TEAM-208

- Aman Rayat
- Harshmeet Johal
- Rachana Tondare
- Viha Bidre

PLAGIARISM DETECTION APPLICATION

SYSTEM FUNCTIONALITY

AMAN RAYAT

FUNCTIONALITIES ACCOMPLISHED

- Students can submit the GitHub link of the repository containing the source code after registering themselves.
- Students can also upload a zip file of the folder containing the source code.
- A Professor can run the plagiarism check for the students of the same class, different sections and even across different semesters.
- Professor can set a threshold above which the plagiarism check will be run and corresponding reports will be generated.
- Emails can be sent to the professor and the student intimating them of the impending danger.

GOALS …… SET AND ACHIEVED

- Initially the goal set for the application was to detect plagiarism in Python using some complex algorithms which can detect any kind of plagiarism that a student can possibly think of.
- Requirements changed with time.
- The requirements changed to make the process more convenient for the client.
- We successfully achieved almost all of the requirements that were set by the client.
- We delivered the application which detects plagiarism not just across sections but also across semesters in Python and 6 other languages with detailed reports being generated and emailed to the respective Professors.

USES OF OUR APPLICATION

- The most useful feature of our application is that, it catches **PLAGIARISM**.
- Our application is useful for the clients since it makes life easier for both, the students as well as the professor.
- One of the major feature of our application is that the client does not need to do anything other than just clicking a button and wait for the students to be caught (if any).
- Our application can be used to check plagiarism in different languages.
- Reports are generated highlighting exactly the lines of code that are plagiarized.

WHAT WE PLAN TO DO…..

- The future of the project can be that the professor does not even have to run the plagiarism report himself. The plagiarism check will be done automagically and professor will just get a notification if the student uploads a plagiarized solution.

EVIDENCE

CS208 board

Backlog

Board

QUICK FILTERS: Only My Issues Recently Updated

VERSIONS EPICS

~~CS208-3~~ Document discussion with Professor	/WebappArchitectur…	3
~~CS208-2~~ Connect Mysql database to springboot maven project	/IntegrationEnviron…	8
~~CS208-1~~ Setting up the maven environment and database connection	/IntegrationEnviron…	5
~~CS208-4~~ Search library to create AST from a single python file	/AlgorithmImplemen…	
~~CS208-5~~ Create and test AST using ANTLR from python file	/AlgorithmImplemen…	
~~CS208-6~~ Integrate and test AST with master	/AlgorithmImplemen…	
~~CS208-8~~ connect moji with MOSS	/AlgorithmImplemen…	
~~CS208-9~~ Test MOSS with python files	/AlgorithmImplemen…	
~~CS208-11~~ Research AST comparison tools	/AlgorithmImplemen…	
~~CS208-12~~ Integrate gumTree AST tool	/AlgorithmImplemen…	
~~CS208-29~~ Set-up AWS-Jenkins	/IntegrationEnviron…	5
~~CS208-20~~ Create Home Page for the web-app and users	/UIFrameworksForP…	3
~~CS208-19~~ adding application to Heroku	/IntegrationEnviron…	
~~CS208-30~~ Set-up AWS project	/IntegrationEnviron…	8
~~CS208-31~~ Set-up SonarQube on Jenkins	/IntegrationEnviron…	8
~~CS208-32~~ create springboot project	/IntegrationEnviron…	5

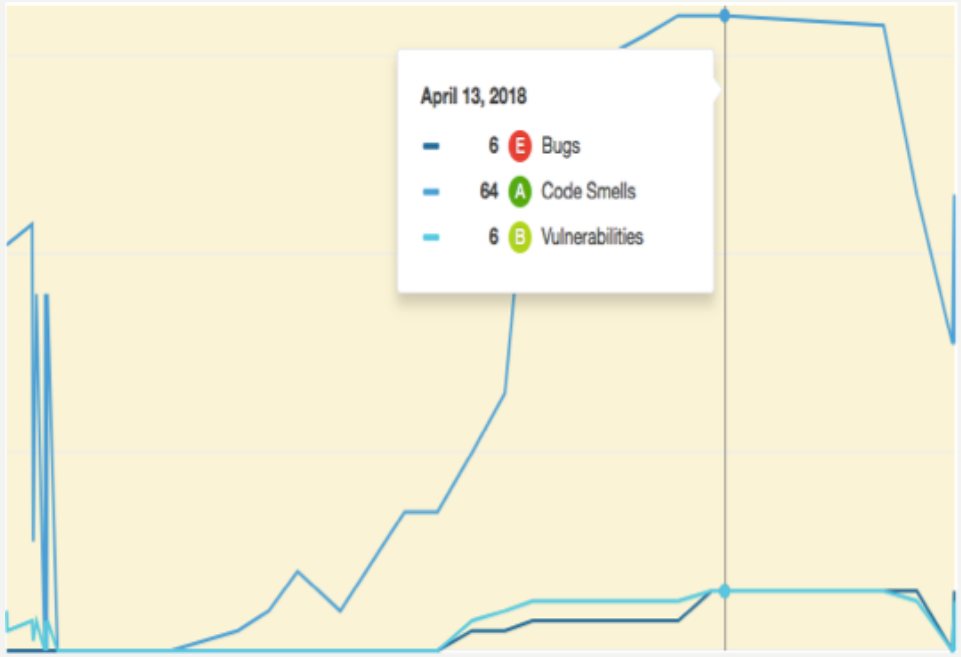
evidence.gif

Show All

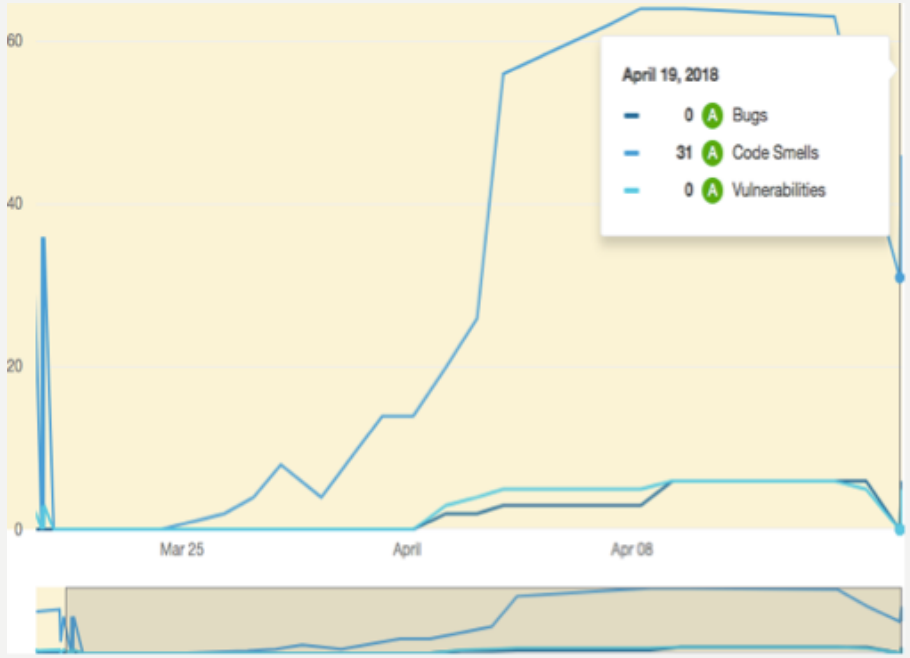
JOB QUALITY

DEVELOPMENT QUALITY

Status as on April 13,2018



Status as on April 19, 2018



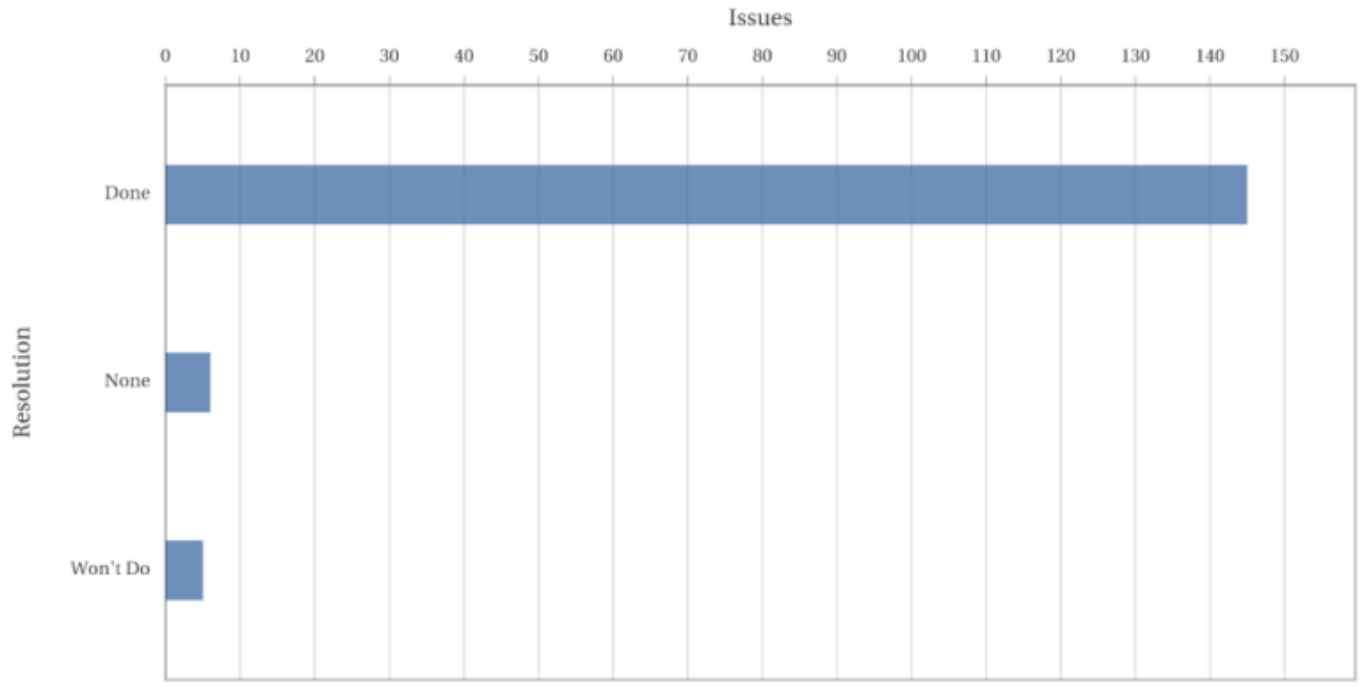
PRODUCT QUALITY

Summary

This report shows data from the project cs5500-team-208, broken down by Resolution.

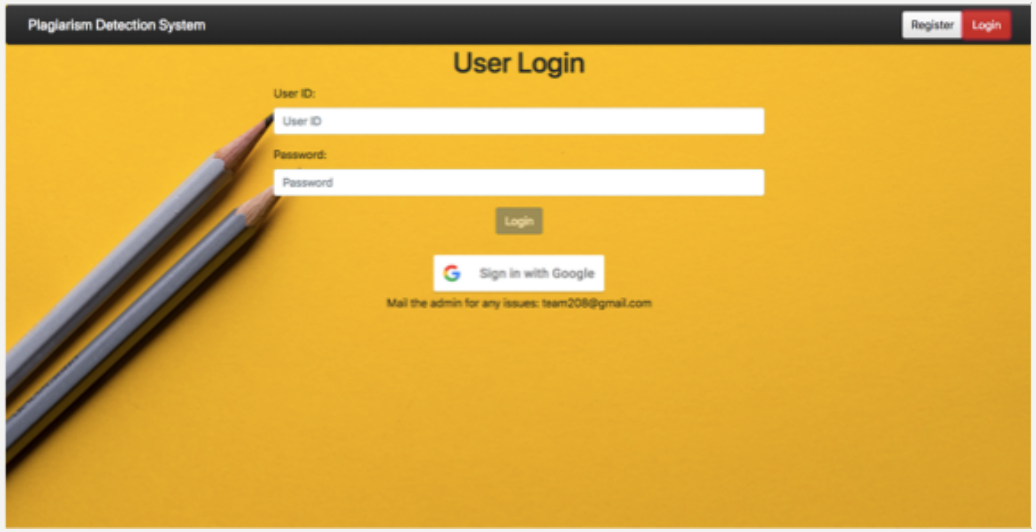
Total Issues	Average Issues	Max Issues	Min Issues
156	52.00	145	5

Results



TYPES OF ISSUES RESOLVED

- Product Design
- Code Quality
- Security



Search Results

Title:	
Programs:	201 - 205
Language:	Java1.7 Parser
Submissions:	2
Matches displayed:	1 (Treshold: 99.0%) (average similarity) 1 (Treshold: 99.0%) (maximum similarity)
Date:	2018-04-19
Minimum Match Length (sensitivity):	9
Suffixes:	.java, .jav, JAVA, JAV
Course ID	CS5500
HomeWork Id	5
Tokens	Display the matched areas

Matches sorted by average similarity (What is this?):

205 -> 201 (99.0%)

Matches sorted by maximum similarity (What is this?):

205 -> 201 (99.0%)

PERFORMANCE PARAMETERS

- Delegation of responsibility
- Defects Resolution Speed
- Number of stretches covered per sprint
- Contributions to master

DELEGATION OF RESPONSIBILITY BY SKILLSET

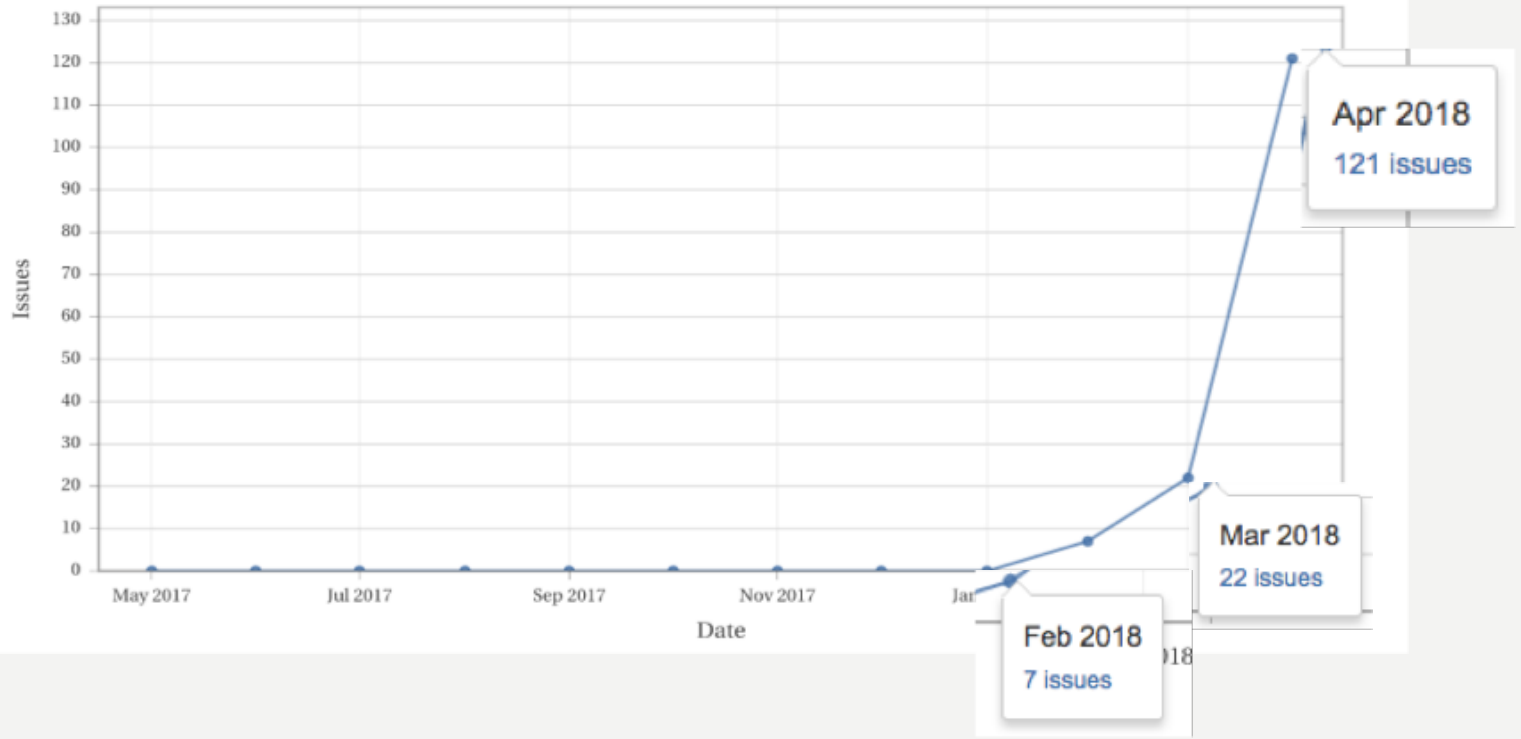
DEFECTS RESOLUTION SPEED

Summary

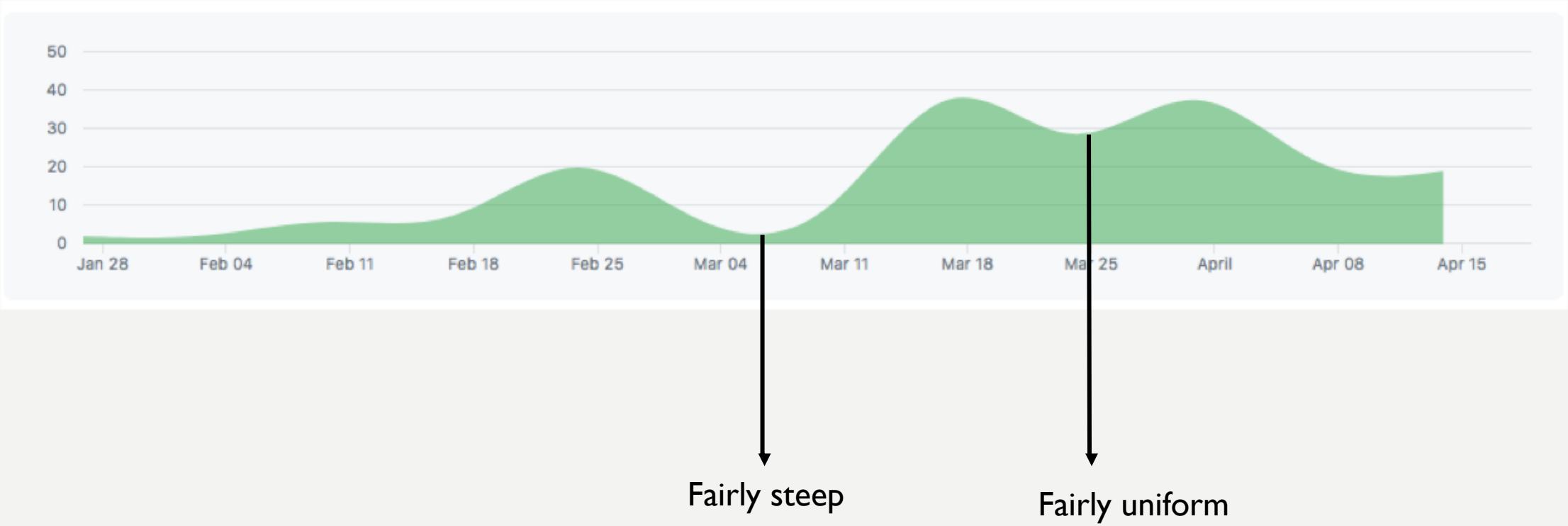
This report shows the count of issues by Resolution Date from the project cs5500-team-208, from May 1 2017 to Apr 30 2018 (grouped monthly).

Total Issues	Average Issues	Max Issues	Min Issues
150	12.50	121	0

Results



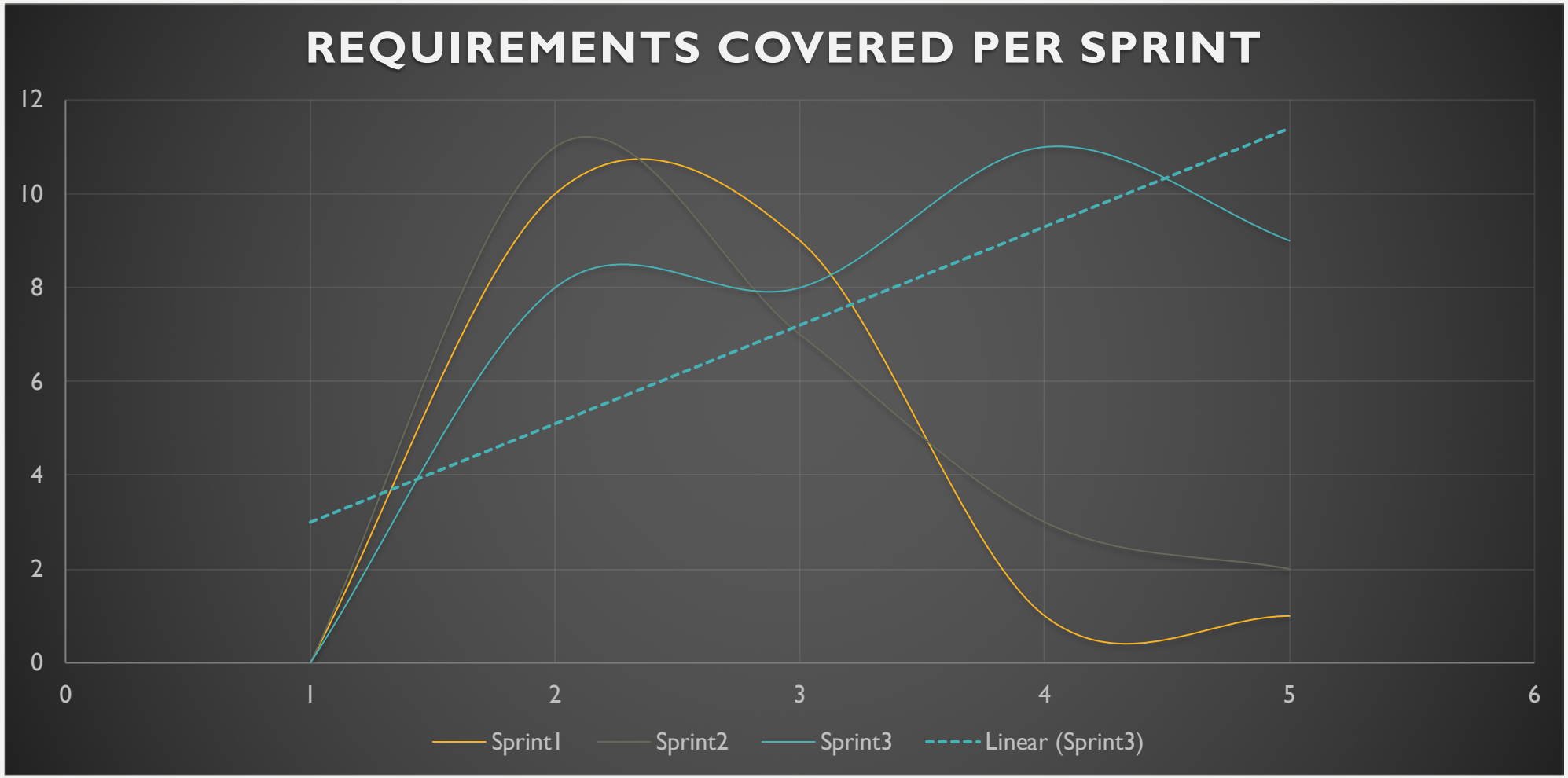
INDIVIDUAL CONTRIBUTION SCALE



NUMBER OF EXPECTATIONS COVERED PER SPRINT

Project phase-C	Total Base Expectations	Base Expectations Covered	Total Stretches	Stretches covered	**Coverage Percentage**
Sprint 1	10	9	1	1	**90.9%**
Sprint 2	11	7	3	2	**65%**
Sprint 3	8	8	11	10	**94.7%**

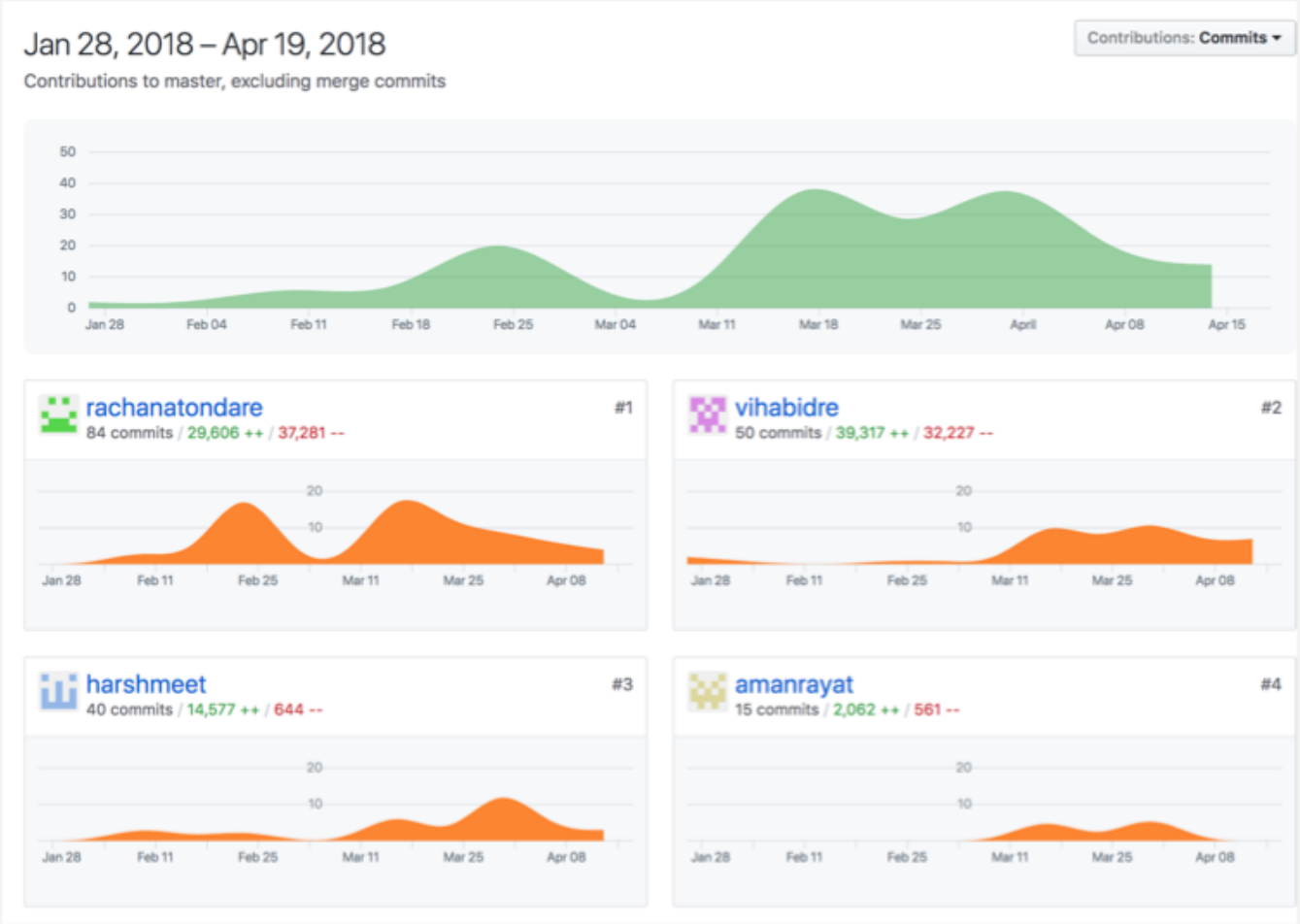
NUMBER OF EXPECTATIONS COVERED PER SPRINT



PROCESS AND TEAMWORK

BY RACHANA TONDARE

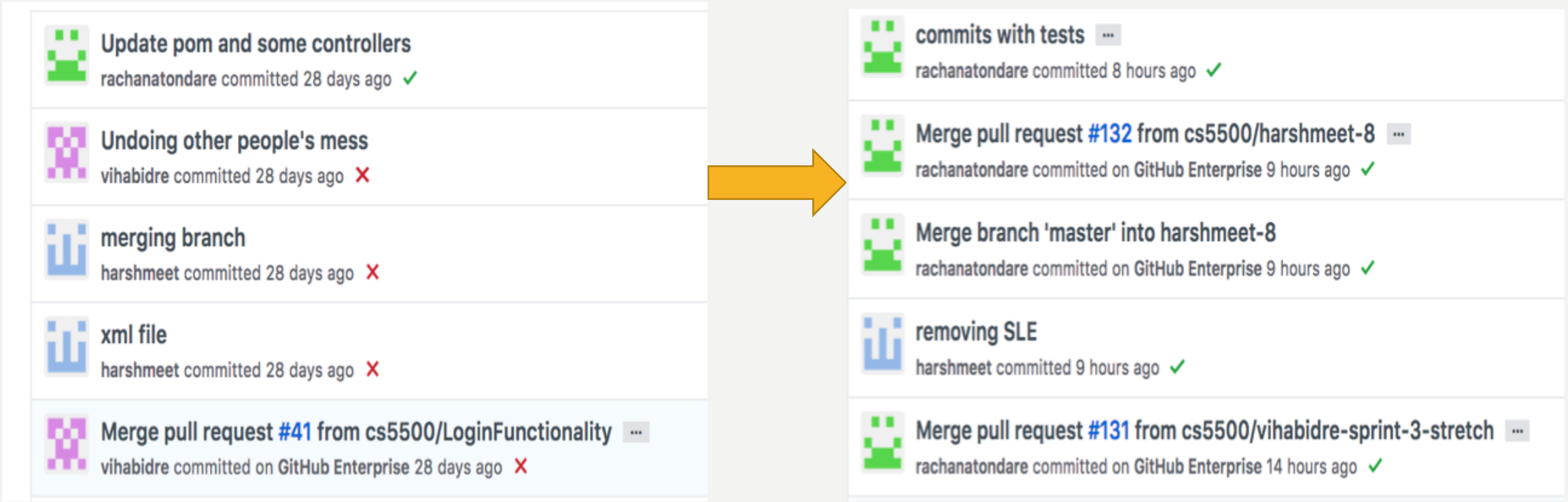
TEAMWORK?



- Initial planning and task distribution was not well organized.
- With time team rapport improved and team members took up initiative to do sprint requirements
- Team performance and the quality of end product improved
- Also the git commit graph is evidence to the team efforts put in.

PROCESS

TO A MORE ORGANIZED TEAM PROCESS

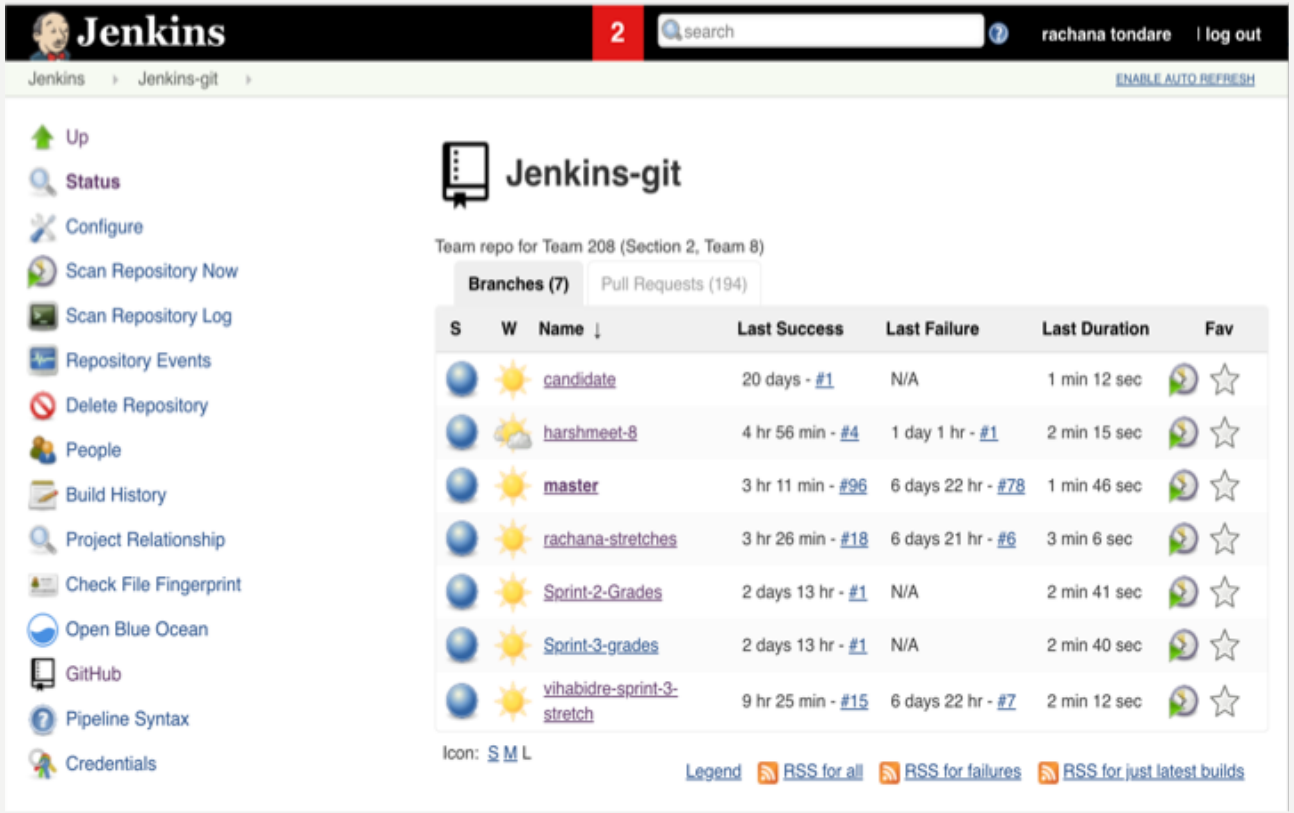


From the above images it is evident the progress the team made after strictly following the process and the lessons learnt the hard way. So we did use the process the well gradually.

BUILD AUTOMATION ACHIEVED THROUGH JENKINS AND GIT WEBHOOKS

```
pipeline {
    agent {
        docker {
            image 'maven:3-alpine'
            args '-v /root/.m2:/root/.m2'
        }
    }

    stages {
        stage('Build') {
            steps {
                echo "Build"
            sh 'mvn -f PhaseC/PlagiarismDetector/pom.xml clean'
                sh 'mvn -f PhaseC/PlagiarismDetector/pom.xml compile'
                sh 'mvn -f PhaseC/PlagiarismDetector/pom.xml package'
            }
        }
        stage('Test'){
            steps {
                echo "Testing"
            sh 'mvn -f PhaseC/PlagiarismDetector/pom.xml clean test'
                sh 'mvn -f PhaseC/PlagiarismDetector/pom.xml test'
            }
        }
    stage('SonarQube') {
            steps {
                withSonarQubeEnv('SonarQube') {
                        sh 'mvn -f PhaseC/PlagiarismDetector/pom.xml clean package'
                        sh 'mvn -f PhaseC/PlagiarismDetector/pom.xml sonar:sonar'
                }
            }
        }
    stage('Quality') {
            steps {
              sh 'sleep 30'
              timeout(time: 10, unit: 'SECONDS') {
                retry(5) {
                script {
                  def qg = waitForQualityGate()
                  if (qg.status != 'OK') {
                error "Pipeline aborted due to quality gate failure: ${qg.status}"
                }
                }
                }
                }
            }
        }
}
```



The build and test were automated and processes were promoted, the above images are evidence for it.

SHORT COMINGS

- Maintaining front end and backend development simultaneously was difficult and was over by deploying API individually and maintaining an up to date API documentation for front end developer for reference
- AWS kept running out of space and Jenkins failure issues – Resolution: monitoring and maintenance of AWS Jenkins server each time after builds and tracking frees space before commits
- Tracking task progress in JIRA was an issue as team members were not regular about updating JIRA and hence one team member had to follow up with others and smart commit were thoroughly verified.
- Learning curve took up a lot of time and reduce productivity and so knowledge sharing among team members was done to reduce learning curve and increase team throughput.
- Team faced issues with incoherent code and improper resolution of conflicts which was resolved with one person strictly monitoring commits and code reviews were followed with constructive inputs.

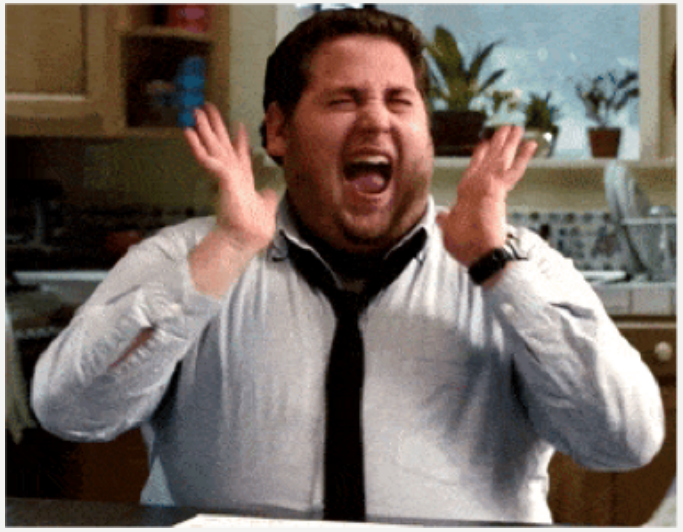
TECHNOLOGY TRANSFER

VIHA BIDRE

CURRENT SYSTEM STATUS

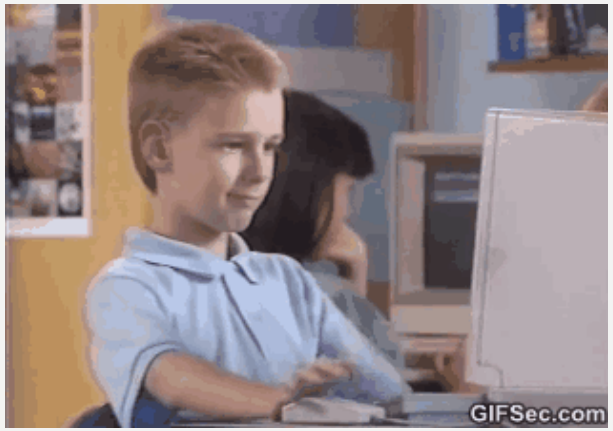
- Production ready and deployed over the AWS server for end users to use the system.

  http://ec2-18-191-0-180.us-east-2.compute.amazonaws.com:3000/



FUTURE SUPPORT READY

- Code is well documented
- Easy to follow system setup
- Logging enabled in case of failures
- API documentation available



- Authors of the code mentioned and their contact details available for any future issues
- Can also contact the admin for any internal issues (details available on the website)

SYSTEM SETUP STEPS

- Run '*run.sh*' script from the root of the project
- Run the command '*npm install*' from '*PlagiarismDetector/msdproject-client*'
- Run the command '*npm start*' from '*PlagiarismDetector/msdproject-client*' to start the UI
- Open *"http://localhost:3000/"* to access the User Interface

SYSTEM UP AND RUNNING IN A FEW SIMPLE STEPS

NEXT STEPS??

- Improvements on the User Interface
- Maintain a cronjob to run plagiarism checks and notify the professor only when plagiarism is detected.
- Integrating more code comparison strategies to achieve more efficient match score.


Gotta catch 'em all!