

Plagus – Plagiarism Detector For Source Code

CS5500 – Group 6

Hassan Khan
Samuel Raphael
Yichuan Philip Ma
Pierre-Alexandre Mousset

Overview of Plagus

- Detect plagiarism in Python and TypeScript
 - Renaming variables
 - Expression Modification
 - Function Composition
- Web-application
 - Angular Front-End
 - Java Back-End
- Presentation Plan:
 - UI Mockups
 - System Architecture
 - UML Design
 - Algorithm Theory



User-Interface : Home Page

File upload

- The first step for the user is to drag and drop or upload two set of files
- An error message will appear if :
 - Less than 2 sets of files have been submitted
 - Files are too large

Programming languages

- The user will have to select on one of the two programming languages supported by our software

Run the detector

- The final step is to run the detector
- A loading icon will pop-up by using (Ajax)

The screenshot shows the PLAGUS web application interface. At the top, there is a navigation bar with 'PLAGUS', 'HOME', and 'CONTACT' links. Below the navigation bar, a welcome message reads: 'Welcome to Plagus! Please upload two sets of files in either Python or TypeScript and then we will check for plagiarism!'. The main content area is divided into three steps, each with a progress indicator (a circle with a dot inside) and a title.

Step 1/3 : Upload Your 2 Files

This step shows two file upload areas. Each area has a circular icon with a document symbol, labeled 'Files Set A' and 'Files Set B' respectively. Below each icon is a blue button labeled 'UPLOAD'.

Step 2/3 : Select The Language

This step shows two language selection options. Each option has a square icon with a language logo, labeled 'PYTHON' and 'TYPESCRIPT' respectively. Below each icon is a blue button labeled 'RUN DETECTOR'.

Step 3/3 : Get Your Plagiarism Score

This step shows a single progress indicator and a blue button labeled 'RUN DETECTOR'.

Arrows from the text blocks on the left point to the corresponding steps in the interface: from 'File upload' to Step 1, from 'Programming languages' to Step 2, and from 'Run the detector' to Step 3.

User-Interface : Home Page

Navigation

- Student first and last name
- List of the files submitted (current file displayed has a different font-color)
- The user won't be able to go to the next step if the system displays error messages

Detection

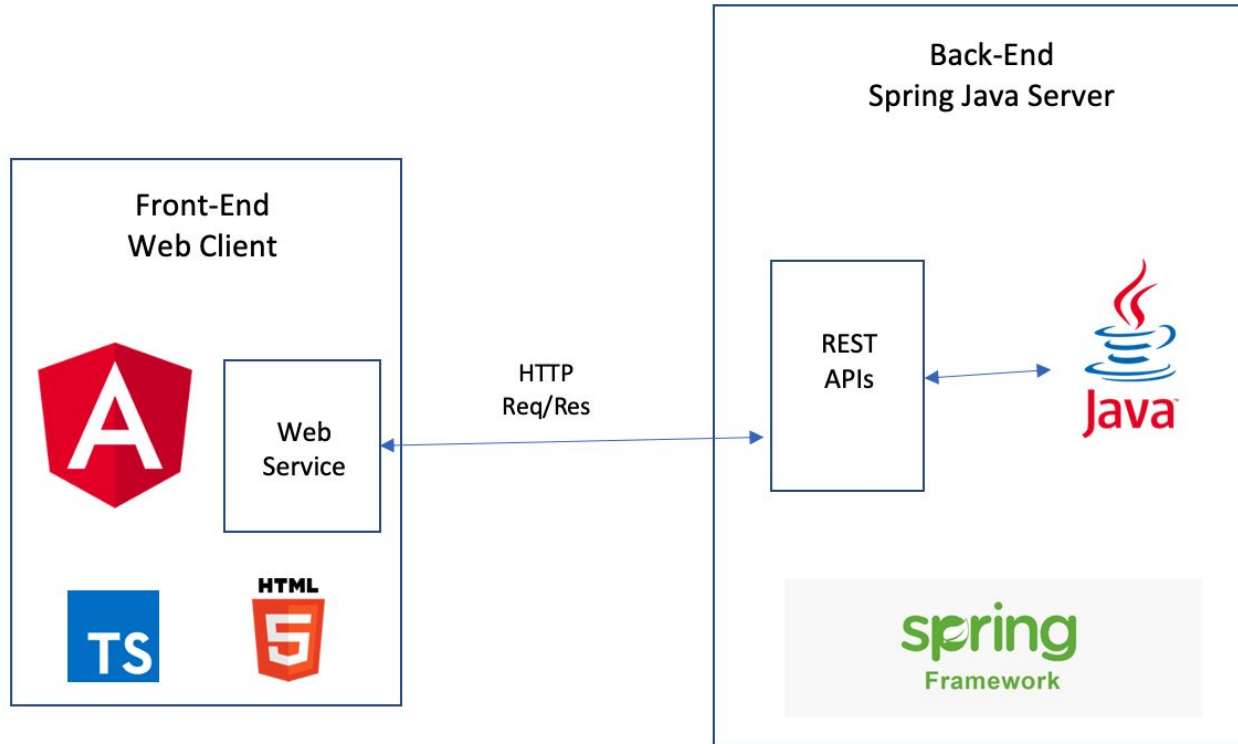
- Code that is likely to be plagiarism is highlighted
- A global plagiarism score is displayed
- The user can navigate between the different set detected by our detector

Report

- At the end of the review, the professor can either report the student to the school administration or not

The screenshot displays the 'PLAGUS' application interface. At the top, there are navigation links: 'HOME' and 'CONTACT'. Below this, two user profiles are shown side-by-side. The left profile is for 'John Doe' and the right for 'Lucia Doe'. Each profile lists submitted files: John has 'main.py', 'user.py', and 'test.py'; Lucia has 'craeo.py' and 'main.py'. A blue box highlights a code snippet for John's 'main.py', showing Python code for a web scraper. Below the code, a large orange circle displays a global plagiarism score of '54%'. Navigation buttons labeled 'PREV' and 'NEXT' are positioned on either side of the score. At the bottom, there are two buttons: a red 'report submissions' button with a minus sign icon, and a green 'validate submission' button with a checkmark icon. Blue arrows from the 'Navigation' and 'Detection' sections point to the user profiles and the code snippet, respectively. A blue arrow from the 'Report' section points to the 'report submissions' button.

System Architecture



Data Structure - File Integrity Over the Web?

```
{
  submission_1: [
    {
      id: 1,
      Type: 'Python',
      Name: 'test1.py',
      code : 'def main(): \n for i in the range (5) : \n print(i) \n main()'
    },
    {
      id: 2,
      type: 'Python',
      name: 'test2.py',
      code : 'def main(): \n for i in the range (10) : \n print(i) \n main()'
    }
  ]
  submission_2: [
    {
      id: 1,
      Type: 'Python',
      Name: 'test3.py',
      code : 'def main(): \n for i in the range (15) : \n print(i) \n main()'
    },
    {
      id: 2,
      type: 'Python',
      name: 'test4.py',
      code : 'def main(): \n for i in the range (10) : \n print(i) \n main()'
    }
  ]
}
```

```
def main():
    for i in range(5):
        print(i)
```

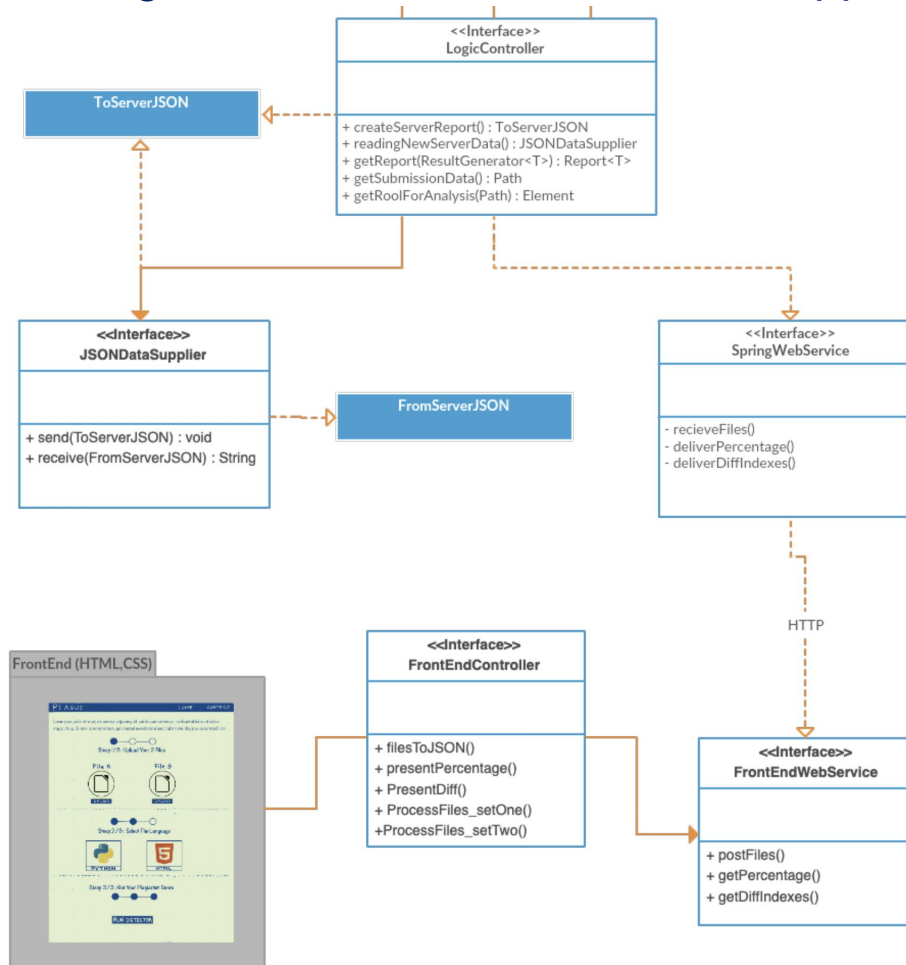
main()

Solution: Base64 Encode/Decode

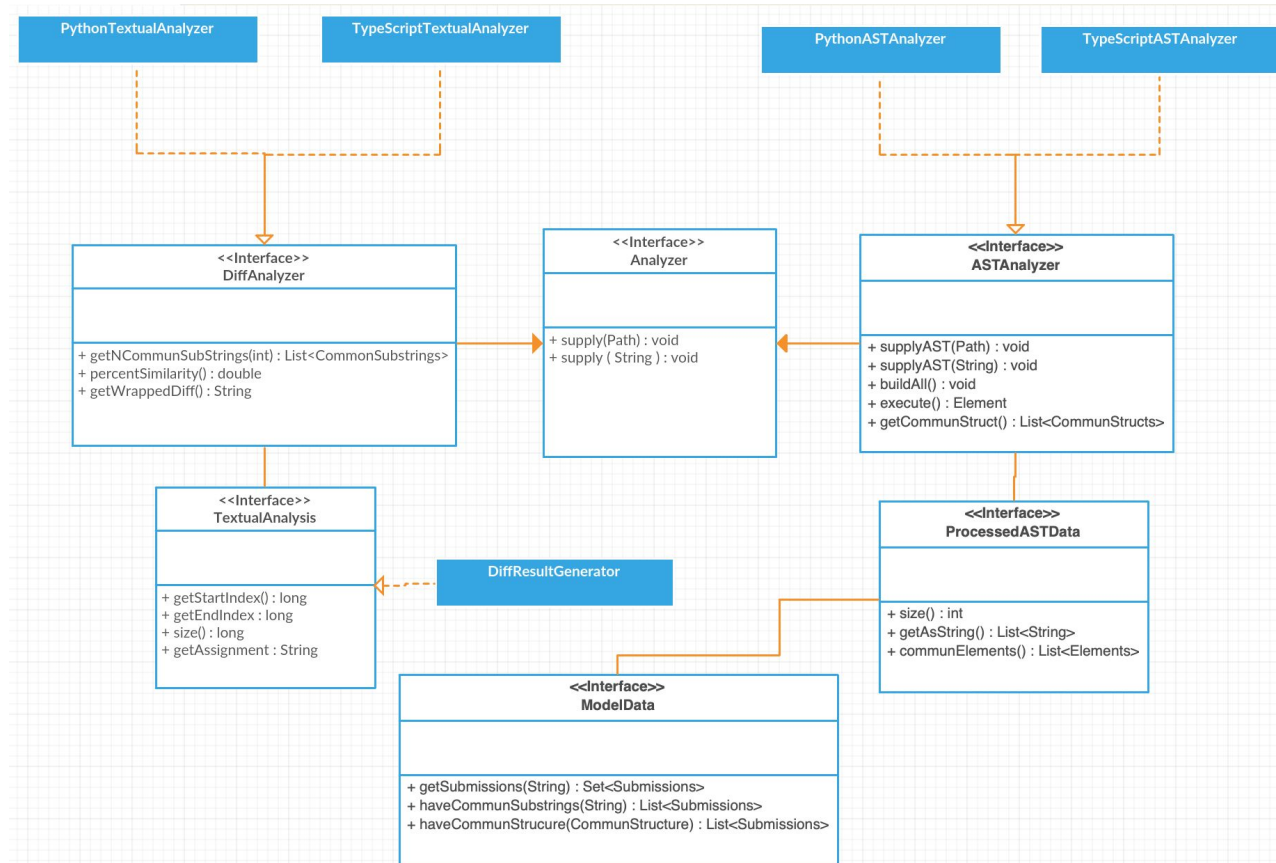
TS Front-End => atob(FileToDecode)

Java BackEnd => Base64.getDecoder().decode(encodedFile)

Class Diagram of Plagus-Communication with Web App



Class Diagram of Plagus-Plagiarism Detection Engine



Class Diagram of Plagus-AST Elements

Example:

<expression>=<literal>|<expression><op><expression>|

<var-exp>

<literal>={“abc”, ‘z’, 123, 3.14, true, false, etc...}

<var-exp>={x, y, foo, bar, etc...}

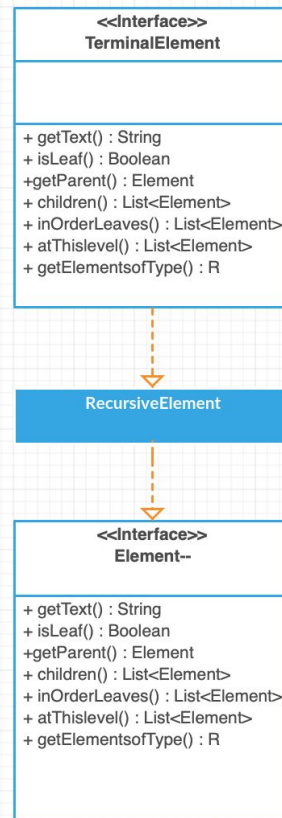
<op>=+|-|*|/|%|&|&[bitwise OR]

<statement>=var <var-exp>|

<var-exp>[assignment =]<expression>|

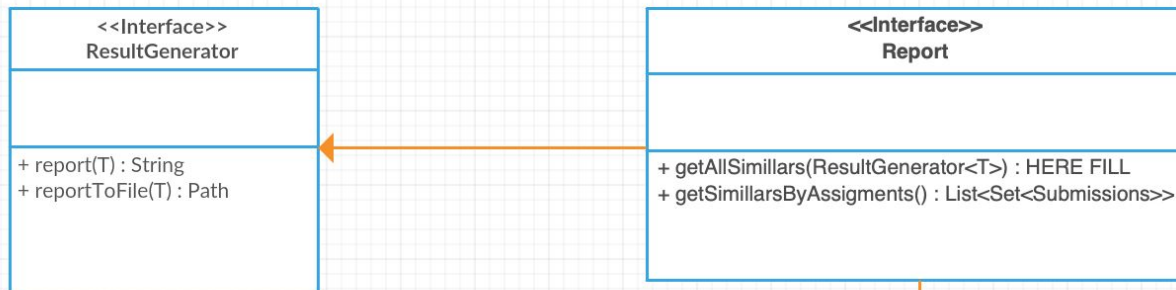
<statement>;<statement>|print(<expression>)

We shall use ANTLR for generating and analyzing ASTs in source code.

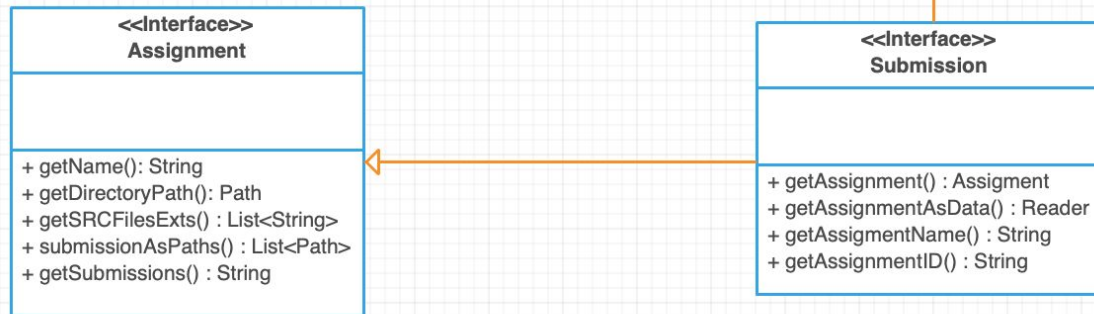


Class Diagram of Plagus-Assignments and Reports

Results



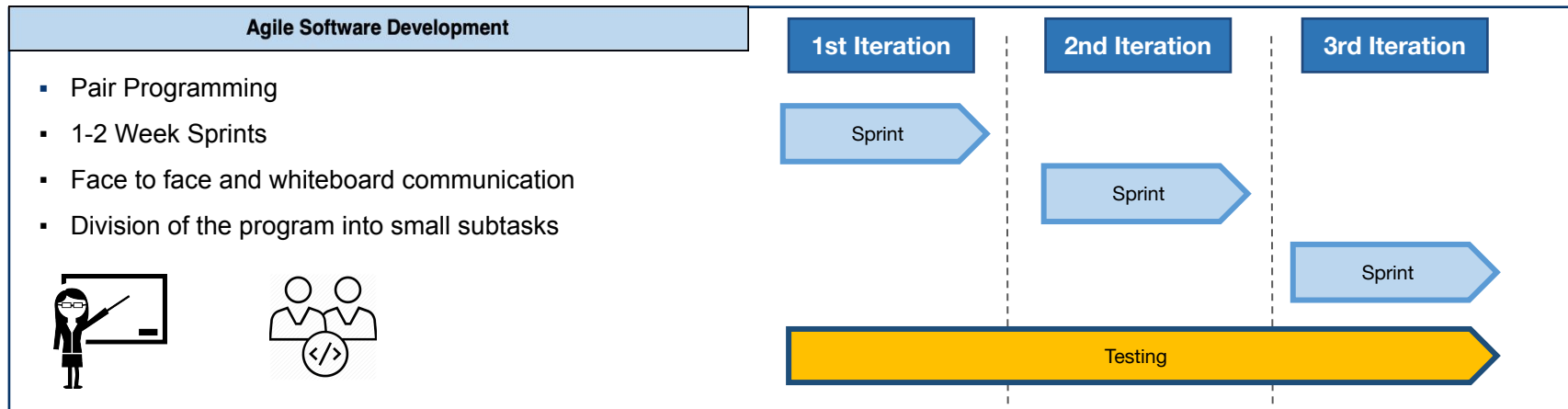
Organization:



Plagiarism Algorithm

- High-Level Algorithm Goals
 - Provide at least one strategy for detecting similarity between at least two sets (src folders) of code.
 - Use enough abstraction that via ASTs that once the language has been processed, there should be no difference between the analysis of ASTs on two different languages.
- Preliminary Ideas about how to achieve these goals
 - AST construction via ANTLR.
 - Going beyond CFG: Mapping generated ASTs to specialized versions
 - Scope and Name analysis for functions, variables classes etc. across files.
 - Tracing Objects, Imports across files.
- Where to start
 - Look into the TypeScript grammar quality.
 - 3rd party?
 - Fallback: Vanilla JavaScript or Ruby
 - Learn ANTLR
 - what is realistic to expect from our tools,
 - What to do with expectations.
 - Adapt expectations and research to shape our strategies.

Software Engineering practices



Work Distribution

	UX/UI	Architecture	Algorithm	Testing	Documentation
Samuel		X	X		X
Hassan	X	X	X		X
Yichuan			X	X	X
Pierre	X		X	X	X

Thank You from Plagus!

Any questions?

Hassan Khan
Samuel Raphael
Yichuan Philip Ma
Pierre-Alexandre Mousset
