Software Requirements Specification

For

Code Plagiarism Detector System

Team 6

September 19, 2017

Version 1.1

**Table of Contents**

1. **Introduction…………………………………………….3** 
   1. Purpose………………………………………………………………..3
   2. Goals…………………………………………………………………..3
   3. Scope…………………………………………………………………..3
   4. Definitions……………………………………………………………..3
2. **Constraints………………………………………………4**
   1. Environment……………………………………………………………4
   2. Users……………………………………………………………………4
   3. Mandated Constraints…………………………………………………..4
3. **Non-Functional Requirements………………………….5**
   1. Usability Requirements…………………………………………………5
   2. Performance Requirements……………………………………………..5
   3. Security Requirements………………………………………………….5
   4. Documentation and Training……………………………………………5
4. **Functional Requirements………………………………..5**
   1. Feature: Check for File Similarity……………………………………….5
      1. Description and Priority
      2. Use Case
   2. Feature: Examine a detailed comparison………………………………...6
      1. Description and Priority
      2. Use Case
   3. Feature: Check Upload History………………………………………….7
      1. Description and Priority
      2. Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1.1 | 9/19/17 | James Rinella | Baseline document |

**Code Plagiarism Detector Requirements**

**1. Introduction**

The onset of personal computing has made digital copying extremely simple. The problem of detecting when an individual or organization has taken an original work and claimed it as their own exists well outside the academic setting. Collaborators create multiple nearly identical copies of documents utilize version control or the existence of software businesses who depend on compensation for digital copies they produce are some real world examples of the ease of duplicating software. Therefore, it is important to be able to detect a plagiarized work in many instances.

**1.1 Purpose**

The purpose of this Software Requirements Document is to provide client and developer requirements for the Code Plagiarism Detector. Client oriented requirements describe the system from the user’s point of view, including a description of the different types of users interacting with the system. Developer oriented requirements describe the system from the developer’s perspective, including a description of functional, data, performance, and other important requirements.

The document does not address *project* issues such as schedule, cost, development methods, development phases, deliverables and testing procedures. Those will be addressed in a separate project document and quality assurance test plan.

**1.2 Goals**

The four main goals of the Code Plagiarism Detector are:

1. Detect when a user uploaded code file is a plagiarized work.
2. Utilize plagiarism detection for at least two different programming languages.
3. Provide the user with a report of comparing his code file against a pool of other user submitted solutions.
4. Maintain a history *per user* of compared files.

Optional goals:

* Ability to compare against Github solutions.
* Implement permission for users to view or compare other users’ projects.
* Role based system.
* Utilize “fair game” code that is excluded from comparisons.

**1.3 Scope**

The CPD will allow a user to submit a code file that will be compared against all other user’s code files in a language of their choice. A report will be generated detailing a percentage of similarity between all other solutions. The user will be able to see line by line comparisons against individual files, in addition to other graphics detailing the analysis.

The system can only provide file comparisons; it will be up to the user to determine if the results constitute as plagiarism. CPD cannot determine *why* two files are similar; therefore, the issue of plagiarism is independent of the CPD system. Directories of solutions will not be permitted; only a single user submitted file is compared against all other files.

**1.4 Definitions**

|  |  |
| --- | --- |
| Term | Definition |
| .NET | Microsoft web application framework . |
| Actor | A user or software that receives a result from a use case. |
| Authentication | The function to verify the identity of someone who connects to a network resource. |
| Authorization | The function of specifying access rights/privileges. |
| CPD | Code Plagiarism Detector |
| MSS | Microsoft SQL Server |
| Role | Category of users that share similar characteristics. |
| Use Case | Describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes. |
|  |  |

**2. Design Constraints**

**2.1 Environment**

Event driven programming to web servers using JavaScript as part of the Node.js platform, providing a lightweight framework for the CPD run on. A Node.js instance will be utilized with a C++ and python thread along with a MySQL or no-SQL backend.

**2.2 User Characteristics**

Anyone from an academic or business setting concerned with the purpose and goals outlined in the preceding section.

**2.3 Mandated Constraints**

The application will run on a Windows or Linux platform.

**3. Non Functional Requirements**

**3.1 Usability Requirements**

95% of users will not need to read the user manual to be able to use the application.

90% of users should be able to perform a file comparison in under 2 minutes.

85% of users should be able to view a line by line comparison in under 2 minutes.

**3.2 Performance Requirements**

A file to file comparison should take under 10 seconds (comparing the file against all files will vary as *n* number of files stored in the system grows).

Report generation should be instantaneous.

**3.3 Security Requirements**

100% of users will access the system via authorization/authentication

**3.4 Documentation and Training**

CPD will be delivered to users as a download without documentation or training.

**4. Functional Requirements**

**4.1 Feature: Check for file similarity**

**4.1.1 Description and Priority**

CPD’s main feature to check for file similarity

Cost: None

Risk: Med

Value: High

**4.1.2 Use Case**

1. User clicks CPD application icon

2. System prompts for login

3. User selects which language to run a comparison

4. User uploads a code file and clicks submit

5. When file comparison are complete, user can browse a dashboard consisting of percentages of similarity against all the files in the system. Other visual aids or graphs appear on the dashboard

6. User clicks logout

**4.2 Feature: Examine a detailed comparison**

**4.2.1 Description and Priority**

Examine a line by line comparison between the uploaded file and another file of the user’s choice.

Cost: None

Risk: Med

Value: High

**4.2.2 Use Case**

1. User clicks CPD application icon

2. System prompts for login

3. User selects which language to run a comparison

4. User uploads a code file and clicks submit

5. On the dashboard, user clicks one of the files that was compared against the uploaded file.

6. System displays the line by line comparison, highlighting the similarities between the two files.

7. User clicks logout

**4.3 Feature: Check upload history**

**4.3.1 Description and Priority**

Check the upload and comparison history of all of the user’s files.

Cost: None

Risk: Med

Value: High

**4.3.2 Use Case**

1. User clicks CPD application icon

2. System prompts for login

3. User hovers over login name, displaying a box for viewing upload history

4. System displays a list of files uploaded by the logged in user

5. User clicks on a file, which brings the user to the dashboard (See step 5 in Use Case 4.1.2)

6. On the dashboard, user clicks one of the files that was compared against the uploaded file.

7. System displays the line by line comparison, highlighting the similarities between the two files.

8. User clicks logout