CodeX Plagiarism Detection System

Developed By : Team-102

Mayuri Kadam
Aditya Kumar
Mrunal Ghanwat
Prathamesh Tajane

Northeastern University
CS 5500 – Managing Software Development
Spring, 2018 – Professors Annunziato and Weintraub

CodeX

Spring-Boot Web Application to Detect Plagiarism in Programming Language

Team-102 - Section A

Team members

Aditya kumar

Mrunal Ghanwat

Mayuri Kadam

Prathamesh Tajane

Contents

1. *Overview* 3
2. *Overview of Result* 4
3. *Backlog Statistics* 5
4. *Quality Claims* 6
5. *Development Process* 7
6. *Retrospective* 9

Overview

The objective of this application is to help professors detect plagiarism among students submitting solutions for Python programming assignments. Examples of plagiarism includes moving functions or methods to another location in the same file, renaming variables, classes, and methods, extracting sequences of statements into methods.

The application should employ different techniques/algorithm to detect different patterns of plagiarism in the programs.

The instructors would require an interactive platform/system which enables students to submit their assignments for the courses they are enrolled in. The platform should allow professors to check plagiarism based on similarities among submission of different students.

The System should be able to apply different comparison strategies to detect similarities and report the results of each strategy. It should be able to generate an overall plagiarism score using weighted polynomial function.

This system should be able to scale the scores generated by comparison strategies using Machine Learning technique in line with MOSS standards.

The system should be able to perform comparison against multiple files and submissions of different students.

The system should be able to generate statistics of the number of plagiarism detection cases run.

Overview of Result

The developed web application provides interactive platform with simple UI. Students can submit assignments and professors can check for similarities among submitted files for multiple students.

The professor can create courses and homeworks corresponding to each course. A Student will register himself in a course created by professor. He will be able to see the list of homeworks created by professors for registered courses. Students can submit their assignments for each homework.

Students can upload files or folders for each submission.

After submission of assignments, professors can check the plagiarism for a student against all submissions in corresponding homework. System allows professor to see the similar chunk of code detected under plagiarism between two programs.

Three different comparison algorithms were implemented to detect various categories of plagiarism in python files.

AST is generated for a python file using antlr library. This AST includes Rule name according to the structure of a python file. The nodes generated in AST are used for comparison for python files.

Hashmap strategy compares python files Based on AST generated for each file. The AST nodes of two files are compared against each other. And Similarity percentage is calculated based on the node count.

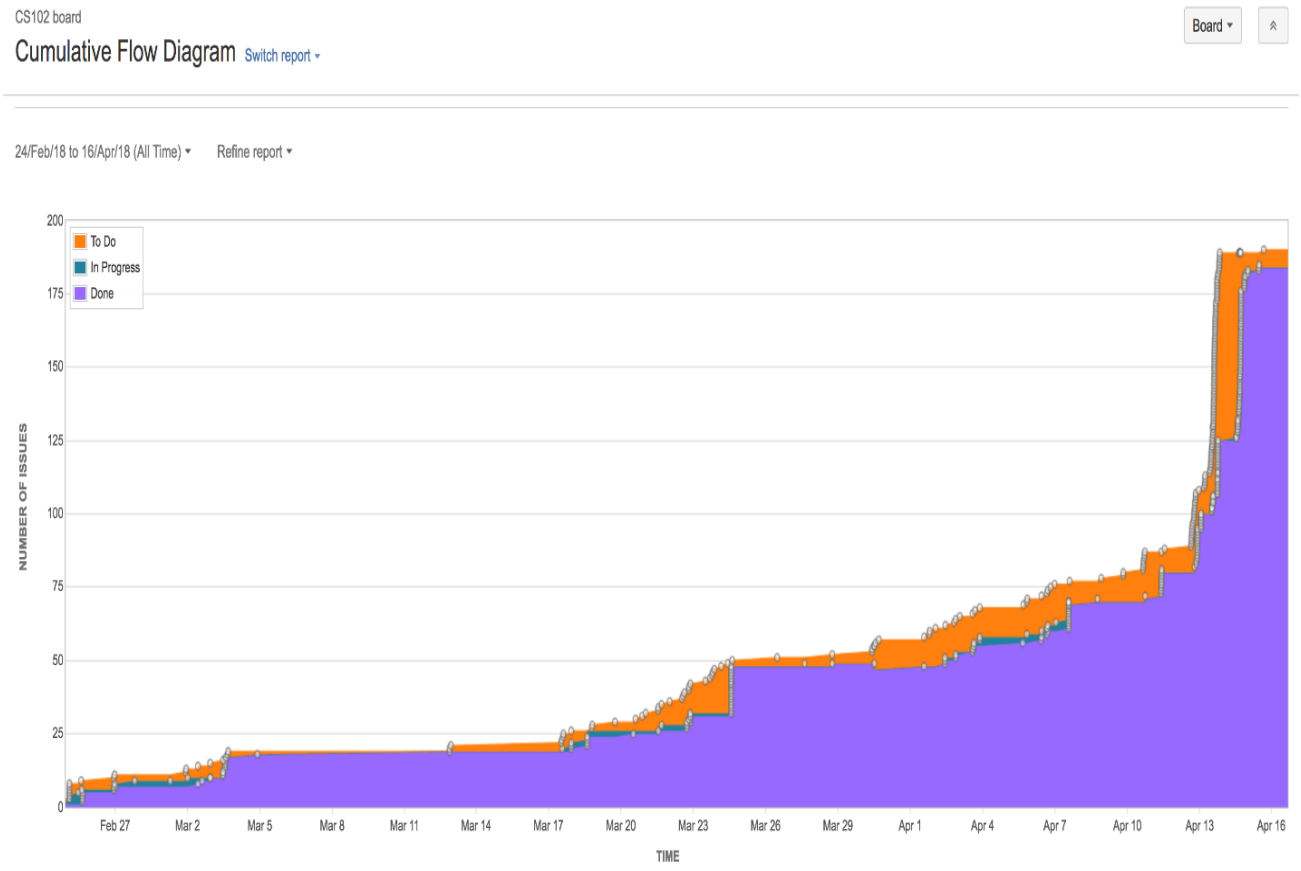
Tree Strategy generates tree nodes for each python file. Each tree node contains current AST node name, current node depth, and parent node name. The list of tree nodes of two different python files is compared against each other and normalized matching percentage is generated.

Levenshtein strategy compares python files based on minimum edit distance of two python files. It returns normalized matching percent.

Email and logging feature is implemented to keep track of errors and logs generated for system.

Backlog Statistics:

Sprint (Spring 2018)	Number of Issues Assigned	Number of Issues Completed	Number of Issues in Backlog
Feb 18 - March 4	20	20	0
March 12 - March 23	30	30	0
March 26 - April 6	31	31	8
April 7 - April 14	100 (including duplicate Bugs and Tasks)	94	6

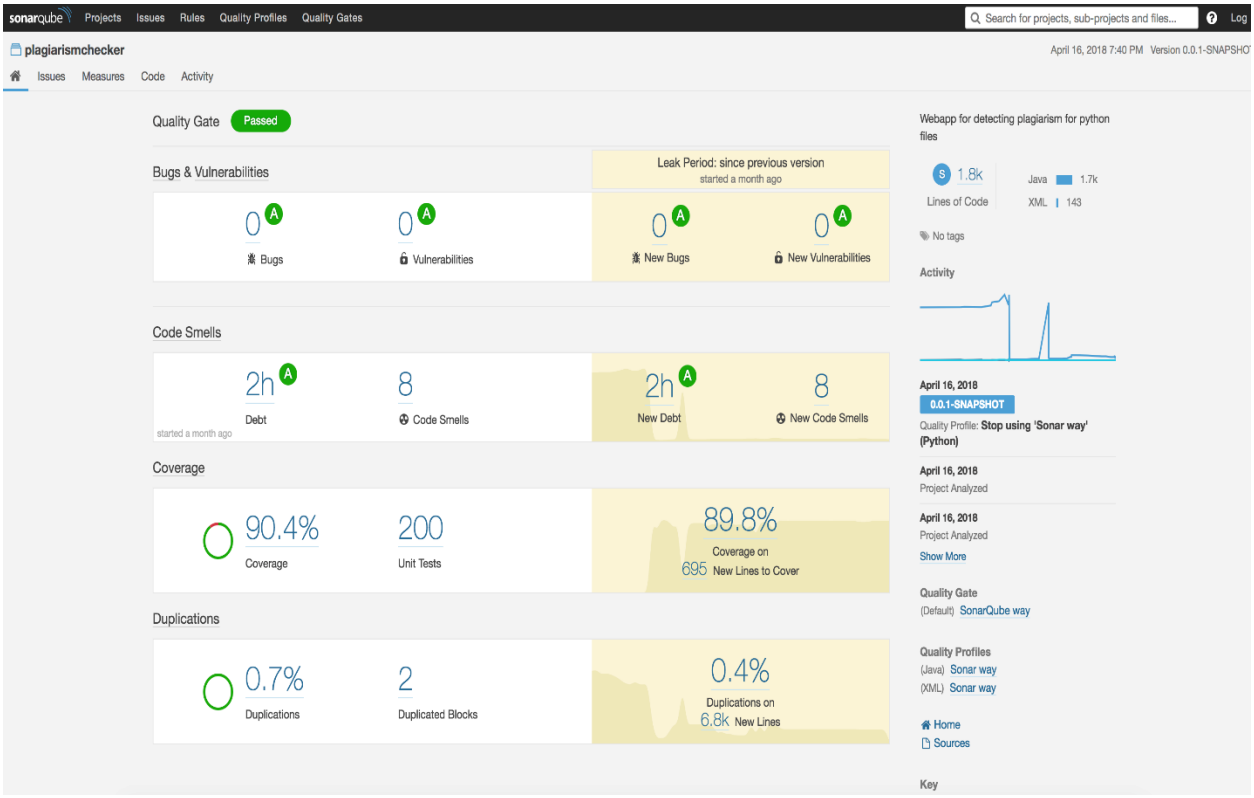


Quality claims supported by testing data :

We have used test driven approach for software development . Unit and Integration testing is used in order to test code implemention. Black box testing is done to validate system functionality.

Incorporated Jenkins for continous integration. Code quality is achieved through SonarLint as well as SonarQube analysis.

Code Coverage : 90.4 %



Development process:

Sprint 1

Functionality Implemented:

1. Basic log in and logout implemented.
2. Used ANTLR to construct an AST for a single python file.
3. Employed a single AST based comparison strategy using Hashmap.
4. Developed UI wireframes.

Environment setup done:

1. Setup Git Repository and restricted the master branch. The work done can be pushed to the master only via pull-requests.
2. The project is created as maven based spring boot application.
3. Work is being managed in Jira.
4. Used smart commits in git.
5. Setup Jenkins to run on all pull-request submissions. Jenkins informs the team about build status through Slack and email.
6. Github informs the team about pull-requests via slack and email.
7. Configured SonarQube in the Jenkinsfile pipeline for quality check.
8. Deployed the system on Amazon AWS.

Sprint 2

1. Implemented a sophisticated Treemap comparison strategy.
2. Implemented more than one comparison strategy on demand using strategy pattern.
3. Used more than one comparison strategy (employed Levenshtein strategy to enable for context match) and report the results of each method.
4. Used the following comparison strategies and computed an overall score using a weighted polynomial function. to scale all algorithm's combined output
   a. HashMap
   b. Tree
   c. Levenshtein

5. Used linear regression to train the strategies mentioned above to learn the weights and to bring it to MOSS as a performance standard.
6. Performed the comparison against multiple files used in two submissions.
7. Implemented interactive UI for professor to register, create courses and homeworks as well as check plagiarism report for submissions done by students.
8. Implemented UI for students to register and submit assignments for homeworks.
9. System logs activity using log4j.
10. System throws an email when error/exception occurs.

Sprint 3

1. System works end-to-end even when the code is split in multiple files.
2. Extensive test suit created.
3. System displays similar code chunks among plagiarized programs.
4. Implemented Admin functionality to approve professor registration.
5. Improved user experience by adding additional feature to sort report summary.
6. Highlighted plagiarized cases in report summary.
7. Enhanced security of web application by implementing sessions.
8. System reports application usage and system statistics information.

Future implementation :

1. CRUD operations on courses.
2. CRUD operations on Homeworks.
3. Help functionality to enable user to navigate.
4. Password encryption.

\

Retrospective:

The course is really helpful to understand full scale industry approach. The project taught us to work in co-ordination with each other as a team in order to have progressive growth in each Sprint. Project was really helpful to learn more about end-to-end application development, integration and deployment.

Surely, we learned more about all industry tools like Jenkins, JIRA, Slack. Configuring Git-Hub Jenkins integration, AWS deployment were good learning points we came across while working on our project. We also think Sonar Qube was really helpful to enforce code quality.

Even though we learnt a lot while building this system we think instead of having same project for the whole class it would be better if teams are given chance to explore different domains and come up with innovative project ideas that deals with real world problems and fulfills course project requirements.

We learned more about best practices, design patterns and agile methodologies and their use in all real time applications. Last homework of testing was more of a turning point for our team to focus on problems that encountered in our project and tackle those difficulties together. It was nice experience to work with responsibility in a team and achieve all goals that we set for our project. We learnt that hustling to only build mediocre software applications falls short in its scope of capturing the essence of quality software.