major menace to the educational procedure. Therefore, detection of plagiarism in programming assignments is an essential task for each instructor to carry out.

Manual detection requires substantial effort and excellent memory and is impractical in cases where too many documents must be compared. It will further involve a good amount of time to detect plagiarism in any submission of students when the number of submissions is sufficiently large and the human resource available to detect plagiarism is less. Additionally, the manual task will be more prone to error and with very fair chance that 100% plagiarism will not be detected as it will largely be based on the memory of the person who is performing this task.

Problem Solution:

Certainly, it is a time-consuming and toilsome task to be performed manually. Software-assisted detection allows vast collections of documents to be compared to each other, in very less amount of time as compared to the manual task and will be less prone to error. It also provides flexibility to the user to set matrix as per their requirement. The manual effort to create a report will be skipped completely as the software can automatically detect the plagiarized content and generate a report out the content compared.

We have built a plagiarism detector application which will detect plagiarism among all the submissions it has and generate a report of the plagiarized material. The current application will:

- input a set of program sources in files or the whole project structure
- analyze all the input files, detect and highlight the students who are involved in plagiarism related to that submission and their submitted content.
- generate a detailed report on the content that marked as plagiarized.

Together with detecting the plagiarism detection, this application supports other features like:

- a student can register and enroll or drop a course
- students will submit their work related to one particular assignment which is managed by the professor and plagiarism will be detected among submissions by the students.
- Admin will manage all the users and will have the god access for the application.

























Tasks completed in Sprints:

Sprint 2 26 issues

09/Mar/18 11:13 PM • 23/Mar/18 11:13 PM































		CS206-18	Front end does not work when I deploy application as a WAR file
		CS206-23	Create flow for File Comparison module
		CS206-20	Build Properties file and integrate with Mongo Fields
		CS206-19	Home page background image gets repeated
		CS206-32	Integrate MOSS plagiarism detector
		CS206-33	Create front end pages for uploading file and showing comparison result
		CS206-34	Create angular services to upload file to server and receive result
		CS206-21	Create Flow for updating Student Data
		CS206-22	Create service for File Comparison and integration with the library
		CS206-24	Writing Email module
		CS206-25	Create user Approval flow
		CS206-28	CSV Course data to Database
		CS206-35	Create Code Similarity Algorithm
		CS206-39	Integrate AST

		CS206-40 Implement Strategy Pattern to differentiate between the strategies
		CS206-41 Write test classes for User
		CS206-42 Write test classes for UserService
		CS206-43 Create admin page
		CS206-44 Create pages for editing and compare python files
		CS206-45 Write test classes for UserDao
		CS206-46 Create Logging for All the Services
		CS206-47 Create Logging service for User Service
		CS206-48 Create Approve User Flow by Admin
		CS206-51 Get weighted score from different strategies used and generate the report of it
		CS206-53 Write test classes for UserController
		CS206-54 Write test cases for Algorithm

▼ **Sprint 3** 30 issues

27/Mar/18 12:37 AM • 10/Apr/18 12:37 AM



		CS206-30 Create Course Domain Object
		CS206-60 Create Student Flow Enroll/Drop in a course
		CS206-62 Create Domain Objects
		CS206-64 Create REST services, business logic and dao for student to enroll in a course
		CS206-63 Create Semester Domain Object
		CS206-64 Create Assignment Domain Object
		CS206-65 Create a Scheduler job which checks for newly added submission and then runs the comparison Test every night
		CS206-66 Create Course CRUD Operations
		CS206-68 POC for S3 Bucket
		CS206-69 Create REST services, business logic and dao for student to drop a course
		CS206-70 Create Semester CRUD Operations
		CS206-74 Create Assignment CRUD Operations
		CS206-72 Create Submission CRUD operations
		CS206-73 Create professor page for semester, course and assignment CRUD

-
- ✓ ↑ [CS206-74](#) Create student page for course enrollment and assignment submission

 - ✓ ↑ [CS206-75](#) Create frontend services for CRUD semester, course and assignment

 - ✓ ↑ [CS206-76](#) Implement code highlighting for detected similar code

 - 📖 ↑ [CS206-77](#) Implement Line Similarity Algorithm

 - 📖 ↑ [CS206-78](#) CRON jobs to compare file automatically

 - 📖 ↑ [CS206-79](#) Build Report Module operation

 - 📖 ↑ [CS206-80](#) Write test classes for Assignment

 - 📖 ↑ [CS206-83](#) Write test classes for Semester

 - 📖 ↑ [CS206-84](#) Create logging for classes

 - 📖 ↑ [CS206-85](#) Create email flow for approval activities

 - ✓ ↑ [CS206-86](#) Create front end service to enroll and drop courses

 - ✓ ↑ [CS206-87](#) Create web pages to show students comparison result





















 - ✓ ↑ [CS206-88](#) Update comparison pages to include student assignment

 - 📖 ↑ [CS206-91](#) Create submission flow to upload the files in the s3 bucket and update the same in the database

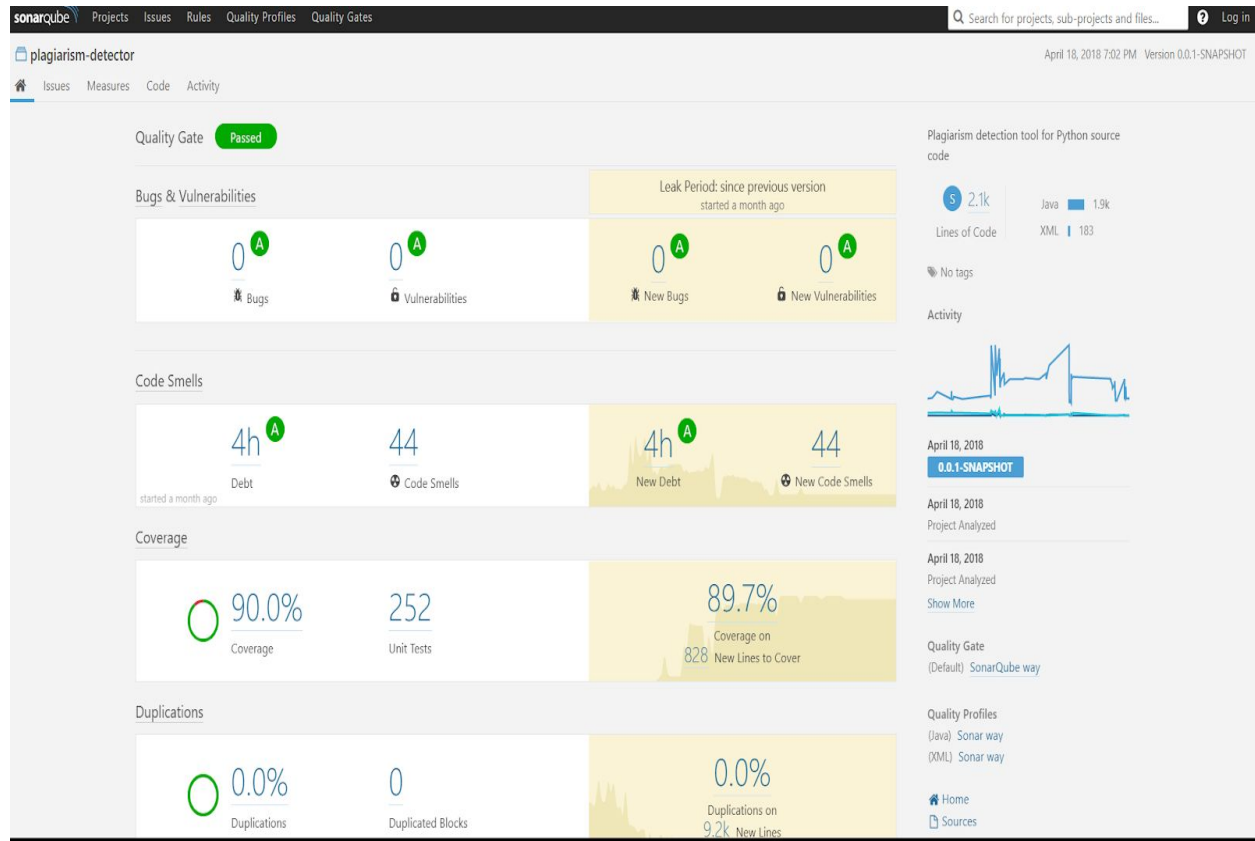
 - ✓ ↑ [CS206-89](#) Create page to show comparison history

 - ✓ ↑ [CS206-92](#) Update Line Similarity class to use custom Pair class
-

Backlog issues which will be incorporated in the future and were not the part of the 3 development sprints:

		CS206-93	Large file comparison failed
		CS206-95	Rest APIs are public
		CS206-130	Trace appears in UI
		CS206-131	When someone logs out, new user when clicks on login, he can see previous user's credentials.
		CS206-143	No tests for MailService class.
		CS206-144	Only one test for FileServiceImpl, all the edge cases should be covered.
		CS206-146	Even if user logs out, still he/she can access all the functionalities.
		CS206-153	If user go back to the main page, user will automatically log out
		CS206-154	user cannot change his password
		CS206-156	Some error happened when refresh the page

The team has tried and successfully maintained the code quality throughout the project. We have written 252 test cases with proper documentation of the code wherever required. Below is the **SonarQube** detailed report of the code quality taken from the Jenkins:



Development Process

The development was divided into 3 Sprints

Sprint 1

- Planning and Design
- Risk Analysis
- General setup of the application
- Basic UI creation
- Research done to develop algorithms to parse python code and generate ASTs which in turn would be used to perform various similarity checks on two python files

Sprint 2

- Backend and Front-end development
- Controller and API development
- Integration with the front-end and database
- All desired functionalities were developed and integrated

Sprint 3

- Front-end and Backend Validations
- Automated CRON jobs integrated with the system to check for submissions in an iterative basis
- Automated Email integrations with various checkpoints integrated
- Unit Test cases were written
- Integration Testing of End to End application

Post Sprint 3

- Defect fixing and refactoring of code
- Functional Testing
- Report and presentation creation

For every sprint we followed the following process

- Discuss the action items to be done for the sprint and define a scope
- Assign tasks to each member of the team
- Discuss the status of assigned tasks every night
- All team member used to meet every 3 days to do integration and allotment of new tasks and discuss if anyone is facing any serious issues.

What Worked:

We developed a hybrid application based on the requirement from both the professors. Not only did we have custom Algorithms to perform similarity check on the files as was required by Prof. Weintraub, but we also have a Student, Admin, and Professor portal so that the application could give a good taste of any student management system (like blackboard) as was suggested by Prof. Annunziato.

What we implemented

- A portal for 3 users namely Professor, Student and an Admin
- Admin has God access over all the other users and approves/rejects new account requests.
- Admin can create a new Semester.
- Professor can create new Courses to be taught under him.
- Professor can add/remove assignments to his courses.
- Professor can set a benchmark upon which plagiarism will be checked.
- Professor can manually upload Files/ Folders from the UI and compare the submissions. for plagiarism.
- Professor can see all the compared files data in a very intuitive tool which highlights all similar lines.
- Professor can see all submissions uploaded by students and compare them
- If plagiarism is found, professor can notify the students for the same and ask for a meeting.
- A CRON scheduler run every night at 11 pm which checks for any new submissions done on that day and notifies the respective professors for any case of plagiarism
- Students can enroll/drop courses .
- Students can upload their submissions to the respective assignments for the courses they have enrolled.
- We implemented our own comparison strategies namely Zhang Sasha Algorithm, Longest common Subsequence, Line similarity which were used to check plagiarism from python files.
- A weighted polynomial function implemented to combine the scores of the above implemented strategies.
- Text highlighter tool used to highlight similar lines in the code

What we could not implement

- Session Management
- Inter-Semester and inter-courses similarity check for Plagiarized files
- Machine learning via Moss
- Single Sign on and Google Login.
- Versioning of Submissions uploaded by Students

Feedback of members from Learning Perspective

Mohd Asim Khan

- Learnt to plan and design a project from scratch while taking the risks into consideration.
- Got a chance to work on AWS (EC2 instance, S3) which turned out to be a great learning curve for me as I was totally new to AWS. I also integrated the S3 bucket with my java code to upload and download files which was really fascinating.
- Got the chance to implement design patterns like Strategy Pattern, MVC and also got a good picture of using Dependency Injection while working on Springs.
- Learnt integration of Java with MongoDB using Springs.
- Got a chance to work on JWT tokens using spring security to secure the web APIs.
- Got a good experience while handling all the DevOps process like Jenkins setup, SonarQube Setup, Jenkins monitoring and Github monitoring.
- Got a chance to work in a team which helped me to view problems and tackling for the solutions with others' perspectives.
- Building an end to end application from the scratch in an Agile Manner.
- Learnt to integrate AOP, GMail API with the Spring

Sanket Mathur

- Gained knowledge of developing a project from the scratch by reading the system requirements
- Gained experience of working in team and follow agile development process and respect timelines and code standards
- Learned to develop a Spring Boot application using MVC architecture and design patterns
- Learned MongoDB and worked on integrating it with Spring Boot
- Learned usage of AWS S3 bucket and integrating it with Spring Boot
- Gained good experience and tricks to use Git and GitHub
- Gained experience to write controllers and APIs to integrate backend with the front end
- Learned to use monitoring system like JIRA

Jiaxing Yang

- Gained knowledge and experience on exercising the Agile development process.
- Gained experience on working together in a team. Learned how to identify and prioritize use cases, and planning for each sprint. Learned a lot from teammates' suggestions and perspective.
- Developed the whole front end of the system from scratch. Learned how to communicate with a Java server via RESTful api. Learned how to integrate Angular in Spring framework.
- Gained experience on Angular development. Learned a JavaScript utility for highlighting code.

Jayanth Gangadhar

- Had the opportunity to work on a new project from scratch and experience the complete software development lifecycle (SDLC)
- Gained useful knowledge on how to work as a team and work with people of different diversities and levels of expertise. Was able to improve/learn new skills thanks to feedbacks given by teammates.
- Learnt about best practices to be followed and follow the agile methodology during the course of a software development.
- Learned about the various design patterns and had the opportunity to implement some of them.
- Learnt about usage of GitHub for version control and the various tasks such as managing and tracking issues via JIRA among other tasks involved in the code review process
- Gained knowledge on developing a spring boot application and work with REST architecture.
- Developed strategies to detect plagiarism and also detect plagiarized lines of codes.
- Understood the fact that just getting the code to work is not enough.

Team Feedback on what can be improved in the course

- The project was really interesting and all the teammates got the chance to learn a lot of new things
- Instead of 3 Sprints we could have 4 Sprints, in this way we can focus on the quality of the project more instead of hurrying things up along with keeping up with added functionalities. Due to having only three sprints, we missed to implement some of the really interesting features.