



Northeastern University
College of Computer and Information Science

PLAGIARISM DETECTION SYSTEM

CS 5500 – Managing Software Development

Team-110

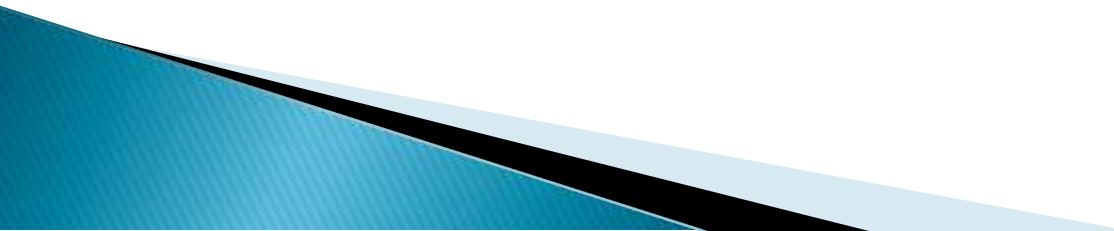
Team Members:

Frenia Pinto

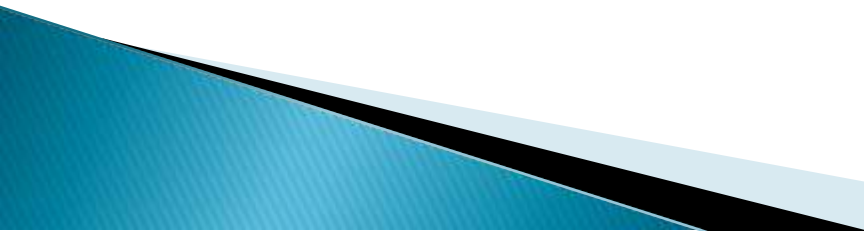
Vaibhav Sukla

Vaibhav Rangarajan


Agenda

- ▶ Problem Overview
 - ▶ Implementation
 - ▶ Considerations and Assumptions
 - ▶ Development process
 - ▶ System Walkthrough
 - ▶ Quality Process
 - ▶ Issues and Challenges
 - ▶ Future Enhancements
- 

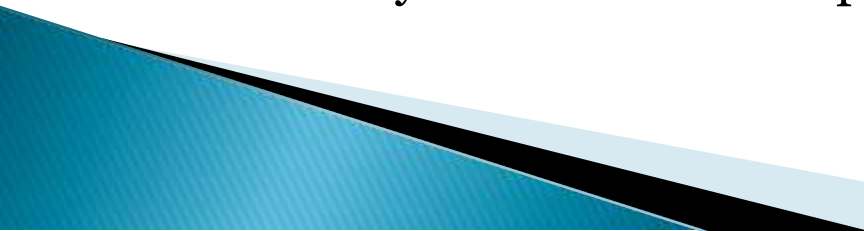
Problem Overview:

- ▶ Unfortunately, plagiarism is a major problem at many schools.
 - ▶ We have considered the specific problem where two or more people produce different versions of the same code.
 - ▶ The goal of this project is to design, implement, test, and evaluate an application to help instructors detect situations where two or more students submit similar solutions to an assignment, in which one version derives from another version through one or more behaviour-preserving transformations such as renaming variables, methods or moving methods to another location in same file etc.
- 

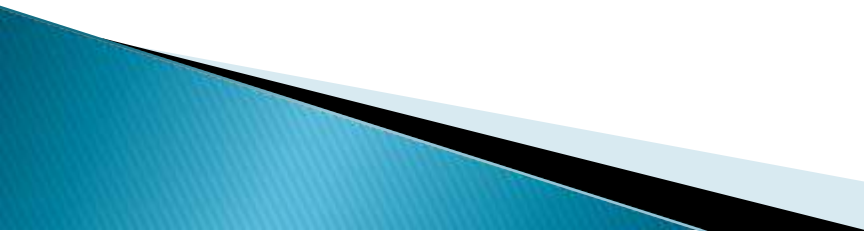
Implementation

- ▶ Our plagiarism detector works on programs written in **python** programming language.
 - ▶ The system is a web application which uses “**HTML, CSS, Bootstrap** and **JavaScript**”.
 - ▶ The majority of implementation of plagiarism detection algorithm is implemented in “**Java**”.
 - ▶ The system uses a “**postgresql**” database.
 - ▶ The project follows a **Maven** structure.
 - ▶ The project and database are hosted as an instance in Amazon Web Service.
- 

Considerations and Assumptions

- ▶ The project or code should be uploaded to a [github](#) repository.
 - ▶ We have assumed the directory structure similar to [CS5500 github repository](#).
 - ▶ The plagiarism detector detects plagiarism even if the python code is broken down into multiple files under one 'src' or 'test' directory.
 - ▶ The system is designed to compare the assignments/submissions of multiple students who have separate repositories inside CS5500.
 - ▶ The system cannot be accessed by students. Only the instructors can use the system to check for plagiarism of student's submission.
- 

Development Process

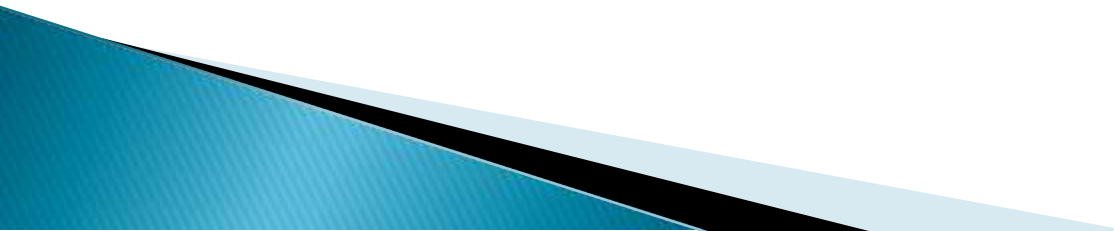
- ▶ Our team followed a modified version of **agile** using scrum. We had stand-ups in person whenever feasible and when not we had **slack-ups** to update the status and progress.
 - ▶ The **TDD**(Test Driven Development) approach was used to implement many modules of our system.
 - ▶ Our team used **Github** for version control.
 - ▶ **Jira** to manage tasks assigned for each sprint.
 - ▶ **Jenkins** and **SonarQube** was used as quality measures.
- 

System Walkthrough

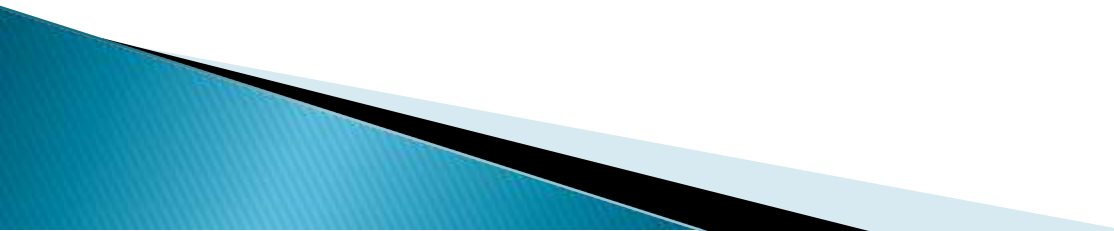
Git Hub User Name:

- After login, the first input to the system is the git hub user name.
- For example, “CS5500” , under which there are lot of student repositories. As stated earlier, the system assumes the repository structure of “Cs5500”.
- Given the user name, all the repositories will be cloned into the system and will be displayed to the user.

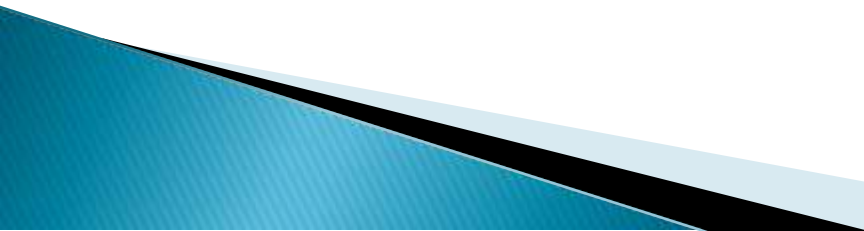
Filter Homework:

- The list of student repositories will be displayed, wherein the instructor can filter the homework or run the detector on all the homework.
 - Once a filter is chosen the list of filtered homework of each student repository is displayed, and the plagiarism detector can be run.
 - The plagiarism detector compares the homework of each student with every other student's repository.
- 

Report Overview and Detailed Report:


- Once the Plagiarism detector is run, a report overview is displayed showing a bar graph of the high, medium and low plagiarism content.
 - Once the View Detailed Report is selected, a list of students sorted based on High, Medium and Low plagiarism content is displayed.
 - Upon selection of a student pair, a detailed report with matching code highlighted and the split percentage of plagiarism computed by each algorithm is displayed.
- 

Usage statistics and logs:

- Usage statistics is displayed as a line chart with the date and the number of reports generated.
 - The logs are stored for every session the detector is used.
 - In case of any exception of system breakdown an email will be sent to the Dev team with a stack trace of the error encountered.
 - All the login information and reports generated are stored in the database.
- 

Quality Process:

GitHub and Quality process:

- ▶ The team used [Github](#) for version control and a separate branch was created for each feature implementation.
 - ▶ Once each development of feature branch is unit tested by the developer, it was promoted to a [Development](#) branch.
 - ▶ Once a particular functionality has been developed, it was merged into the Development branch to avoid redundant code being merged directly to master.
 - ▶ Upon passing of all test cases a pull request was created to merge the Development with master.
- 

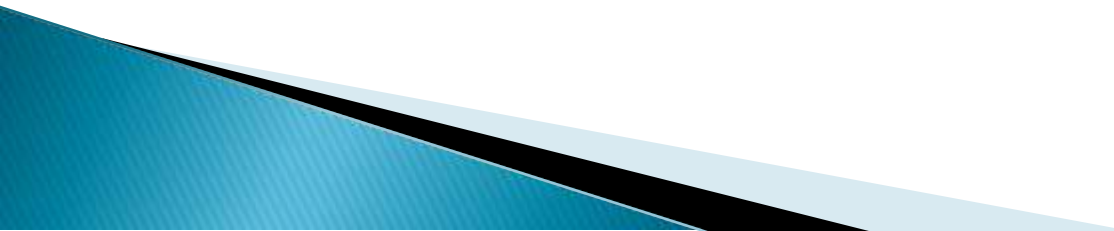
Git working on Branches:



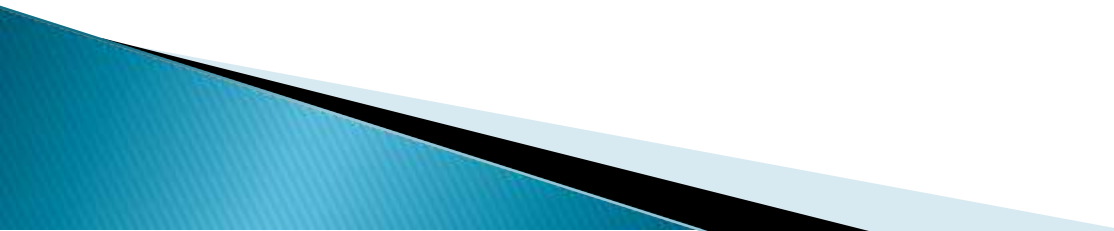
Jenkins and Sonarqube:

- ▶ For each pull request, **Jenkins** runs a set of **Junit** tests to make sure there are no conflicts.
- ▶ The final stage of Jenkins is the **SonarQube** which assesses the system quality based on test coverage, bugs, vulnerabilities etc. and allows the code to be merged to master only when it passes the SonarQube quality gate.
- ▶ Jenkins and github pull request notifications in slack.
- ▶ Once the code is merged to master, the code is deployed to AWS.

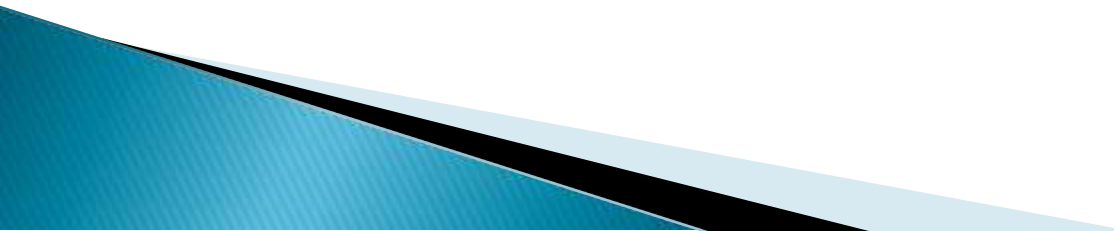
Quality Assertion:

- ▶ The system has passed the [SonarQube](#) quality gate with test coverage of 90.2% and [51](#) unit tests.
 - ▶ All the Java programs are supported with Javadoc and function description.
 - ▶ The system has received a reliability, security and maintainability rating of 'A' in SonarQube.
 - ▶ The system is tested with test cases of various scenarios such as full, partial and no plagiarism and all essential part of the algorithms have been exercised and also the services have been tested with [mockito](#).
 - ▶ All the bugs raised by the testers have been fixed.
- 

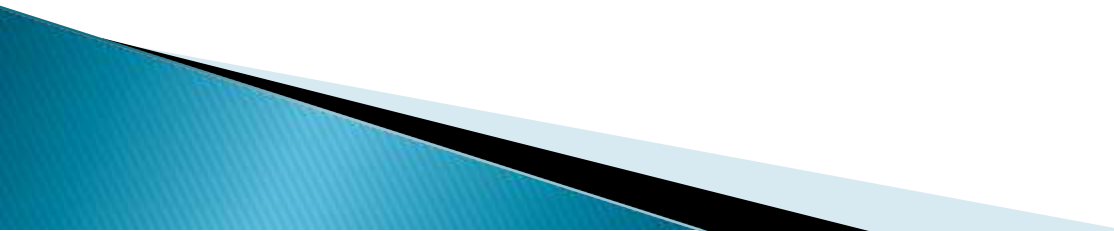
Jira for Project Management:

- ▶ The team used Jira to manage tasks assigned for each sprint.
 - ▶ For each sprint the expectations would be framed as a ticket in Jira and a developer would be assigned to it.
 - ▶ Once the development for a particular ticket is done, it will be closed and hence all progress of the team can be tracked through Jira.
 - ▶ All the bugs and backlogs can also be tracked through the team's Jira page.
- 

Issues and Challenges

- ▶ All our team members had similar skill set and had no experience in User Interface, hence had to do a lot of learning on the front-end side.
 - ▶ Our team faced some issues with AWS environment and Jenkins had memory issues which were resolved with the help of peers through discussion in Piazza.
 - ▶ Time management was a problem initially which we faced as the project progressed.
- 

Future Enhancements

- ▶ More sophisticated User Interface
 - ▶ Enhanced algorithm and implement machine learning through usage of the system and human training.
 - ▶ Download report as PDF.
 - ▶ Compatibility with any repository structure.
 - ▶ Introduce more homework and file filters.
 - ▶ Plagiarism detection on more programming languages.
- 

Thank You..

