Plagiarism Detection Tool

CS 5500 - Managing Software Development

Team 206

Members:
Asim Khan
Sanket Mathur
Jiaxin Yang
Jayanth Gangadhar

Agenda

System Functionality - Mohd Asim Khan

Job Quality - Sanket Mathur

Process and Teamwork - Jiaxin Yang

Technology Transfer - Jayanth Gangadhar

System Functionality: Requirements

Professor Weintraub

- Build an application that detects plagiarism
- Include different strategies for comparison
- Weighted polynomial functional to incorporate all strategies
- Different types of submissions allowed (Folders/Files)
- Customised plagiarism threshold value which can be changed on the Users will
- Text highlighting to show similar code

Professor Annunziato

- A portal for Students, Professor, Admin
- Get a taste of Student management system
- Automated report generation system with integrated notification Emails
- Good UX / Least user interaction
- Integrated logging for important events (Registration, Plagiarism detection, etc)

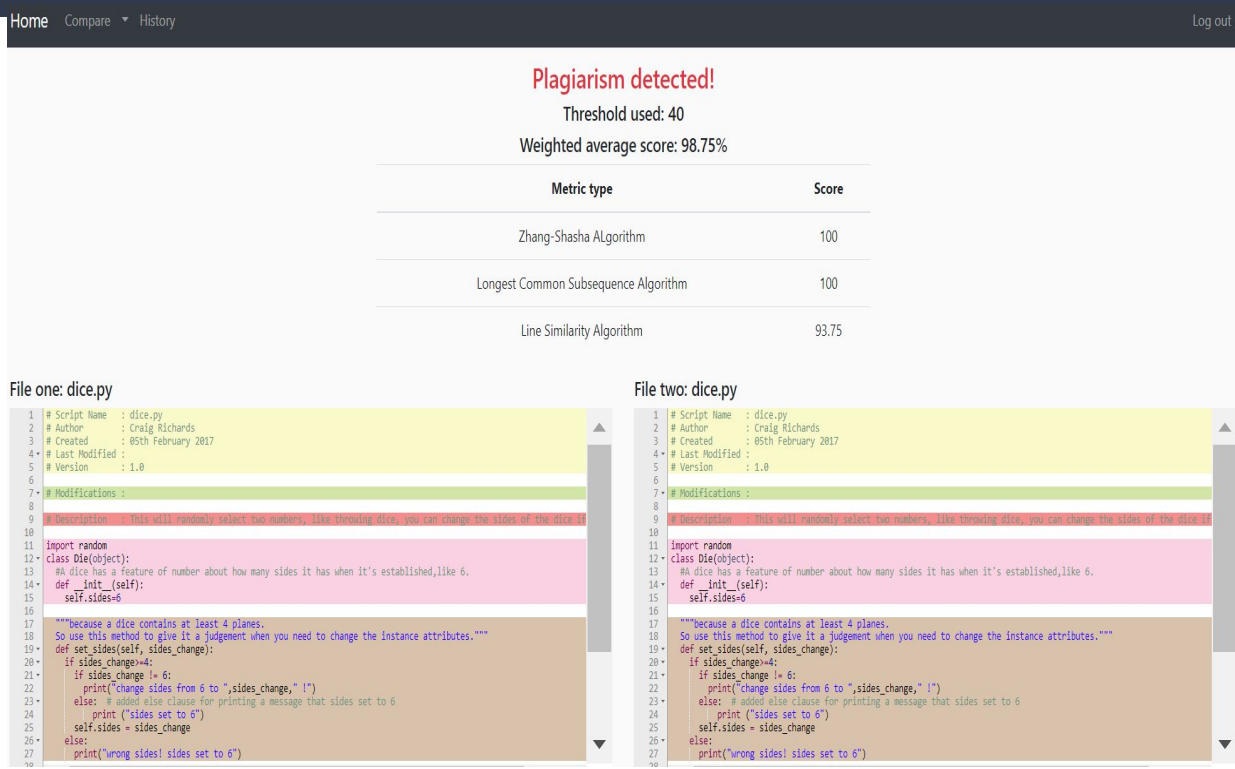
System Functionality: Goals Achieved

We succeeded to build our project based on both the requirements published

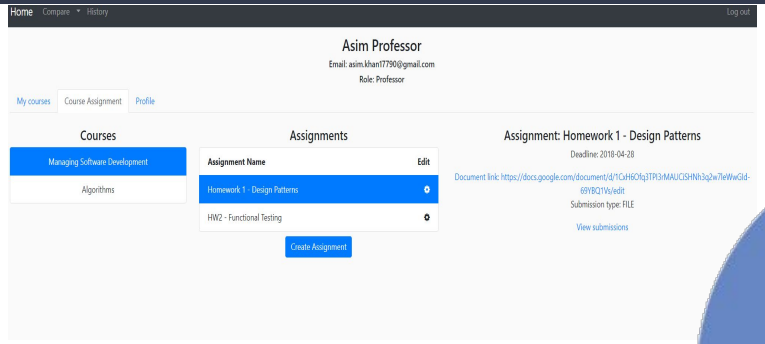
- 3 portals to support Students, Professors and Admins
- Developed 3 Comparison Strategies using ASTs to check for Code Similarity
- A dedicated CRON Job to automatically run everyday to check for new submissions for comparison in an iterative manner
- A great UI which highlights code similarity with even minute details
- Automated Emails integrated with the application to send real time notifications
- Most advanced technologies used while development - AOP, Springs, MongoDB, Angular 4, AWS S3, Gmail API with OAuth2, Mockito, Sonar.

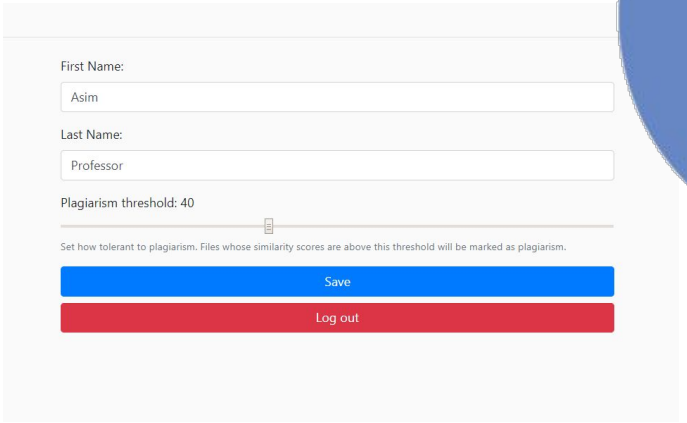
Application Usability

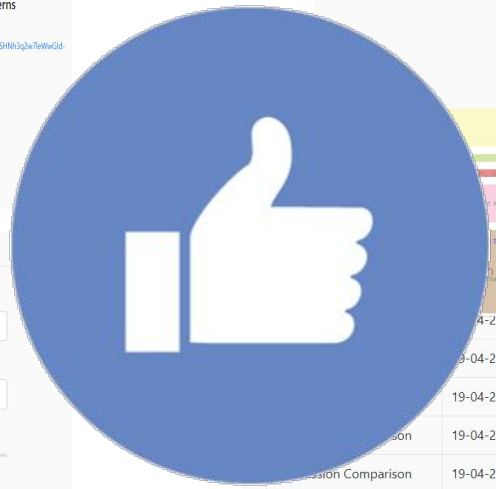
- Intuitive User Interface
- Multiple functionalities available for the user
- Automated as well as Manual Support for Plagiarism Detection
- Automated Email notifications
- Admin portal for God access to all users
- Student portal for course enrollment and homework submission

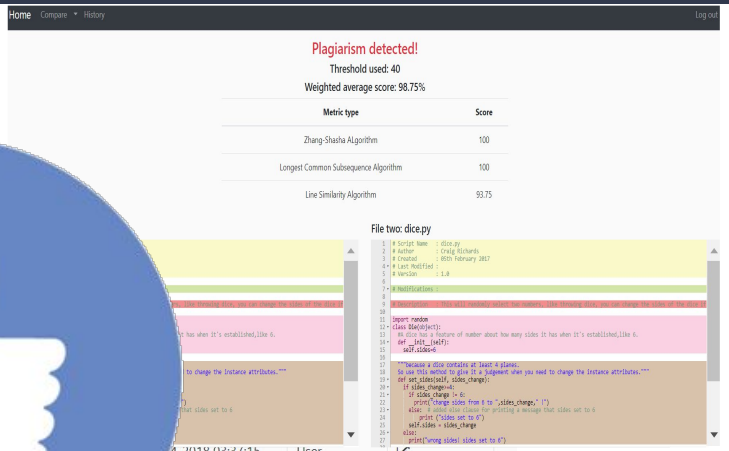


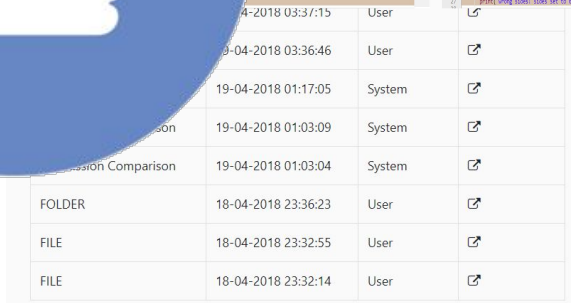
Were we able to do the Job?











Backlogs

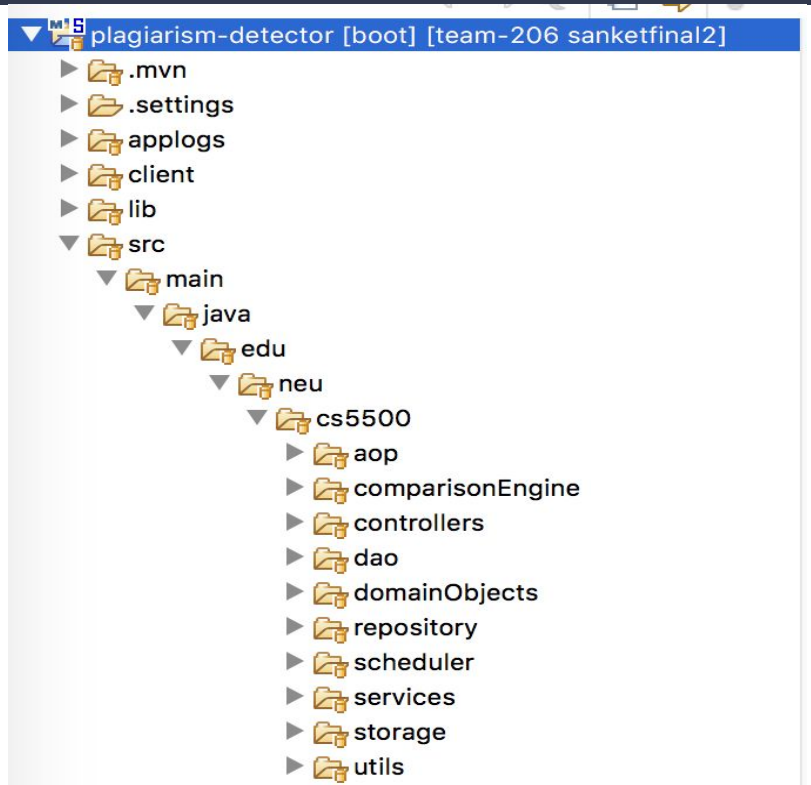
- CS206-93 Large file comaprison failed
- CS206-95 Rest APIs are public
- CS206-130 Trace appears in UI
- CS206-131 When someone logs out, new user when clicks on login, he can see previous user's credentials.
- CS206-143 No tests for MailService class.
- CS206-144 Only one test for FileServiceImpl, 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 happend when refresh the page

Job Quality

- Followed Model View Controller architecture to provide proper structure to the backend system
- Followed quality standards by using SonarLint to maintain the code quality of the application
- Unit test coverage code has been set to at least 85% which includes instructions and branch covered
- Intuitive and simple user interface with color schemes to highlight plagiarism detected files
- Followed Spiral Model to build the UI. Earlier sprints provide very basic functionality and later sprints and final product have richer implementations.

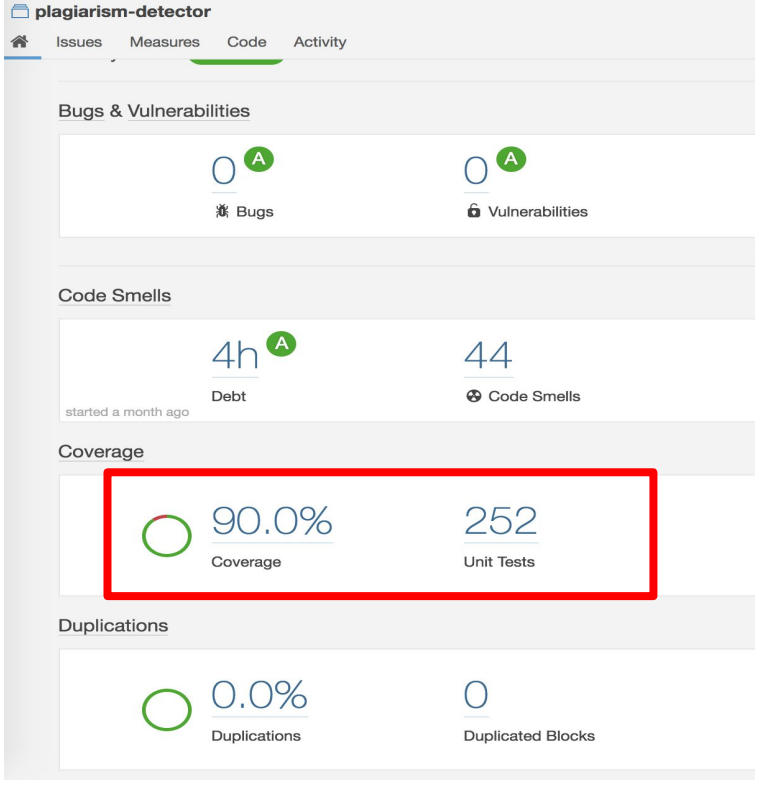
Job Quality: Model View Controller Architecture

- All the controllers will be have the apis which will be used to interact with the front end
- The business logic is present in the service layer
- All the database related work tasks are present in the Data Access Objects (dao) layer



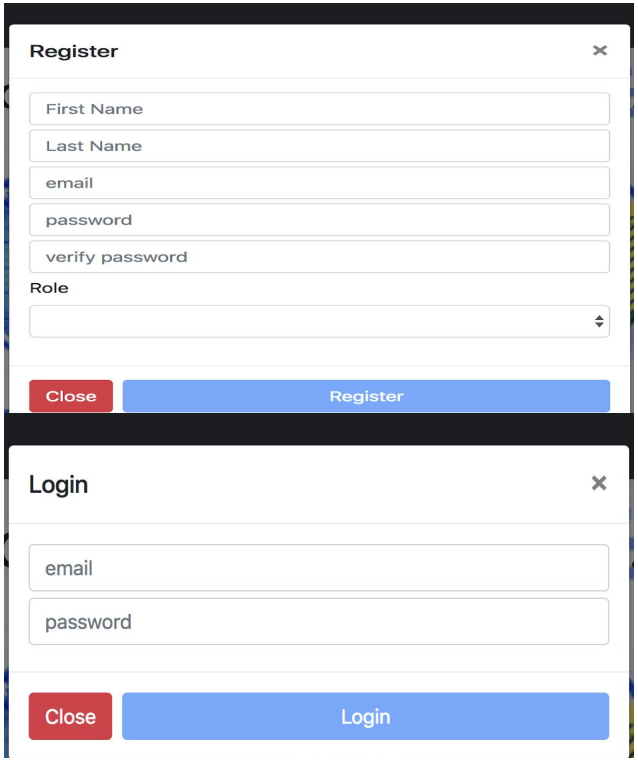
Job Quality: Test coverage

- Lower limit of the test coverage has been set to 85% for the project
- Maintained to cover 90% of the code under test
- 252 total test written



Job Quality: Intuitive UI

- Well organized and simple UI
- No prior knowledge of the system is required to complete their tasks
- Easy to identify the steps users need to complete their task
- Tasks are accessible based on the role of user: Admin, Professor or Student



Job Quality: Intuitive UI

Compare Type: FOLDER

Student List

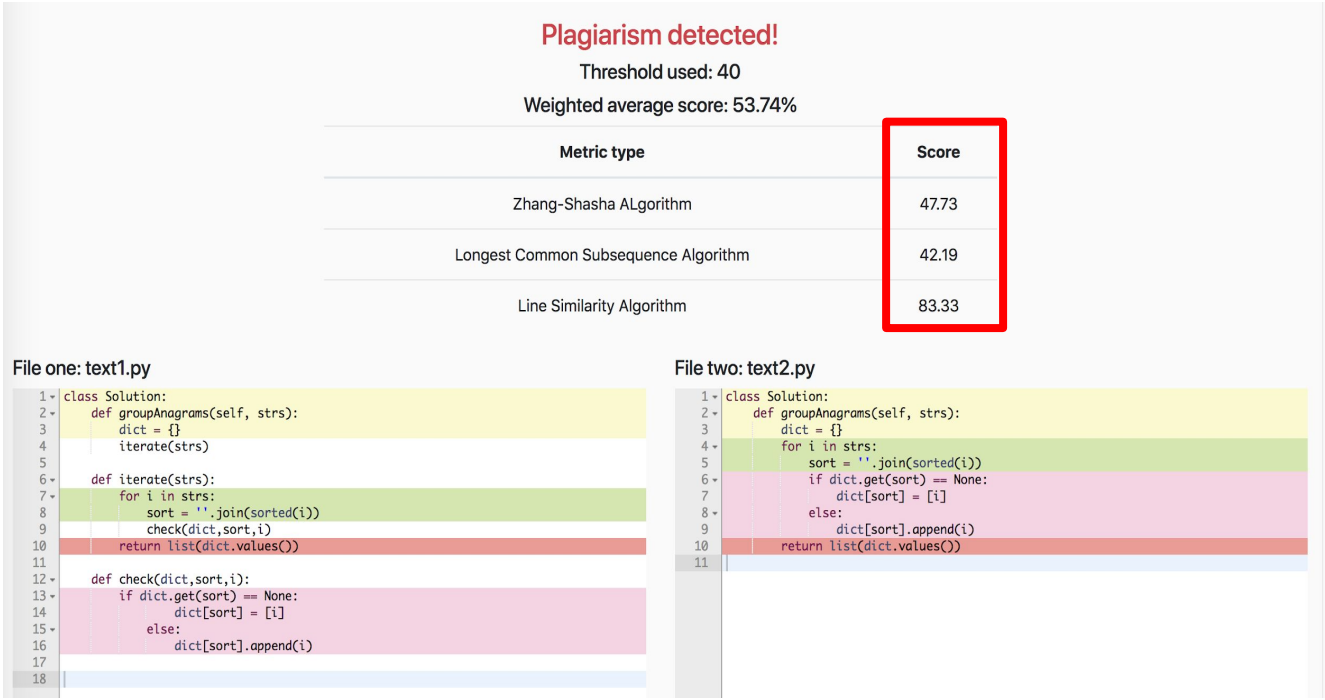
Student 1	Student 2	Result
S1 Linda	Issac Newton	No Plagiarism
S1 Linda	Sheldon Cooper	No Plagiarism
Issac Newton	Sheldon Cooper	Plagiarism Found! Notify students

File List

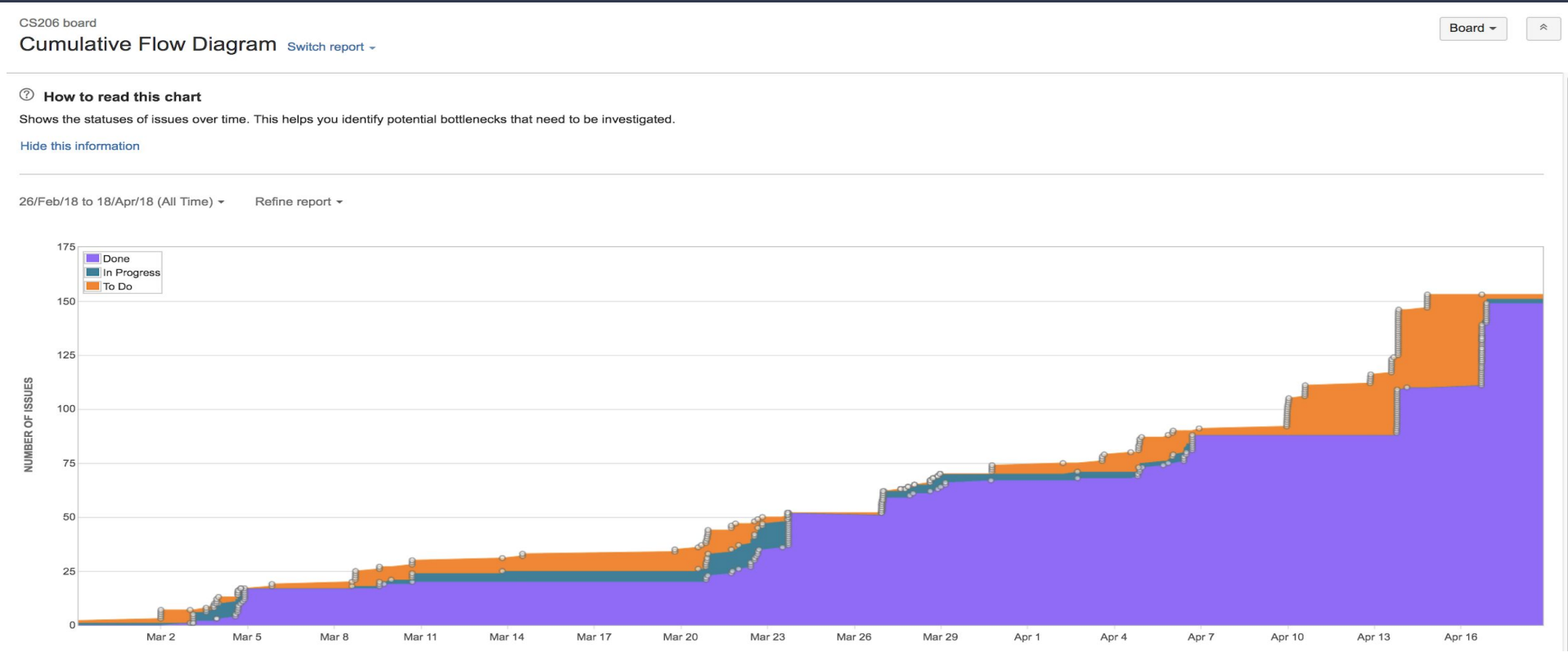
Threshold used: 20

File 1	File 2	Average Score	View Detail
package1/File1.py	project2/package2/package3/File1.py	100	
package1/File2.py	project2/package2/package3/File2.py	100	
package1/File1.py	project2/package2/package3/File2.py	20.47	
package1/File2.py	project2/package2/package3/File1.py	20.47	
package1/File1.py	project2/dice.py	15.45	
package1/File2.py	project2/diceV2_dynamic.py	15.15	
package1/File2.py	project2/dice.py	13.08	
package1/File1.py	project2/diceV2_dynamic.py	10.05	

Job Quality: Intuitive UI



Job Quality: Team Performance



Process and Teamwork

Three sprints

- Discuss, identify and prioritise use cases
- Create stories in Jira and assign tasks to each member
- Define apis and interfaces
- Communicate in person and via WhatsApp
- Integrate the project at the end of Sprint

Automatic deployment

- Jenkins builds on every change
- Code quality is checked on every pull request
- Automatic deploy on every successful pull request

Process and Teamwork

Two challenges

- Different time frames for every teammate
  - Program to interface
  - Communicate effectively

- Frequently changed requirements
  - Be prepared for changes
  - Be selective to features

Technology Transfer

Obstacles With Technology transfer

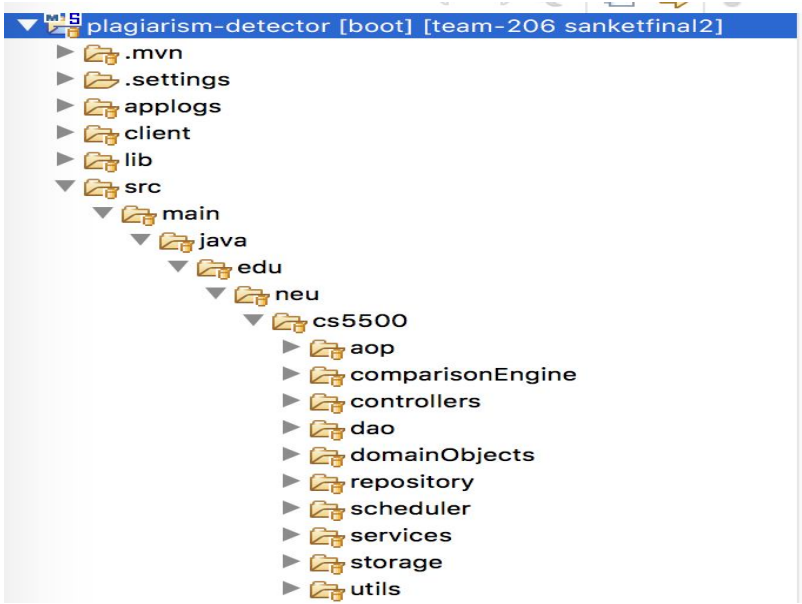
- Strict constraints of time
- Incomplete transfer of knowledge
- Lack of function recognition.
- Team members from a completed project are usually needed for the next project, and their new team leaders therefore must recruit them into new teams as soon as possible.

Solution:

- Well documented code so it becomes easier for the future team.

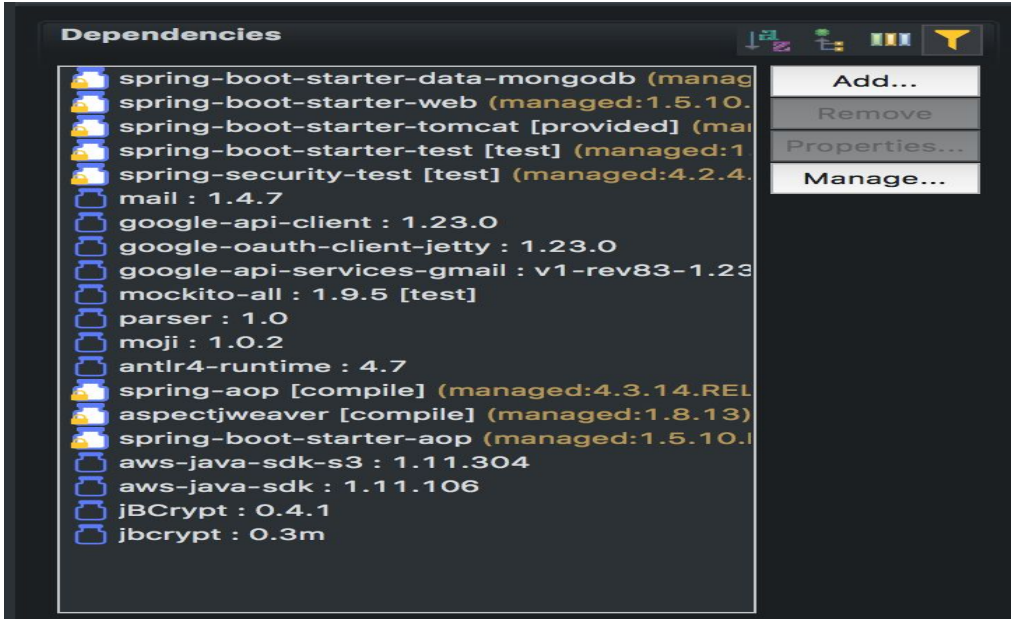
```java
import java.io.File;
/**
 * Context is a class that uses Strategy
 * @author JayanthGangadhar
 *
 */
public class Context {

	private Strategy strategy;
//Constructor
	public Context(Strategy strategy){
	   this.strategy = strategy;
	}
/**
 * execute Strategy demonstrates change in Context behavior based on strategy
 * @param f1 first file
 * @param f2 second file
 * @return similarity of the two files
 */
	public double executeStrategy(File f1, File f2){
	   return strategy.compare(f1, f2);
	}
	/**
	 * To return the name of strategy
	 * @return strategy name
	 */
	public String getStrategy() {
	    return strategy.getStrategy();
	}
    }
```

- Followed MVC architecture, Hence proper organization of code and good readability.



Scalability

- Implemented Aspect Oriented Programming (AOP) which serves to provide cleaner code.
- System is a spring boot application. Adding of a new service is simplified due to dependency injection.



Solution

- Followed strategy pattern and hence following open/closed principle.
- Makes it easy for future development to add add/update existing strategies.

```java
import java.io.File;
/**
 * Context is a class that uses Strategy
 * @author JayanthGangadhar
 *
 */
public class Context {

	private Strategy strategy;
//Constructor
	public Context(Strategy strategy){
	   this.strategy = strategy;
	}
/**
 * execute Strategy demonstrates change in Context behavior based on strategy
 * @param f1 first file
 * @param f2 second file
 * @return similarity of the two files
 */
	public double executeStrategy(File f1, File f2){
	   return strategy.compare(f1, f2);
	}
	/**
	 * To return the name of strategy
	 * @return strategy name
	 */
	public String getStrategy() {
	    return strategy.getStrategy();
	}
    }
```

```java
package edu.neu.cs5500.comparisonEngine.strategy;
import edu.neu.cs5500.comparisonEngine.ast.*;
/**
 * Strategy1 corresponds to the Zhang Sasha Algorithm used to generate a similarity between ASTs
 * @author JayanthGangadhar
 *
 */
public class Strategy1 implements Strategy {
    //To store the name of the strategy
    String strategy;
    //To get similarity as per the strategy
    double metric;
    /**
     * implements compare methods to compare similarity of two files
     */

    // Logger
    private static Logger logger = Logger.getLogger(Strategy1.class.getName());

    public double compare(File f1, File f2) {
        //represents the similarity of the two files in term of '%'
        metric=-1;
        //To Parse the first file
        ParserFacade parserFacade = new ParserFacade();
        AstPrinter astPrinter = new AstPrinter();
        //To Parse the second file
        ParserFacade parserFacade2 = new ParserFacade();
        AstPrinter astPrinter2 = new AstPrinter();
```

Future Work/ Recommendation

- Application can be trained using some of the best currently available tools such as MOSS.
- Implement session management to better serve the user.
- Implement a single university sign on.
- Update system to compare submissions across different courses and also different semesters.
- Add more strategies to detect plagiarism.