**Remote Server, general algorithms, and network communication**

**Remote Server:**

**FUNCTIONALITY 1:**

To receive a file for analysis over a network connection. The file will contain all the source code that is to be compared.

**FUNCTIONALITY 2:**

To determine the similarity of two samples of code:

* Sample 1 is the student’s code that is being examined.
* Sample 2 is everything that is being tested against Sample 1, which includes other assignments from the same class, and possibly other source files that the instructor provides.

This is repeated for each student’s assignment.

It may be sensible to develop the analyzer in stages, starting with a simple comparator and then improving upon it. This will lead to an early working version and will likely also lead to a better final product. Stages could look something like the following:

* Stage 1. Detect identical code (straight comparison after removing whitespace characters, tokenizing, etc.).
* Stage 2: Detect reordering within subroutines.
* Stage 3: Detect more complicated changes such as breaking subroutines into smaller ones.

NOTE: Stages 2 and 3 above are not necessarily the approaches that will be taken, but rather are examples of how the analyzer could become more sophisticated.

**ALGORITHMS TO CONSIDER FOR ANALYZER:**

* MOSS method: <http://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf>
* Longest Common Sequence
* KMP
* Rabin-Karp algorithm

**FUNCTIONALITY 3:**

To assign a similarity score to each assignment to give an accurate representation of its similarity to other assignments.

This can either be presented as a percentage of suspect code (see example (1), below), or as a split between exact copies and suspect lines (see example (2), below).

1. “25% of this source is suspect.”
2. “15% of this source is as exact match, and an additional 10% is suspect.”

The reason to go with option (1) is that it removes a certain liability if the algorithm makes a mistake (which can also be overcome using a waiver). Moreover, the consideration here is to prevent the program from influencing the judgement of plagiarism by the instructor.

Option (2) would be more informative and helpful to the instructor.

**FUNCTIONALITY 4:**

To return these results to the client.

* Either the client or the server can handle most of the post-processing of the results.
  + It may be preferable to have the client handle the report generation since:
    - There is already work that must be done to restore identities to the submissions.
    - It may save bandwidth.
  + It might also make more sense from a design standpoint to have the server handle all the processing, and just return the result.

**General Algorithm of programs:**

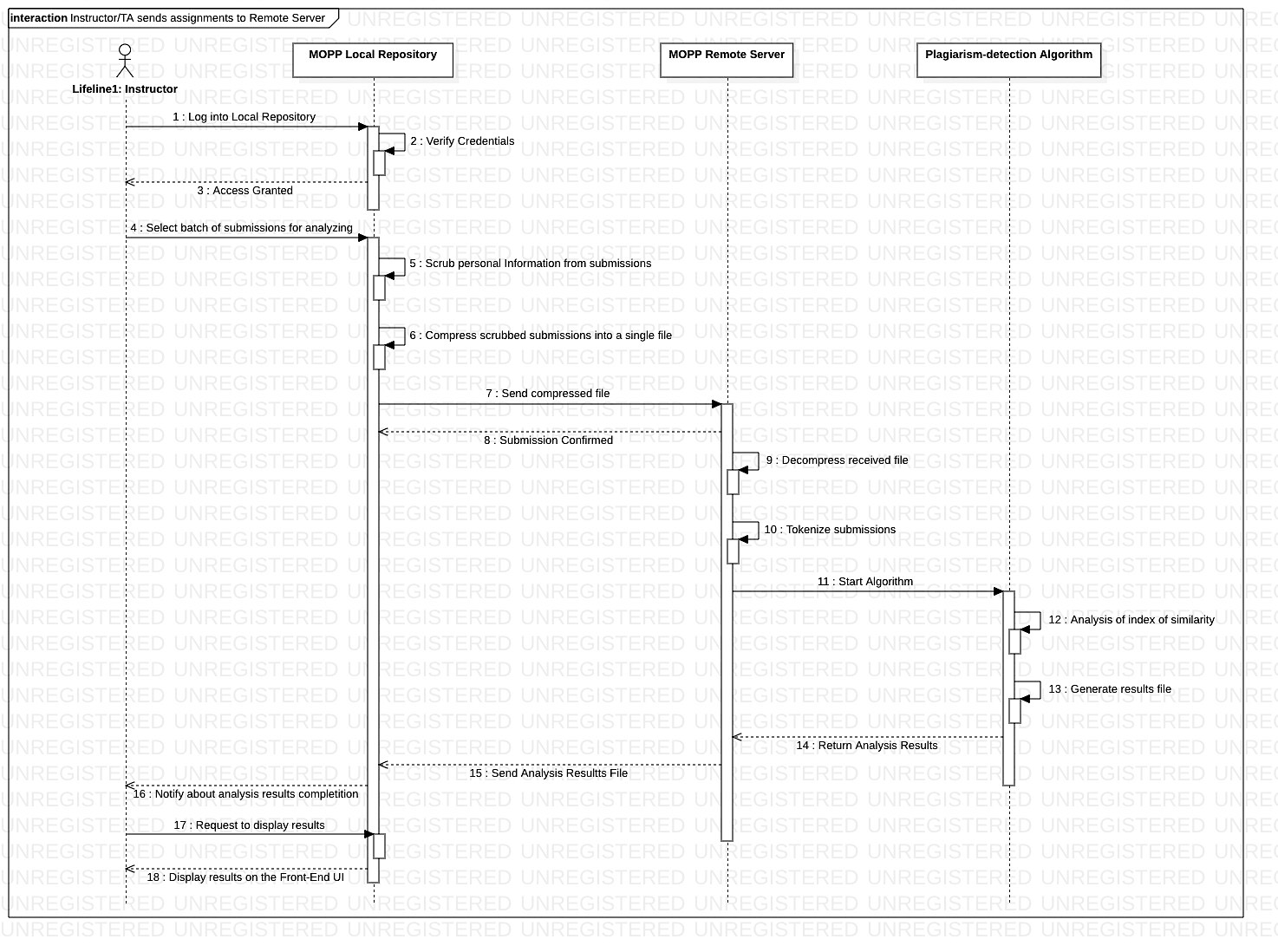
**Client program** **(submission system):**

* Accepts selection of different assignments to be compared.
* For each assignment,
  + strips it all of personal information and assigns a serial number for identification.
  + appends each source file of the submission together.
* Takes each appended file and combines them all into one file to be transferred to the server.
  + Transfers to server.
  + Server analyzes the code.
* Accepts back report from server.
* Restore student identities to the report.
* Provides report to UI.

**Remote Server (analyzer):**

* Accepts the submitted file.
* Tokenizes data.
  + Depending on how intensive this is, it could also be done on the client side. If it takes any significant length of time, it is preferable to have the server deal with it to reduce submission time.
* Compares the source for each assignment against every other assignment.
  + When a similarity is determined, it is tracked.
* For efficiency, only the current class assignments need to be checked against all other assignments. If a previous year’s assignments are included in the submission, these do not need to be checked against each other for similarity.
* Result is returned to client.

**Sequence diagram from Requirements Specification:**



**NETWORKING:**

A server/client networking system will need to be developed to handle the communication between the submission and analyzing servers:

* As discussed in meetings, Java will ideally be used for implementation.
  + C++ is also a language that can be investigated if an alternative is necessary.
* When the analyzing server accepts a server connection from the submission server, it should start a new thread to allow for multiple connections from different institutions.

**Possible stretch features:**

1. It could be useful to have a page where the instructor views the results of multiple assignments for each student. For example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Assignment 1** | **Assignment 2** | **Assignment 3** | **Assignment 4** |
| **Person A** | **23%** | **34%** | **12%** | **29%** |
| **Person B** | **5%** | **2%** | **4%** | **13%** |

**… etc. Note:** the percentages are the similarity scores.

This visual aid could assist instructors in their discovery of a student who is plagiarizing.

1. <ADD AS REQUIRED>