Group 2 Plagiarism Detector

Phase B: System Design

Authors: Shagun Bhardwaj
Kenji Fujita
Mason Leon
Prakash Tarun Kumar

Course: CS 5500 - Section 1
Foundations of Software Engineering
Dr. Frank Tip
Fall 2019

Khoury College of Computer Sciences
Northeastern University, Boston, MA

Phase B Deliverables:

- User Interface
  - Login Screen
  - Upload Screen
  - Results Screen
- System Architecture Diagram
- UML Diagrams
  - Deployment Diagram
  - Use Case Diagram
  - Sequence Diagram
  - Class Diagram (for Model)
- Java Interfaces for the components and major data structures

---

Abstract

Background
Plagiarism is a violation of academic integrity and often limits both the personal and collective success of students in the academic environment. In order to reduce the impact of plagiarism within the computer science program at Northeastern University, the department's faculty has reached out for assistance in acquiring a software tools that can detect plagiarism in student submitted programming assignments.

Language Choices

The plagiarism detector program will be written in Java and will support both C and Python. This is due to the fact that C and Python are popular core foundational programming languages taught to NEU students and therefore there is a high likelihood for potential plagiarism and abuse by students. Additionally, all three languages share similarities in that both Python and Java were written and heavily influenced by the C language.

Algorithms

The program will support both basic transformations such as a simple lexicographical comparison between two single file programs as well as complex transformations and attempts to hide plagiarism within multi-file projects. This will require use of and design of algorithms potentially falling in the following categories: substring matching, fingerprinting, “bag of words analysis”, stylometry, parse trees, locally sensitive hashing, k means clustering, or code- sequence-matching.

Features

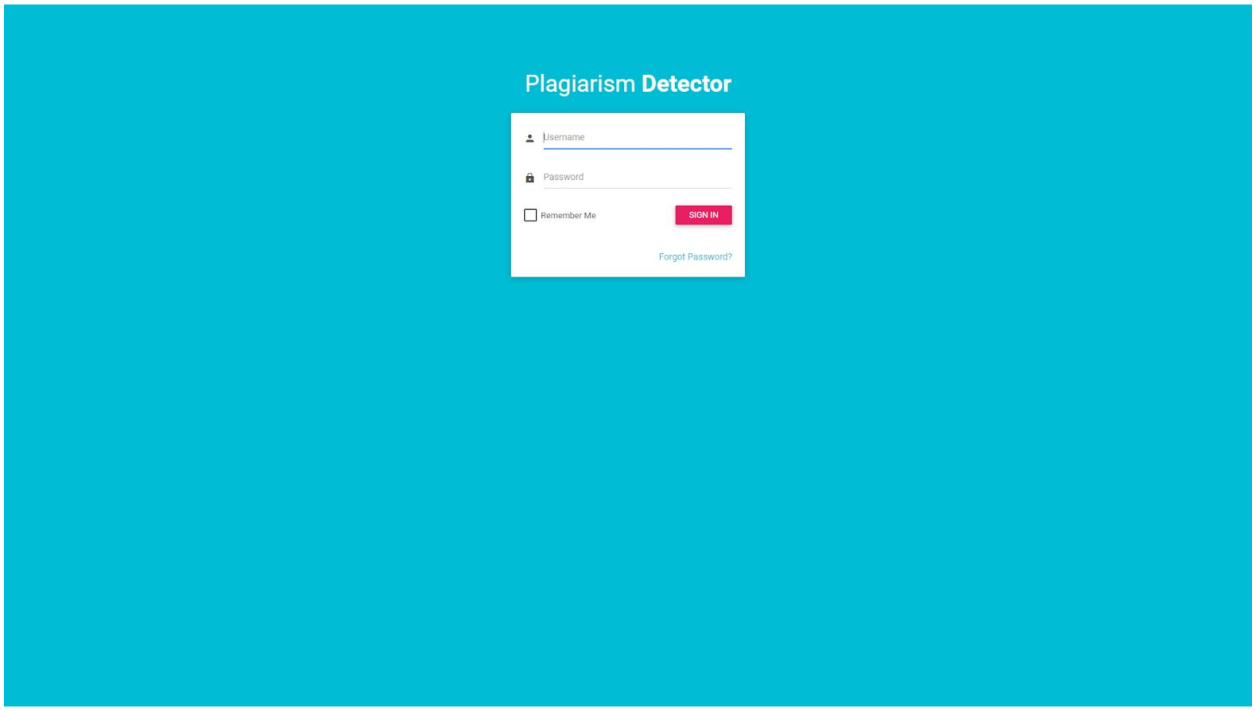
The program will incorporate a database to allow registered Professors, Instructors, or Teaching Assistants to keep track of their students and the results of plagiarism detector analysis. The database will also allow for storing of student email addresses and code files in order to ease further processing of the same code.

User Interface

The user will interact with the program using a web application.

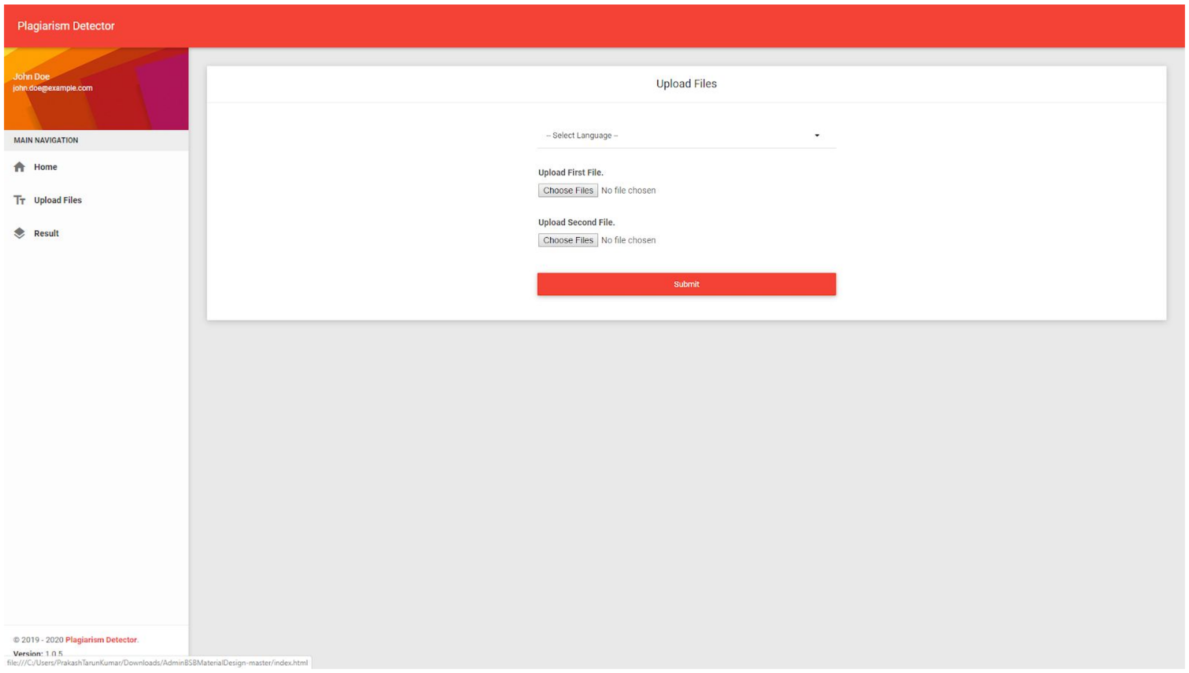
User Interface

Login screen



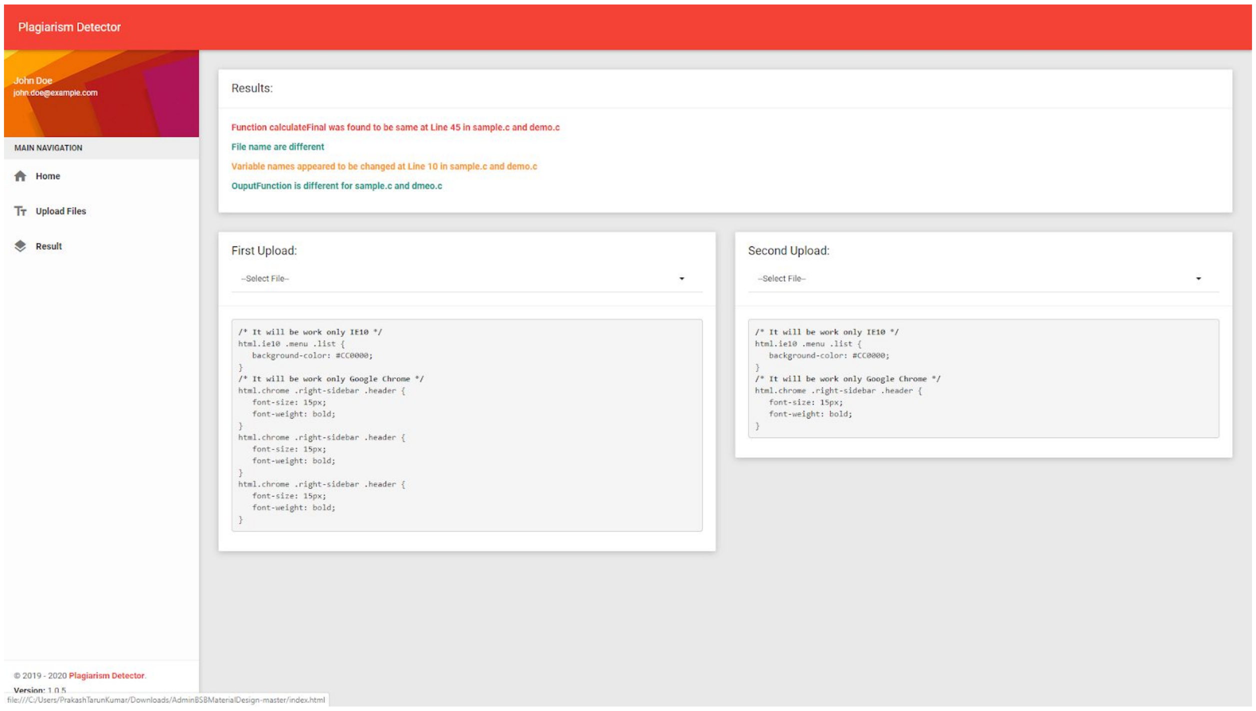
This is the login screen, where the professor (and in future releases professor designated Teaching Assistants and students) enter their credentials to sign in. The onClick() method of the sign in button will generate the Upload Files UI.

Upload Screen



This is the upload files UI, where the professor uploads the two separate project folders. The onClick() method for the submit button generates the results UI.

Results Screen



Function calculateFinal was found to be same at Line 45 in sample.c and demo.c
File name are different
Variable names appeared to be changed at Line 10 in sample.c and demo.c
OuputFunction is different for sample.c and dmeo.c

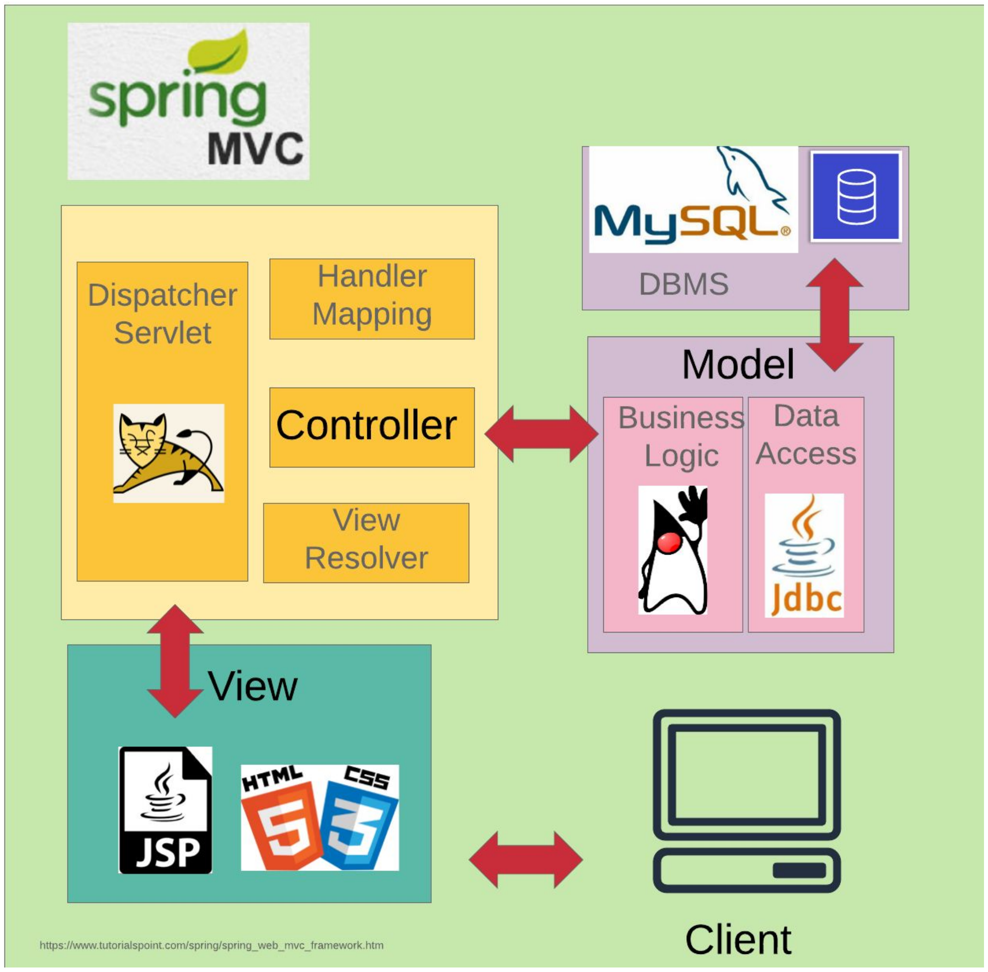
```
/* It will be work only IE10 */
html.ie10 .menu .list {
   background-color: #CC0000;
}
/* It will be work only Google Chrome */
html.chrome .right-sidebar .header {
   font-size: 15px;
   font-weight: bold;
}
html.chrome .right-sidebar .header {
   font-size: 15px;
   font-weight: bold;
}
html.chrome .right-sidebar .header {
   font-size: 15px;
   font-weight: bold;
}
```

```
/* It will be work only IE10 */
html.ie10 .menu .list {
   background-color: #CC0000;
}
/* It will be work only Google Chrome */
html.chrome .right-sidebar .header {
   font-size: 15px;
   font-weight: bold;
}
```

This screen will show the results to the professor. The two packages will be shown side by side, with any potential plagiarism flagged.

Major Components and Data Structures

System Architecture



Dependencies, Frameworks, and External Libraries:

SpringMVC is a Java-based application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, with extensions for building web applications on top of the Java EE (Enterprise Edition) platform. The SpringMVC flavor of the Spring platform features its own model–view–controller (MVC) web application framework.
https://spring.io/

Apache Tomcat is a web server for Java Web applications. It is an implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies.
https://tomcat.apache.org/

MySQL is an open-source relational database management system utilizing its own flavor of the Structured Query Language standard.
https://www.mysql.com/

Oracle JDBC is a Java Database Connectivity API providing universal data access from the Java for accessing relational databases from Java program.
https://www.oracle.com/database/technologies/appdev/jdbc.html

IntelliJ is a Java integrated development environment (IDE).
https://www.jetbrains.com/idea/

Maven is a build automation tool used primarily for Java projects allowing dynamic downloads of Java libraries and plug-ins from external repositories.
https://maven.apache.org/

ANTLR known as ANother Tool for Language Recognition is a parser generator for reading, processing, executing, and translating structured text or binary files. I supports import of programming language "grammar" for building and walking Abstract Syntax Tree parse trees.
https://www.antlr.org/

Major Classes

In order to allow for extensible enhancement of our application via future releases, we chose to center the design around the Model-View-Controller (MVC) architectural pattern. MVC supports application extensibility, simultaneous development, high cohesion, loose coupling, ease of modification, and allows the use of multiple views and controllers for a model.

Model
The Model is our application's primary data structure, responsible for processing inputted code files independent of the user interface. This module is responsible for classes and interfaces relating to the business logic, our lexer and parser algorithm engines, and storage and retrieval of data housed in a MySQL DBMS.

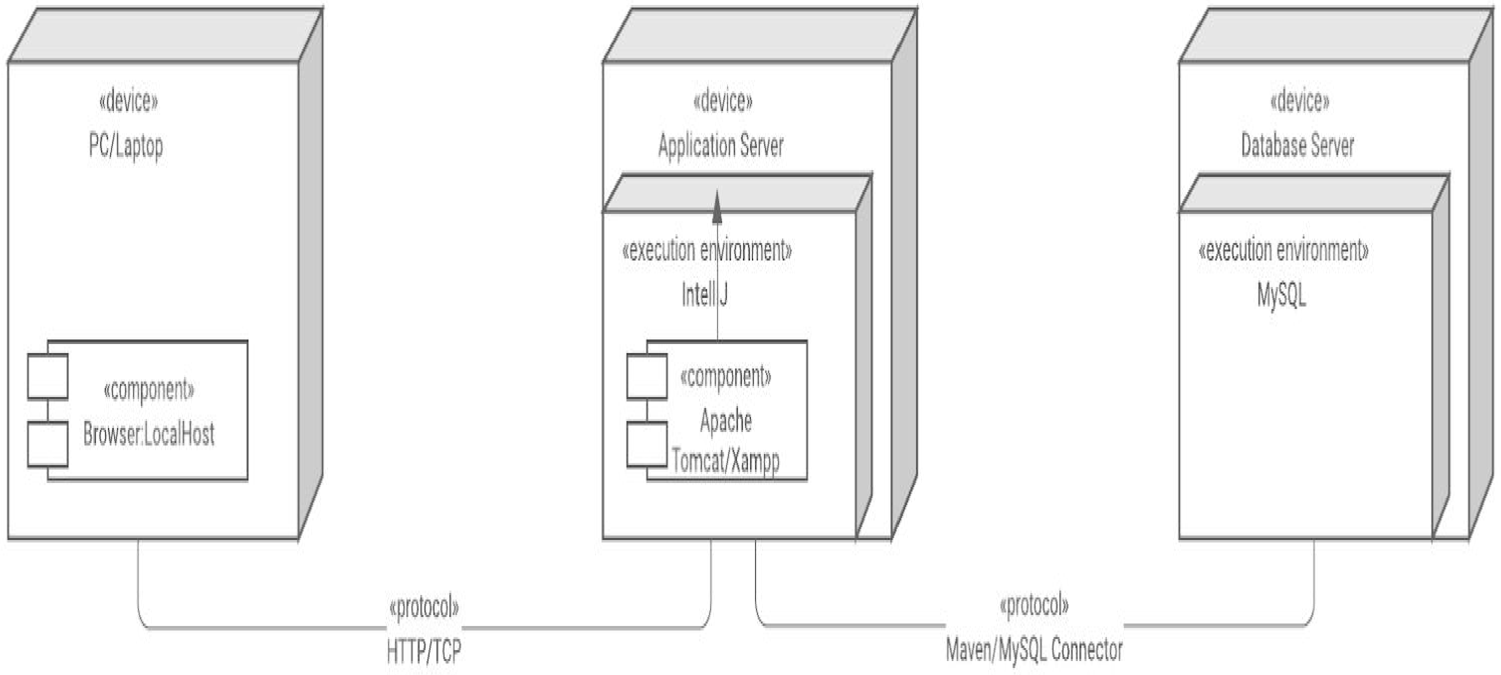
View
The view will provide a graphical way for users to interact with the system. It will be accessed via a web application. There will be buttons allowing upload of files and for the user to initialize code analysis. The view will also display both "diff" comparisons of

analyzed code as well as potentially other visualizations such as highlighting of similar parsed data structures or charts highlighting code analysis results.

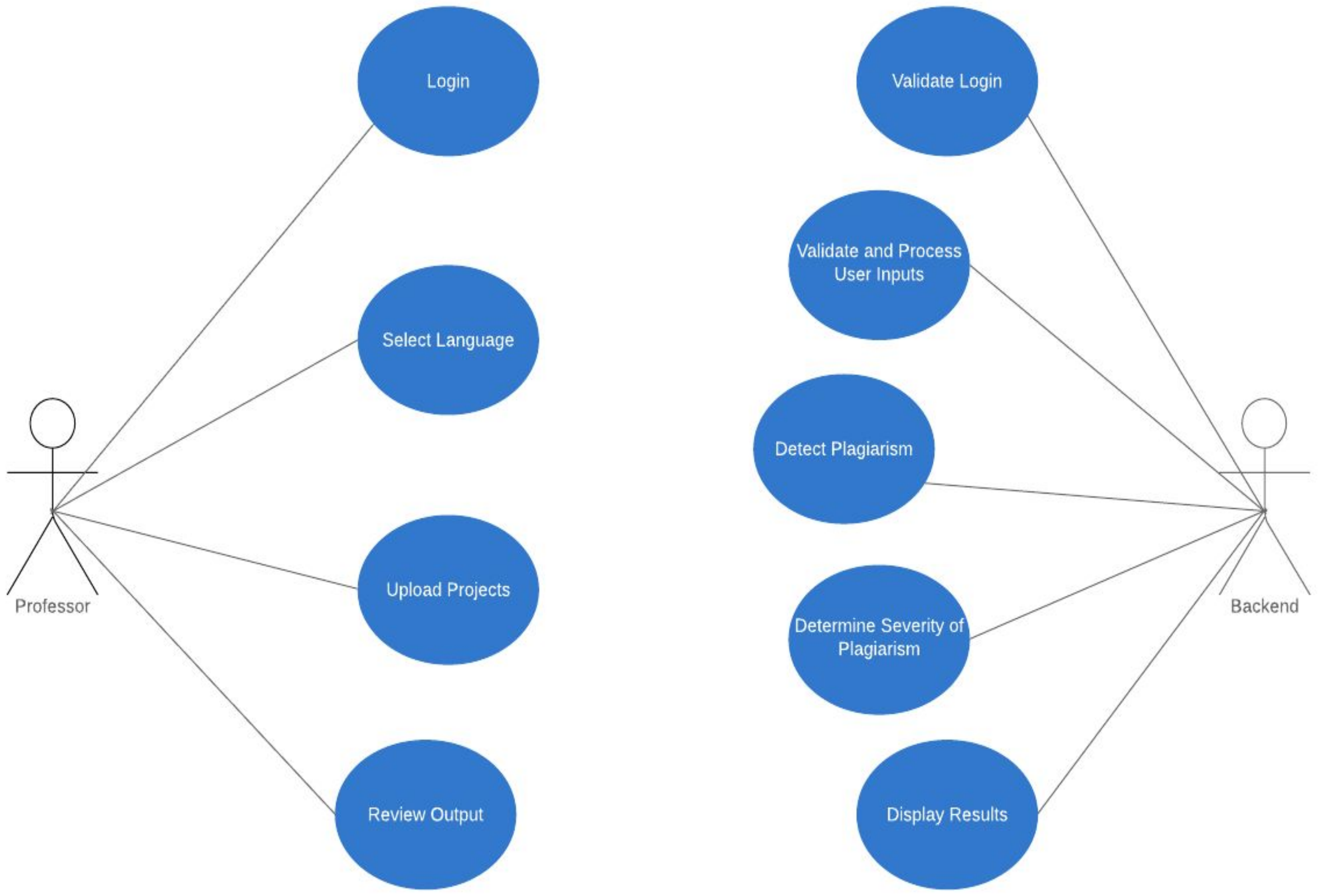
Controller
The Controller is the main driver for our program. This module is responsible for taking input (file packages) from the user, and giving it to the model to process. The controller will also take the result (where potential plagiarism is detected) from the model, and pass it back to the view to display to the professor.

UML Class Diagrams

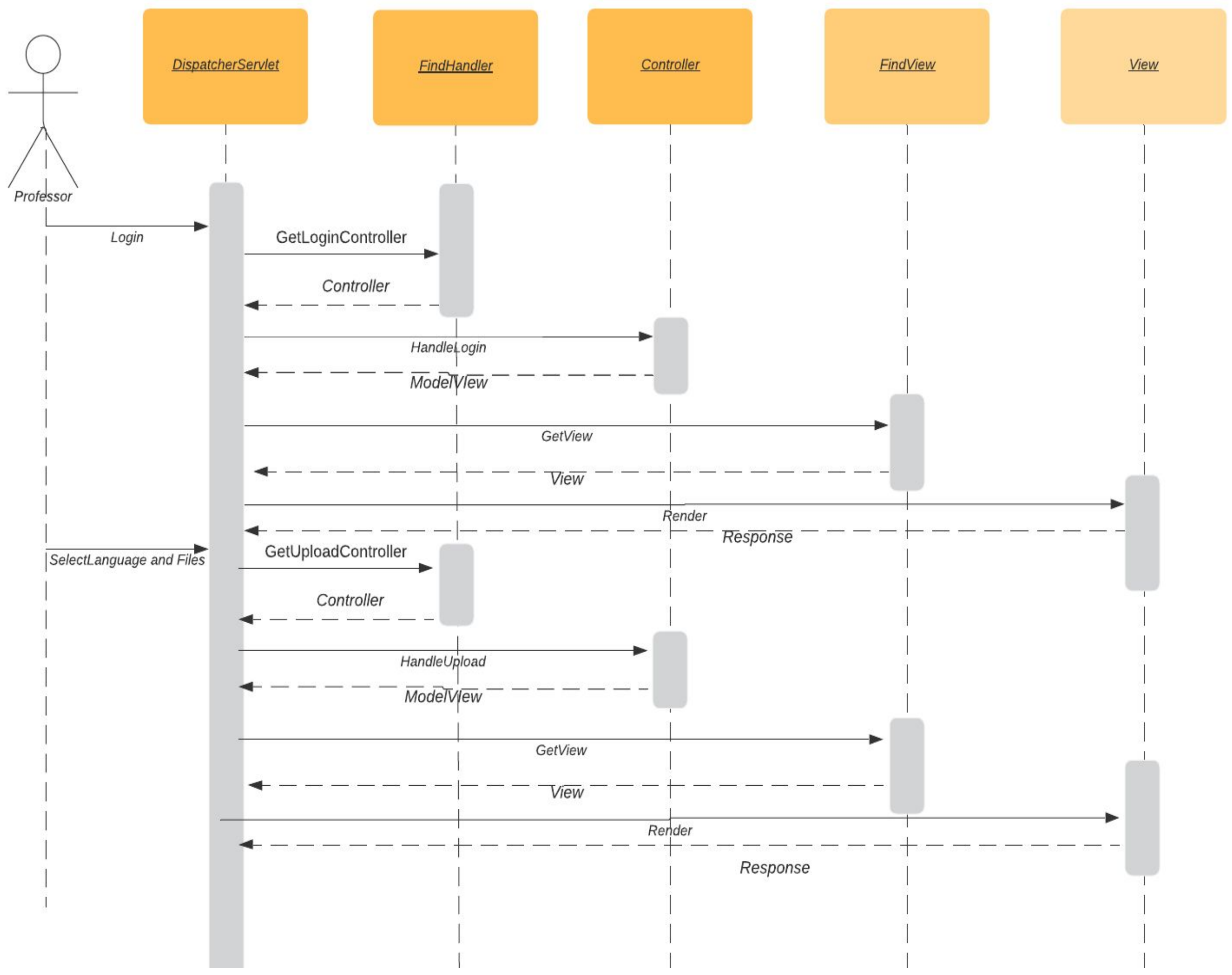
Deployment Diagram



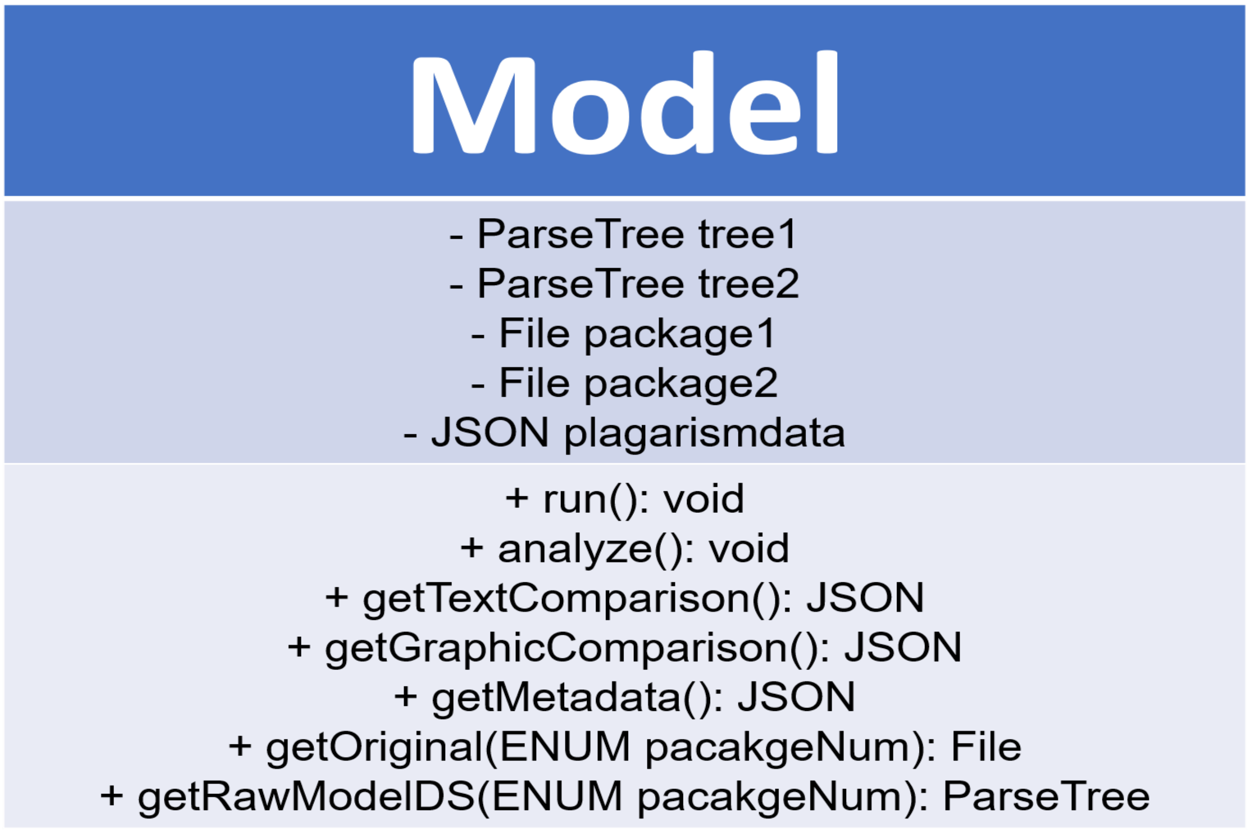
Use Case Diagram



Sequence Diagram



Model Class Diagram



Methods:

run: executes ANTRL on our 2 packages to get the 2 ASTs
analyze: runs our algorithm on the 2 ASTs to find potential plagiarism
getTextComparison: getter method for results
getGraphicComparison: getter method for results in a graphical form
getMetadata: getter method for the metadata generated from our algorithm
getOriginal: returns original code for 1 of the packages (depending on the parameter)
getRawModelDS: returns AST for 1 of the packages (depending on parameter)

Java Interfaces for Components and Major Data Structures

There will be multiple interfaces used in this project.

User interface will present any user (student or professor) that will use our program. This also leaves us room to add other users, such as TAs.

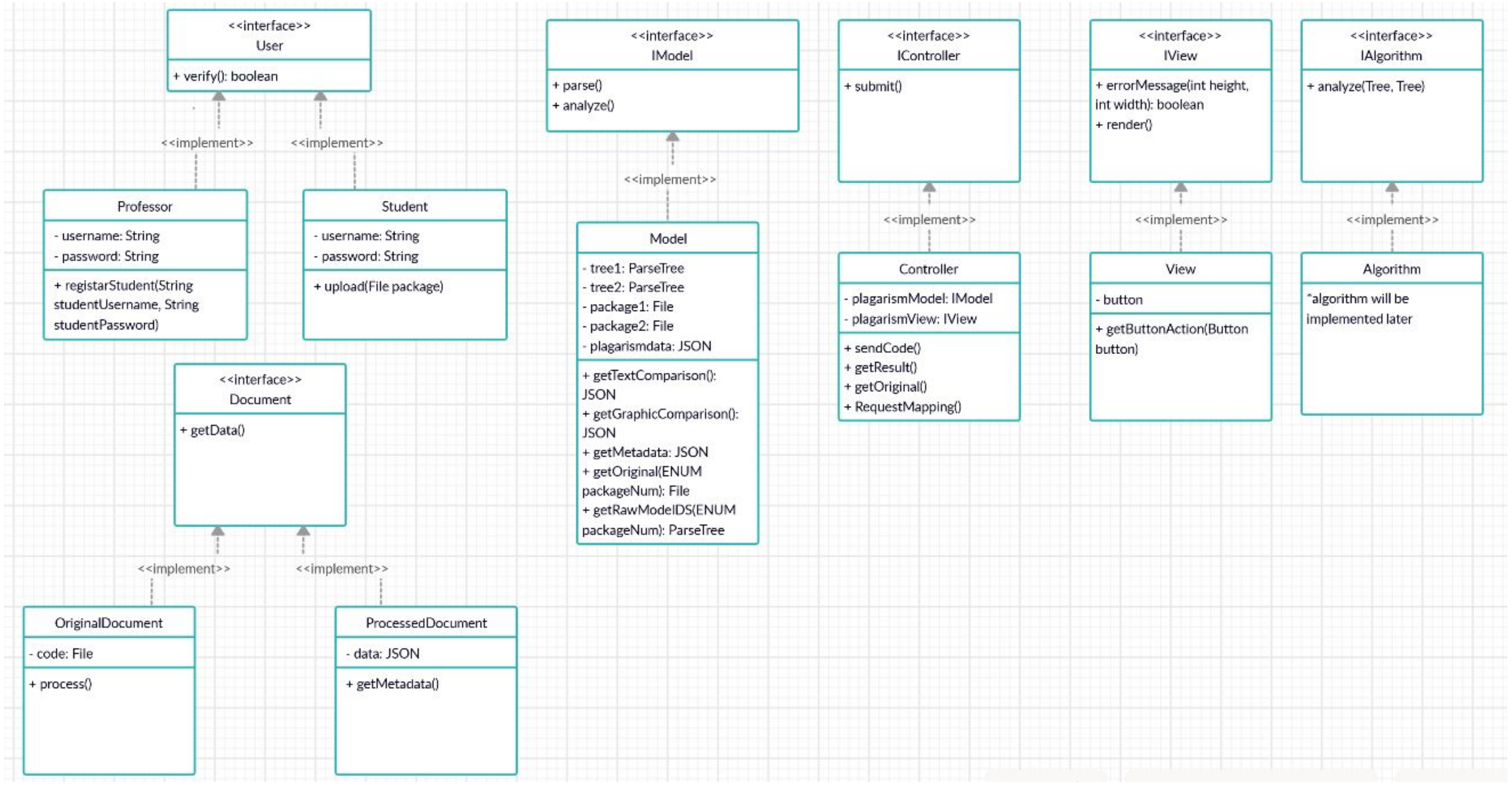
IModel is the interface that presents our Model. It will have 2 main operations: create the AST (using ANTLR) and analyzing the data for plagiarism.

IController will be the interface for our Controller. This will move data from the model to the view (and vice versa).

IView is the interface for our View. This will help design the web interface that will have actions such as upload and submit.

IAlgorithm is the interface for our algorithm. The main function of the algorithm is to analyze the two ASTs generated from ANTLR.

All of these interfaces allow us to be flexible to change each component without altering the other parts.


<<interface>>
User
+ verify(): boolean
<<implement>>
<<implement>>
Professor
- username: String
- password: String
+ registarStudent(String studentUsername, String studentPassword)
Student
- username: String
- password: String
+ upload(File package)
<<interface>>
Document
+ getData()
<<implement>>
<<implement>>
OriginalDocument
- code: File
+ process()
ProcessedDocument
- data: JSON
+ getMetadata()
<<interface>>
IModel
+ parse()
+ analyze()
<<implement>>
Model
- tree1: ParseTree
- tree2: ParseTree
- package1: File
- package2: File
- plagarismdata: JSON
+ getTextComparison(): JSON
+ getGraphicComparison(): JSON
+ getMetadata: JSON
+ getOriginal(ENUM packageNum): File
+ getRawModelDS(ENUM packageNum): ParseTree
<<interface>>
IController
+ submit()
<<implement>>
Controller
- plagarismModel: IModel
- plagarismView: IView
+ sendCode()
+ getResult()
+ getOriginal()
+ RequestMapping()
<<interface>>
IView
+ errorMessage(int height, int width): boolean
+ render()
<<implement>>
View
- button
+ getButtonAction(Button button)
<<interface>>
IAlgorithm
+ analyze(Tree, Tree)
<<implement>>
Algorithm
*algorithm will be implemented later

References

Albert, Guo. “Spring Mvc.” *LinkedIn SlideShare*, 6 June 2010, www.slideshare.net/junyuo/spring-mvc

Cornic, Pierre. *SOFTWARE PLAGIARISM DETECTION USING MODEL-DRIVEN SOFTWARE DEVELOPMENTIN ECLIPSE PLATFORM*. School of Computer Science, University of Manchester, 2008, https://studentnet.cs.manchester.ac.uk/resources/library/thesis_abstracts/MSc08/Abstracts/CornicPierre-fulltext.pdf.

Hamilton, James. *Static Source Code Analysis Tools and Their Application to TheDetection of Plagiarism in Java Programs*. Goldsmiths University of London, 12 June 2008, https://jameshamilton.eu/sites/default/files/msci-report.pdf.

Mariani, L. and Micucci, D. 2012. AuDeNTES: Automatic detection of teNtative plagiarism according to a rEference Solution. ACM Trans. Comput. Educ. 12, 1, Article 2 (March 2012), 26 pages. DOI = 10.1145/2133797.2133799 http://doi.acm.org/10.1145/2133797.2133799

Thomas Lancaster & Fintan Culwin (2004) A Comparison of Source Code Plagiarism Detection Engines, Computer Science Education, 14:2, 101-112, DOI: 10.1080/08993400412331363843

Wang, Y. Y., Shen, R. K., Chiou, G. J., Yang, C. Y., Shen, V. R. L., Putri, F. P. (2019), Novel Code Plagiarism Detection Based on
Abstract Syntax Tree and Fuzzy Petri Nets. International Journal of Engineering Education, 1(1), 46-56. doi:
http://dx.doi.org/10.12777/ijee.1.1.46-56]

V. Anjali, T.R. Swapna, Bharat Jayaraman, Plagiarism Detection for Java Programs without Source Codes, Procedia Computer Science, Volume 46, 2015, Pages 749-758, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2015.02.143.
(http://www.sciencedirect.com/science/article/pii/S1877050915002070)

“Spring Framework Web Integration Framework.” *Spring Tutorial*, Tutorials Point, https://www.tutorialspoint.com/spring/index.htm.

“Spring MVC Tutorial.” *Spring Tutorial*, Java T Point
https://www.javatpoint.com/spring-mvc-tutorial

“Web MVC framework.” *Serving Web Content with Spring MVC*, Spring by Pivotal.
https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html