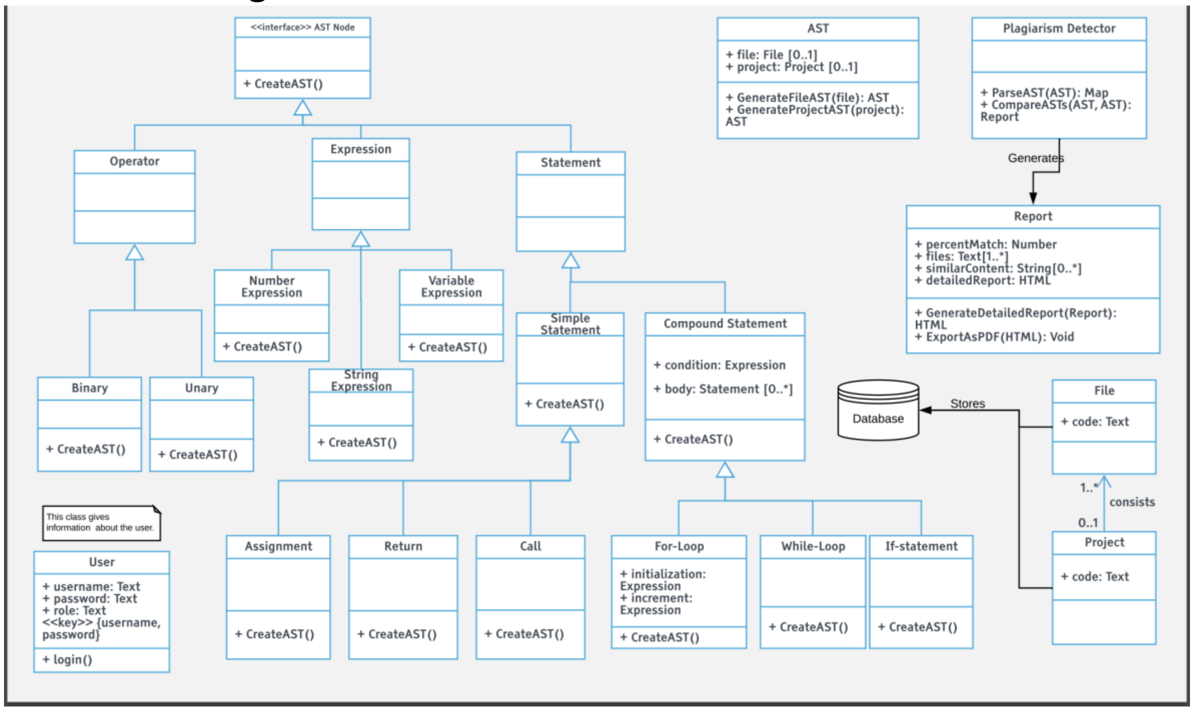
Team 214

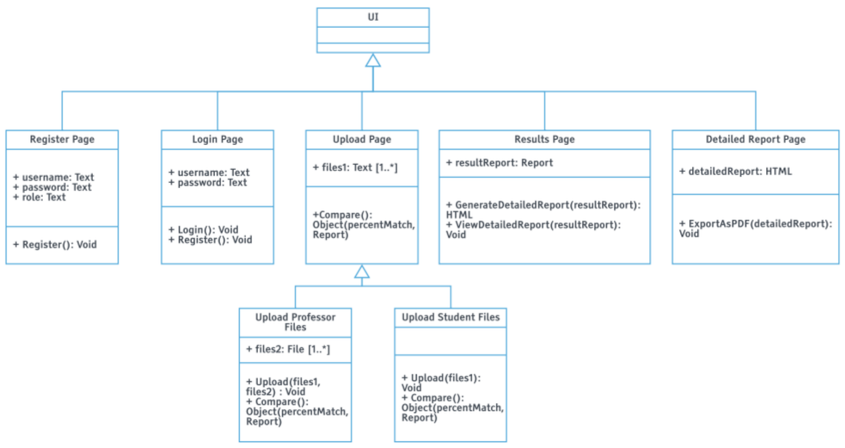
Aditya Batheja
Mihir Gandhi
Tanvi Ranadive
Deepak Surana

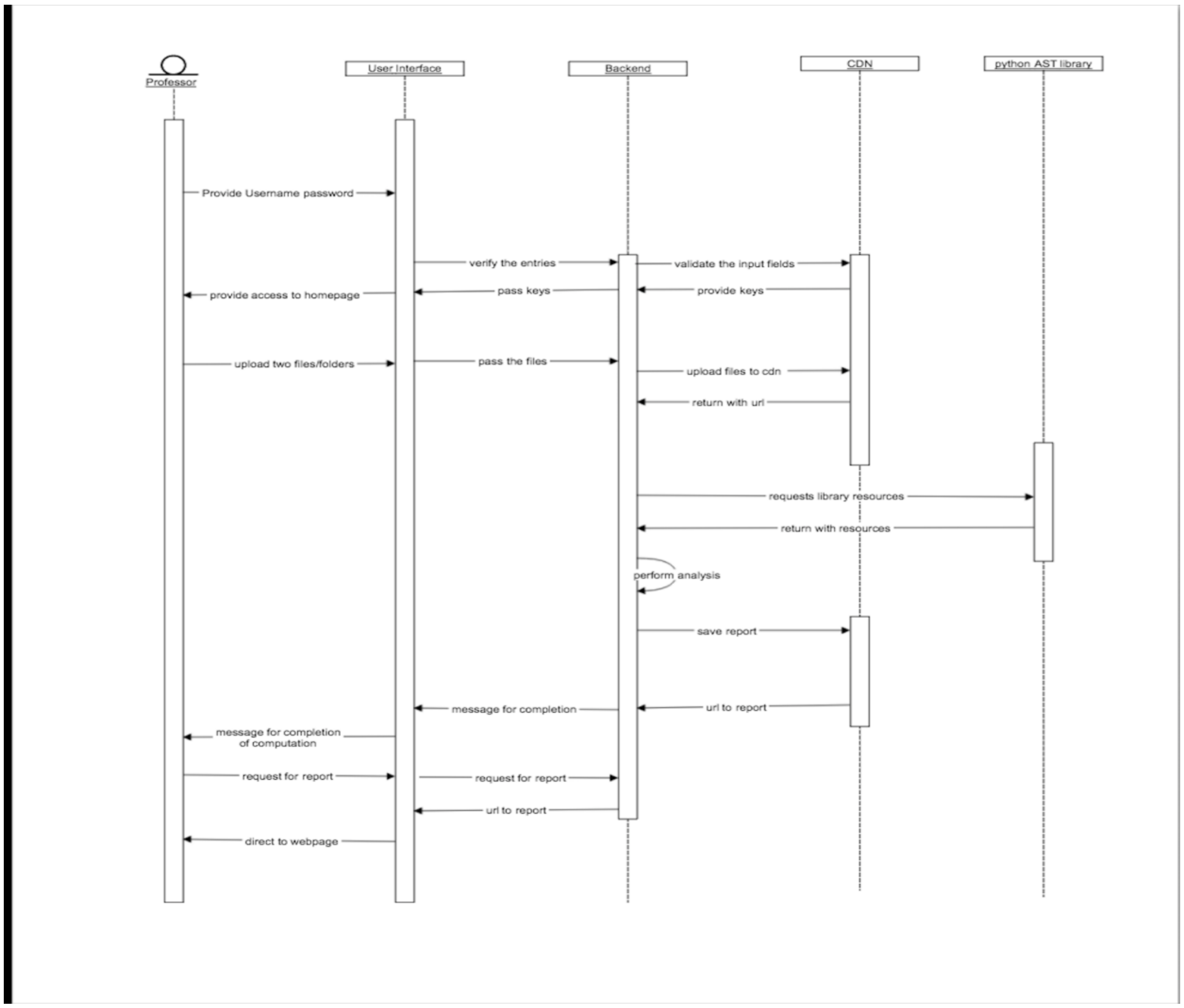
Phase B Requirements Submission

This document gives a high-level view of the architecture of the “Plagiarism Detector” with the help of UML Class Diagrams and important java classes, Sequence diagrams and data structures. The document will also explain the design patterns used and the process of working of the system.

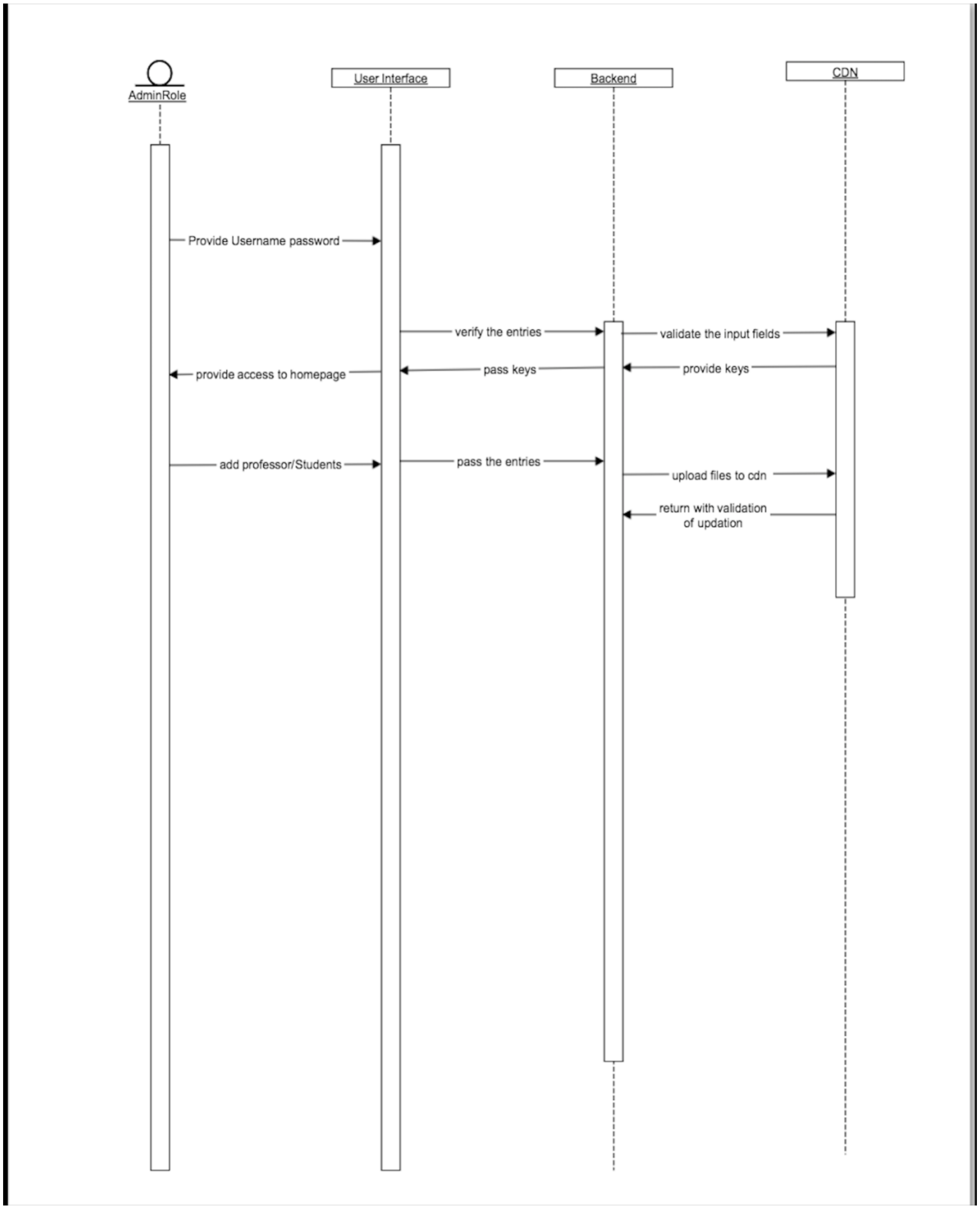
UML Class Diagram



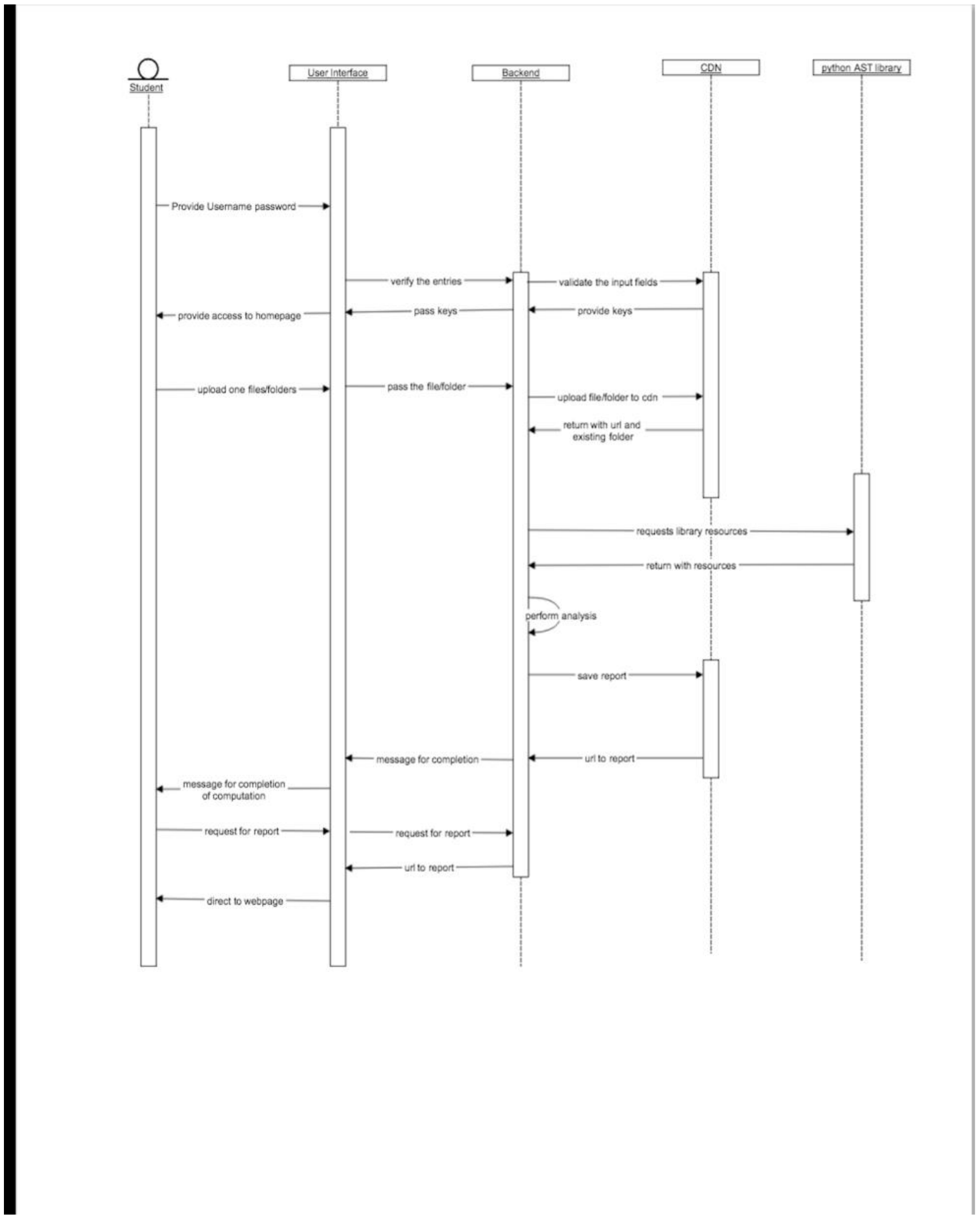

UI
Register Page
+ username: Text
+ password: Text
+ role: Text
+ Register(): Void
Login Page
+ username: Text
+ password: Text
+ Login(): Void
+ Register(): Void
Upload Page
+ files1: Text [1..*]
+Compare(): Object(percentMatch, Report)
Results Page
+ resultReport: Report
+ GenerateDetailedReport(resultReport): HTML
+ ViewDetailedReport(resultReport): Void
Detailed Report Page
+ detailedReport: HTML
+ ExportAsPDF(detailedReport): Void
Upload Professor Files
+ files2: File [1..*]
+ Upload(files1, files2) : Void
+ Compare(): Object(percentMatch, Report)
Upload Student Files
+ Upload(files1): Void
+ Compare(): Object(percentMatch, Report)



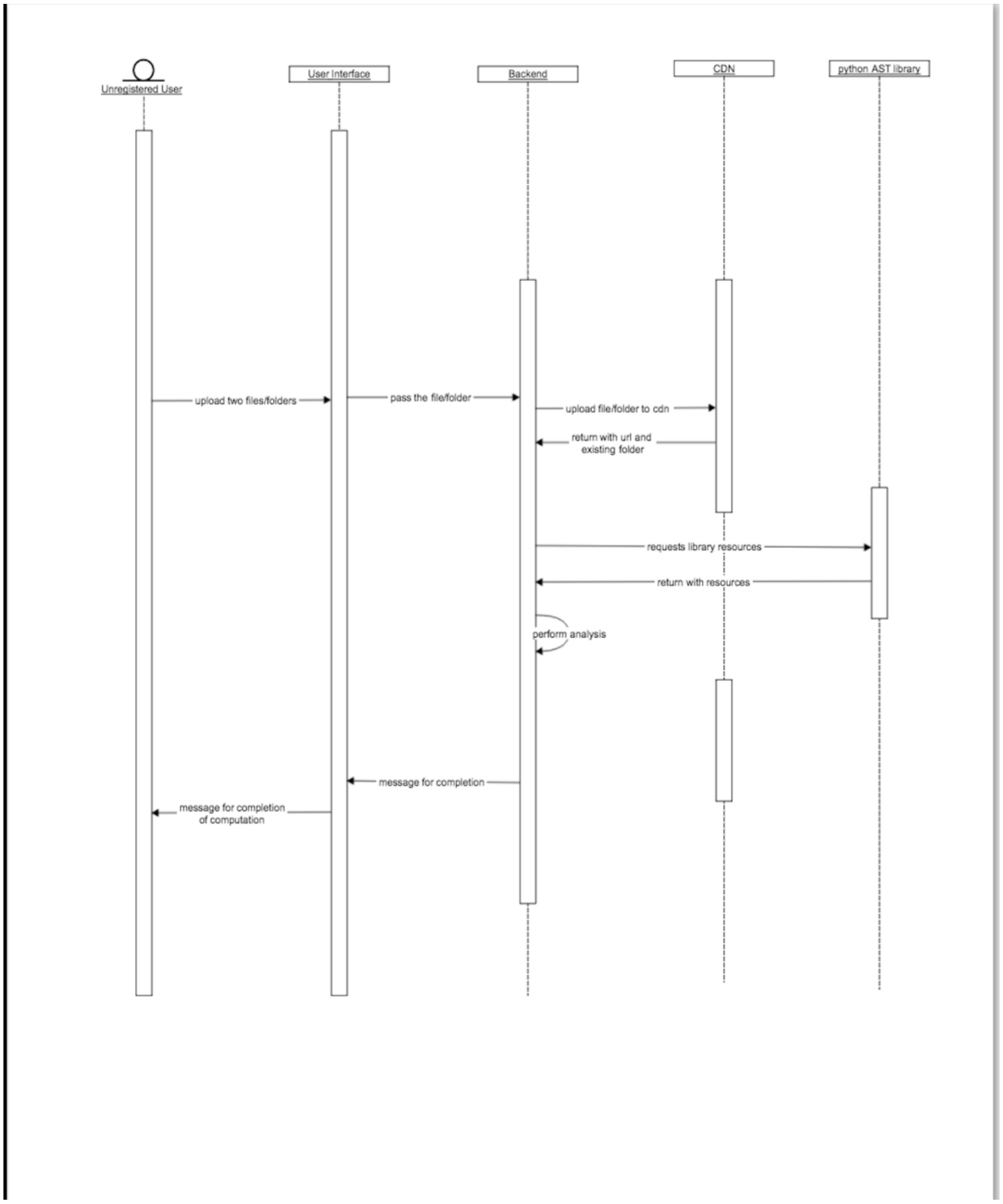
Sequence Diagram with Professor interacting with the system



Sequence Diagram for the AdminRole Interacting with the system



Sequence Diagram for the Student Interacting with the System



Sequence Diagram for a non-registered user interacting with the system

Introduction

The object of the project is to build a plagiarism checker that checks for the similarity between two input files/folders. In this document we briefly discuss what we intend to do in the system and the design of it.
We present a design that explains the working of the system with its' key data structures.

Design Decision

We are considering using design patterns to implement our system.
We decided on using the singleton pattern to make sure that only one instance of the connection to the database is created.

In software engineering, the singleton pattern is a software design pattern that restricts the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system. The concept is sometimes generalized to systems that operate more efficiently when only one object exists, or that restrict the instantiation to a certain number of objects. The term comes from the mathematical concept of a singleton. (reference: Wikipedia)

We are also considering using other design patterns for making the implementation of our system more fluid.

We are also going to compare the two ASTs and therefore to separate the visited from the traversal logic in our code we are considering using the Visitor pattern.

In object-oriented programming and software engineering, the visitor design pattern is a way of separating an algorithm from an object structure on which it operates. A practical result of this separation is the ability to add new operations to existent object structures without modifying the structures. It is one way to follow the open/closed principle. In essence, the visitor allows adding new virtual functions to a family of classes, without modifying the classes. Instead, a visitor class is created that implements all of the appropriate specializations of the virtual function. The visitor takes the instance reference as input and implements the goal through double dispatch. (reference :Wikipedia)

Apart from these we also want to implement several object-oriented design principle and principle of UI Development to make sure our front end is also as developed as our back end. We agree with Steve Jobs on the fact that design is not how it looks and feels but how it works.

Description

The Class diagram and the abstract classes give a representation of how the components are going to interact with each other.
1] ASTNode Interface: This Interface is the base interface. It contains a method called CreateAST.
2] Operator: It is an interface extending ASTNode Interface. They will deal with all operators in the node.
3] Expression: It is an interface extending ASTNode Interface. They will deal with all expressions in the node.
4] Statement: It is an interface extending ASTNode Interface. They will deal with all statements in the node.
5] Assignment: It is a concrete class that extends SimpleStatement to deal with simple statements in the code.
6] Binary: It is a class that implements the Operator Interface.
7] Call : It is a class that extends SimpleStatement Class. It has a CreateAST method.
8] Compund Statement: This class implements the Statement interface. It also consists of a CreateAST method.
9] File : This class handles that files that are uploaded by the user of the system.
10] ForLoop : This class extends the CompundStatement class. It consists of a CreateAST method.
11] IfStatement: This class extends the CompundStatement class. It consists of a CreateAST method.
12] NumberExpression: This class implements the Expression interface. It also consists of a createAST method.
13] PlagiarismDetector: This class has two methods. ParseAST parses the ast generated by a file and compareAST which returns the report generated from its' report method.
14] Report: This class has three methods. Report method generates the report. generateDetailedReport method generates a detailed report. exportAsPdf generates the pdf report.
15] Return: This class contains the CreateAST method. It extends simpleStatemetnt.
16] SimpleStatement: This class implements Statement. It contains CreateAST method.
17] StringExpression: This class implements the Expression interface. It also contains the CreateAST method.
18] Unary: This class implements the Operator interface. It also contains the createAST method.
19] VariableExpression : This class implements the expression interface. It also contains a createAST method that will create the AST representation of the variable expression.
20] WhileLoop : This class extends the CompoundStatement Class. It contains the createAST method which creates an AST representation of the while loop.

Future Plan

We plan to have more than one algorithm to detect plagiarism detection.

The choice of using one of many algorithms to detect plagiarism is a feature we would like to incorporate.

Algorithm

The general concept is to compare two ASTS. Then we want to get the hash values of nodes in the AST and then return list of nodes with same hash value to detect plagiarism.

References

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.186.4783&rep=rep1&type=pdf

https://arxiv.org/pdf/1704.08829.pdf

https://www.sciencedirect.com/science/article/pii/S0950705115004918

https://dl-acm-org.ezproxy.neu.edu/citation.cfm?doid=1150402.1150522&preflayout=flat

https://greentreesnakes.readthedocs.io/en/latest/

https://stackoverflow.com/questions/9309706/how-do-i-generate-a-control-flow-of-an-ast-represented-in-xml-using-python

http://grepcode.com/file/repo1.maven.org/maven2/org.python/jython/2.2.1/org/python/parser/ast/Compare.java

https://en.wikipedia.org/wiki/Comparison_of_parser_generators

https://tomassetti.me/parsing-any-language-in-java-in-5-minutes-using-antlr-for-example-python/

http://www.eclipse.org/articles/article.php?file=Article-JavaCodeManipulation_AST/index.html#sec-parsing-a-source-file

https://github.com/LouisJenkinsCS/DSL

https://github.com/cheburakshu/Javascript-Explorer-Callgraph