PLAG DETECTOR

Team 204
Krrish Mittal
Varun Nandu
Ming chao wu
Harsha Rahul Boggaram

SYSTEM FUNCTIONALITY

FUNCTIONAL REQUIREMENTS

- We as a team, we were able to finish all the basic functional requirements
- For each sprint we were able add many stretches as additional features for our system.
- Our system ‘Plag Detector’ is designed to primarily reduce the load and automate the process of checking for Plagiarism.
  - Receive and store assignments
  - Automatically run detection engine with the user base
  - And notify the necessary people when required
  - with least user interaction
  - Display all the results in user friendly Graph format for visualization.

GETTING ASSIGNMENTS

- The Student can create account and upload assignments for the courses they are registered.
- The assignments can be zipped file of the entire project or Github link of their project.
- The student can submit multiple times to the same assignment but cannot overwrite the previous submissions and they will be always tracked.

RUNNING DETECTION

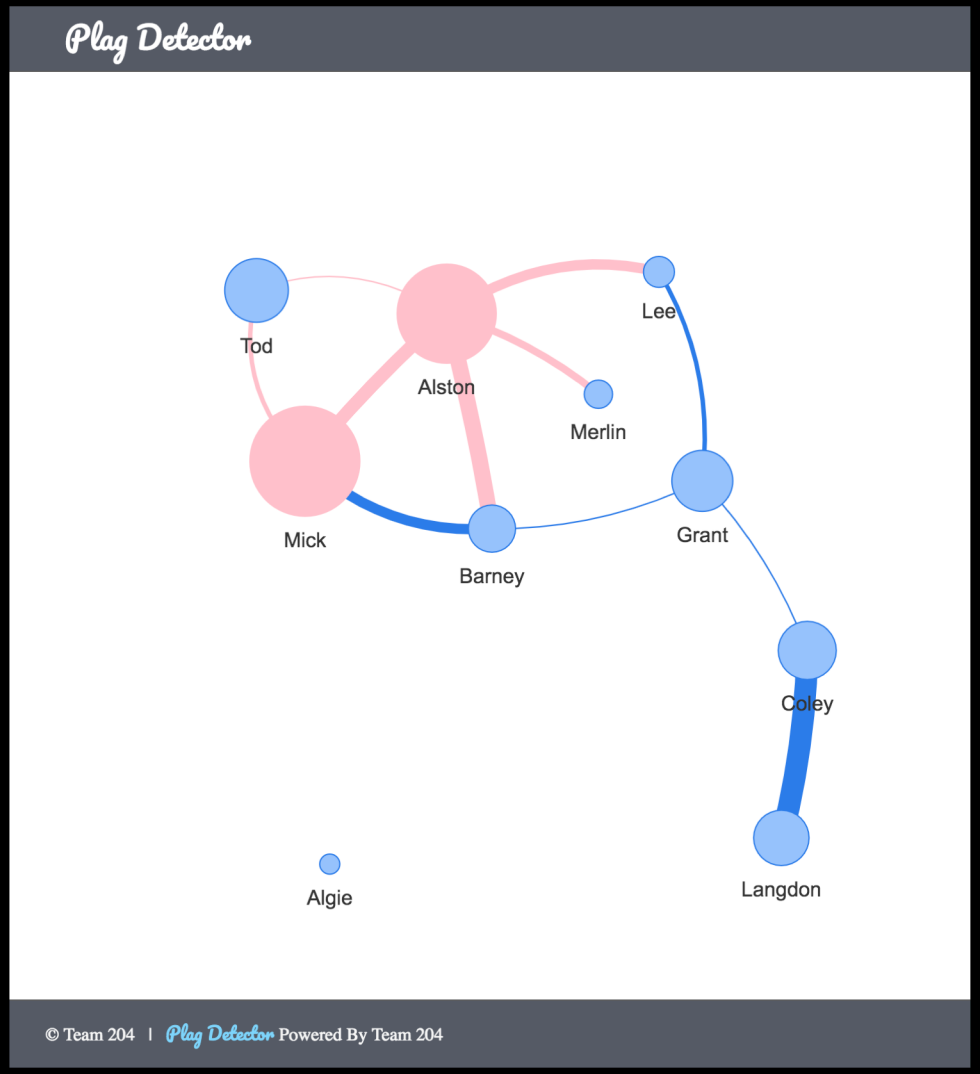
- The system can detect plagiarism for four languages which are Python, Java, C, C#
- The under the hood algorithm which is detecting plagiarism is Jplag library.
- The system runs comparison for every upload of assignment with other students assignments.
- All results are saved as snapshots at that point of time
- New snapshots are created for every upload
- All snapshots are maintained and can be viewed anytime by Professors.

VISUALIZATION

- The Professor when logs in can see the notifications of the latest detections.
- He can view all submission and run an on demand run for plagiarism.
- The professor can view results for each assignment for every semester .

GRAPH

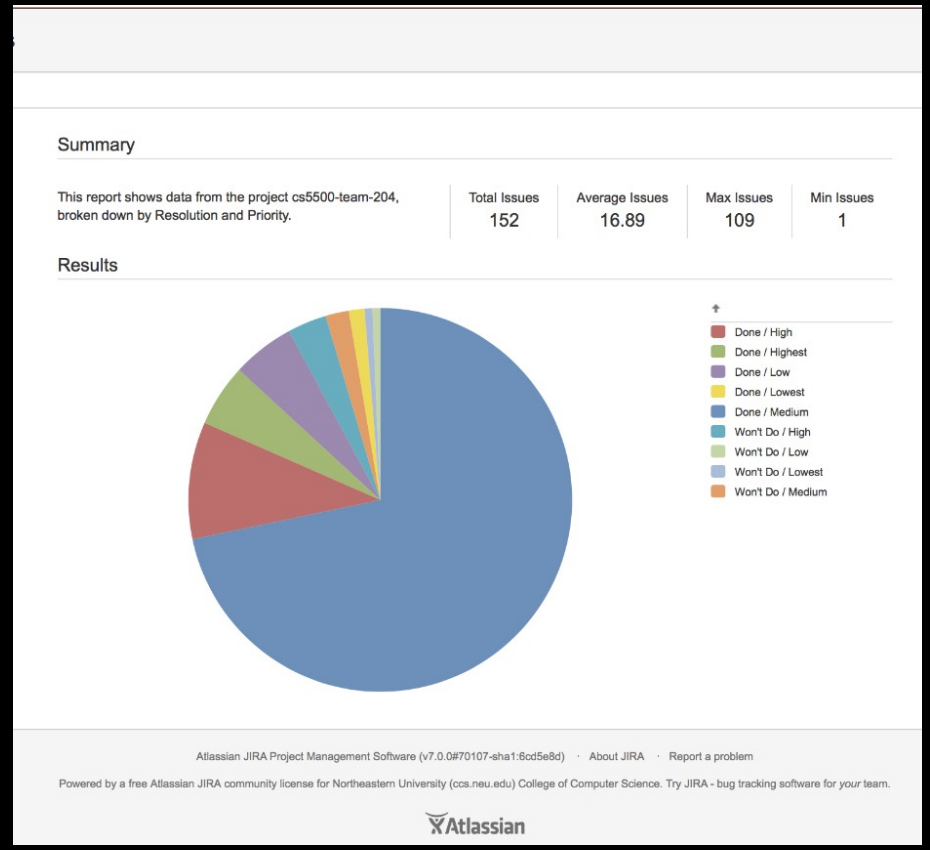
- The results are displayed in the form of a graph.
- Nodes represent the students and the edges represent the plagiarism level between them.
- The students who have crossed the threshold will be highlighted in different color.
- When the edge is selected the relation and the files plagiarized between them is show for further analysis
- The professor can also send additional email notification to the concerned team.
- The professor can view the plagiarized files side by side where the copied code is highlighted.



COMPLETION & SYSTEM USEFULNESS

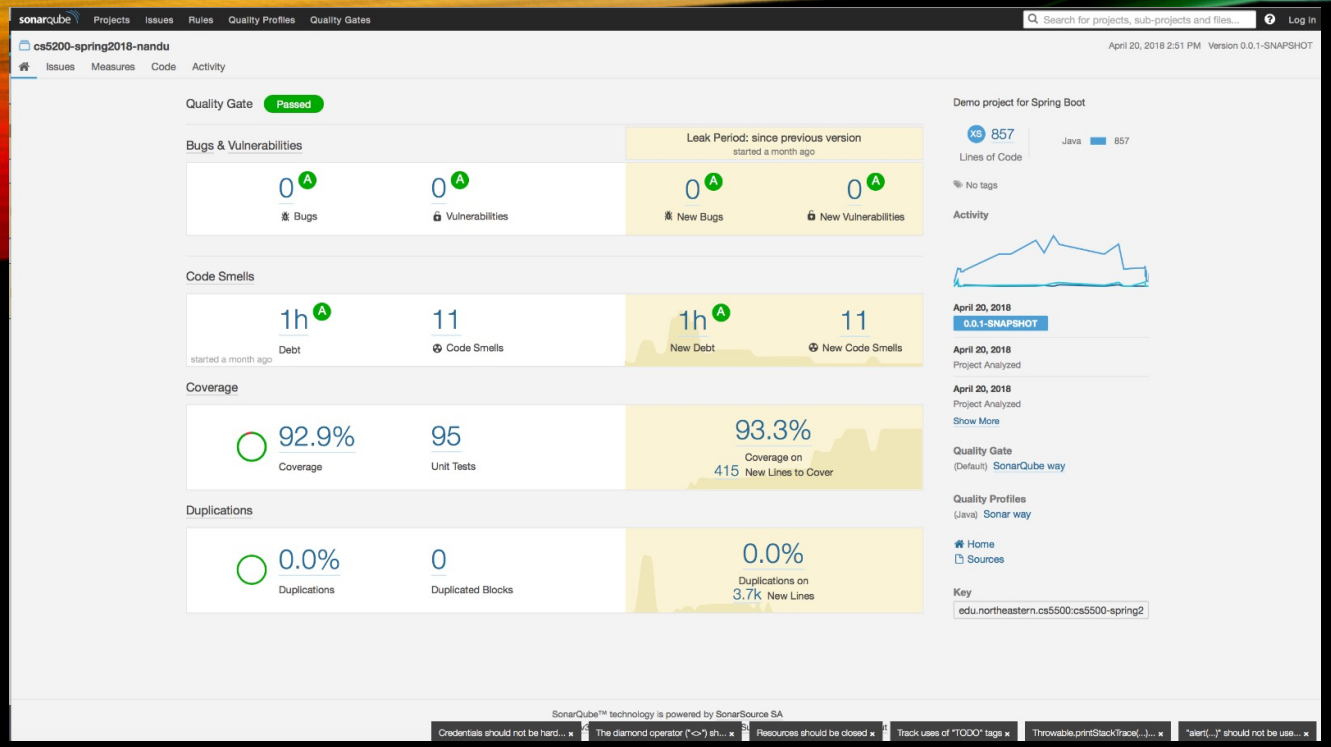
- The before mentioned features mentioned and running on AWS.
- The various features makes it really useful for the people who use it regularly
- Also for professors who monitor it and take action only when required.
- The system UX and UI as we feel is user friendly and easy to use and adapt to any type of system.

- This is the pie chart of backlogs based on priority and importance to functionality.
- We finished all the Basic functionality and also most of the stretches for the project which is indicated by the Jira graph.



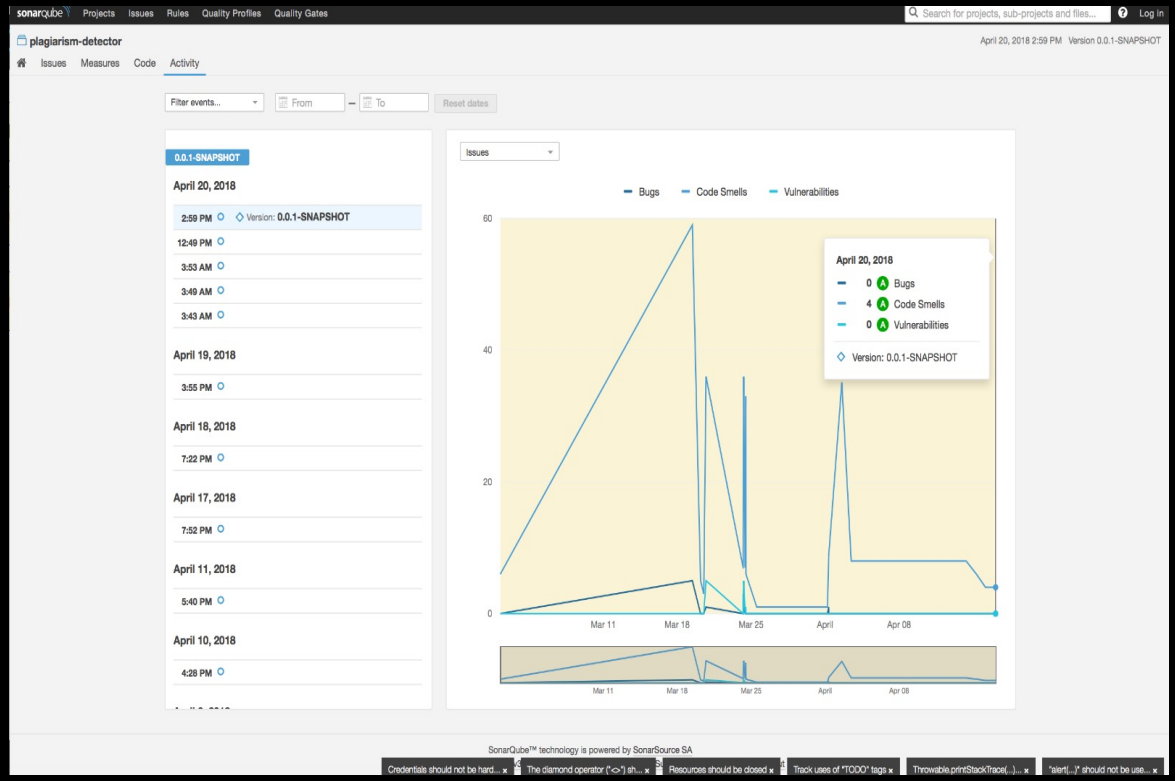
JOB QUALITY

- Quality was a factor that was always kept as a part of the process
- The code was made sure that it was secure, bug free and also has documentation .
- For Code duplication
    - We used Autowired which helped us to reduce Connection code with DB.
    - Used RowMapper to reduce mapping code to our models.
    - Used modular code which can be reused other places



- At the end we were able to bring the bugs, vulnerabilities to zero.
- Reduced the code smells

- The Graph shows that the quality was maintained over time.
- The quality was tackled each sprint and finally we were able to bring all the quality to ‘A’



PROCESS AND TEAMWORK

- Did the team work as a team?
  - We as a team we feel we did work as a single strong unit.
  - We supplemented each other weakness and were able to work around it .
  - Team was always in sync
  - Individual component were developed separately
  - During integration phase the entire team got together and integrated pieces into a working unit

TEAM AND PROCESS WE FOLLOWED

- Yes, we did use the process .
- It can be seen on our Github repo as well
  - The master protected and only merge from branches which were approved were put into master
- All the phases of the project were completed on time.
  - In Phase A
    - we gathered requirements , drafted out a rough plan
    - Selected the target programming language and associated technology
    - We created use cases and mockup UI

- Phase B
  - We created a UML class diagram which signified various entities and relationships
  - We created skeleton interfaces for each of them

- Phase C
  - This was the longest phase in which we covered , implementation, testing and debugging .
  - This phase was divined into three sprints
    - The first sprint included
      - setting up of development environment
      - Basic comparison strategy between files using libraries
    - The second sprint consisted of
      - Creating Different entities for our project like Assignments, Semesters, Users, Courses, Assignments, Submissions and Snapshots
      - Scaling system for multiple users and difference user roles
      - It consisted of working on more advanced comparison strategy across files.

- The third sprint consisted of
  - adding CRUD operations to each entity of project and we added extra functionality
  - Redesigned UI to create a ricer UX experience
  - Rigorous testing and debugging of system
  - Made it stable and deployable.

AUTOMATING THE BUILD, TEST, AND PROCESSES

- We used various technologies to automate, monitor and track our system
  - SonarQube for quality and testing
  - Jenkins for continuous integration and deployment to AWS
  - Jira for maintaining
    - Process
    - Backlog
    - And issue tracking
    - Bug and fixes
  - For team communication we used slack and emails to keep updated and in sync

CHALLENGES ALONG THE WAY

- There were many challenges that were encountered during building the project
  - Deploying Jplag libraries to AWS, the team took ot as a priority and each tried various approaches and finally on approach succeeded
  - The other challenege was integration of the entire system, so team exchnaed parts which were in others expertise and solved them
- So yes, we would say that we worked around challenges successfully.

TECHNOLOGY TRANSFER

FINAL SHAPE OF THE SYSTEM TO BE HANDEDOVER

- *Yes,*
- The system is stable and ready to be handed over to the client with all; functions.
- The various small and big functionalities which make the UX streamlined for the user.
- The system can handle large files and multiple comparisions for students across systems semesters.
- Store historical results to view anytime for the professor and also keep track of latest submissions.

RECOMMENDATIONS FOR THE PROJECT IN FUTURE

- We can use Machine learning techniques like LSTM for to even better the results which are given out by the Jplag libraries.
- We can create AST tree of the files and apply additional algorithms like Levenshtein Distance, LCS and EditTreeDistance to get more ways to detect plagiarism syntactically and semantically.
- We can create use a combination of the above methods to create a weighted polynomial function which will be more accurate and precise then our current method.