Software Requirements Specification

for

PPALMS

Version 1.0 approved

Prepared by Sam Williams, Daniel Swarts, Jack Sellner, Farhan

University of Minnesota: CSCI 5801

10/03/2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this software is to automatically generate fresh Pearson's problems. Those problems will then be used to populate an Learning Management System with an effectively infinite number of variations to train new CS students with.

1.2 Intended Audience and Reading Suggestions

The intended audience will include Clients, Instructors, and Programmers. Each should get a full understanding of the features included in the current iteration of the system best through getting and understanding via the overview, and the detailed description of the features below.

1.3 Product Scope

The scope of this project is in creating the problems themselves, not with iterating with the LMS's.

1.4 References

The sources referenced in this document are as follows:

- https://en.wikipedia.org/wiki/Parsons_problems

2. Overall Description

2.1 Product Perspective

The product is a new standalone system that is being developed in order to provide a persistent, machine-based application. It is designed to mimic similar products which have been developed for research purposes, and distributed and used over the cloud. Since the research would eventually end, this leaves those products out of service for smaller teachers. This product is to be designed as a relatively permanent solution which can remain on the machine of a teacher without expiration. Essentially, the product's goal is to take source code and generate Parson's Problems. Using the problems provided, an instructor can upload the output file to the corresponding LMS which administers the quizzes. The diagram shown in Appendix B shows the system flow with the instructor, the main user, and the LMS.

2.2 User classes and Characteristics

The only User within the scope of this project is the instructor. This software is explicitly designed for instructors to be able to make quizzes for their students.

2.3 Product Functions

Users should be able to:

- Upload source code
- Generate a new quiz
- Upload quiz to LMS
- Access previously generated quizzes

2.4 Operating Environment

The programming language is currently TBD. No matter which operating system you use, you must be able to compile and run executable programs in the decided programming language.

2.5 Design and Implementation Constraints

The main constraints are going to involve limitations on time and space. A large chunk of source code, for example, could potentially produce a near limitless number of problems which would leave the product generating for extended periods of time. Additionally, an output file of such a scenario would be a massive file which would be difficult to transport throughout a system and to an LMS by normal means in addition to the storage space it occupies. Adhering to this, a design must have a way to limit or provide tools for limitations on such edge cases.

2.6 User Documentation

Instructions for operating the systems will be included in the README. More TBD.

2.7 Assumptions and Dependencies

It is currently TBD if we will lean on prebuilt software/libraries.

3. External Interface Requirements

3.1 User Interfaces

The system will be programmed to run on a Linux bash terminal, and with a general file reading interface for complex information transfer (such as quiz generation) Other interfaces, such as a GUI, are not planned but will be considered.

3.2 Hardware Interfaces

The system will be programmed to run on Intel processor running machines, running Ubuntu Linux, but allowing other hardware interfaces is still TBD.

3.3 Software Interfaces

An API for interacting with different LMS systems is in the works, and will be how the system uploads quizzes to there. No other software interfaces are planned at this time.

3.4 Communications Interfaces

No communication interfaces are planned at this time.

4. System Features

4.1 Upload Source Code

4.1.1 Description and Priority

The system should allow instructors to upload source code for the system to generate problems out of. This is of the highest priority, as without it there will not be questions for the quizzes to be generated from.

4.1.2 User Process

- Step 1; identify source code file
- Step 2; upload file to the “input” directory within the system

4.1.3 Functional Requirements

- REQ-01: System should accept code of many different supported languages (.py, .java, .c, and .cc files)
- REQ-02: System should be able to identify files based on file extension.

4.2 Create Quizzes

4.2.1 Description and Priority

The system should allow instructors to request quizzes to be created from the problems generated from previously uploaded source code, the quantity and makeup of which will be defined by them. This is the highest priority, as generating quizzes is the primary function for the PPALMS system.

4.2.2 User Process

- Create a quiz specification .txt document.
  - This document must follow the appropriate syntax so that the system will recognize it
  - It also must include all the appropriate information in order to generate a quiz
- Upload this document to the "document" directory within the system
- compile and run the executable program that will generate the quiz

4.2.3 Functional Requirements

- REQ-03: Create many different types of pearson's problem
- REQ-04: Create problems in many different programming languages
- REQ-05: Compile problems into a human readable "quiz" .txt file

4.3 Upload Quiz to LMS

4.3.1 Description and Priority

The system should allow instructors to upload a generated quiz file directly to their preferred LMS system, via the API being produced by the backend operators. This is a medium priority, as that functionality has not been implemented yet, but eliminating an instructor's need to take the quiz file to a separate system greatly increases the ease of use of the system.

4.3.2 User Process

- Run the appropriate executable file (TBD)
- Input the name of the quiz .txt file when prompted
- Input the name of the preferred LMS when prompted

4.3.3 Functional Requirements

- REQ-6: must convert quiz .txt file to be recognized by LMS (this is not within the scope of our project)
- REQ-7: must support all common LMS systems.

4.4 Access Generated Problems

4.2.1 Description and Priority

The system should allow the instructor to access examples of problems generated from uploaded source code during quiz generation, to view, extract, and remove them as necessary. This is a low priority, it would allow an instructor to pull problems for outside quiz use (lectures), and verify questions are formatted as intended, but the system can run without this.

4.2.2 User Processes

- Step 1; User prompt system to display pre-generated problems
- Step 2; System returns numbered list of pre-generated problems and awaits instructions

- Step 3; User chooses a problem to open.
- Step 4; System prints details of problem, and awaits instruction to do with problem.
- Step 5; Users may choose to extract or delete the problem.
- Step 6; System follows directive to extract or delete the problem.
- Step 7; System prints summary and exits.

Alternative Paths: Users may exit at any time in steps 3 and 5.
Exception Paths: If no problems exist, step 1 skips to step 7.

4.2.3 Functional Requirements

- REQ-08: When prompted, list stored problems in readable format
- REQ-09: All extraction of stored problems
- REQ-10: Allow removal of stored problems

5. Other Requirements

5.1 List of Parsen's problems:

- Fill in the blank
- spot the bug
- ordering
- matching
- multiple choice
- etc.

5.2 List of supported languages:

- c
- java
- python
- pseudo-code
- etc.

5.3 Performance requirements:

- Must be able to generate quizzes and upload to LMS within a reasonable amount of time. This should cap at around 30 seconds.

5.4 Security requirements:

- Encrypt quiz text files to ensure that students cannot cheat by finding the answer key. This is of low priority.

Appendix A: Glossary

PPALMS - Parsons Problem Application for Learning Management Systems

Parsons Problem - Parson's problem is a program in which the lines of code containing the solution have been mixed up and the person solving the problem must rearrange the blocks of code into the right order.

LMS - Learning Management System

Learning Management Systems are essentially web tools designed to help instructors run courses. This includes giving announcements, posting information or lectures, the distributing, collecting, and grading of homework, quizzes, tests, and projects; displaying grades, and other general course activities. Examples include Canvas, Moodle, and Blackboard.

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

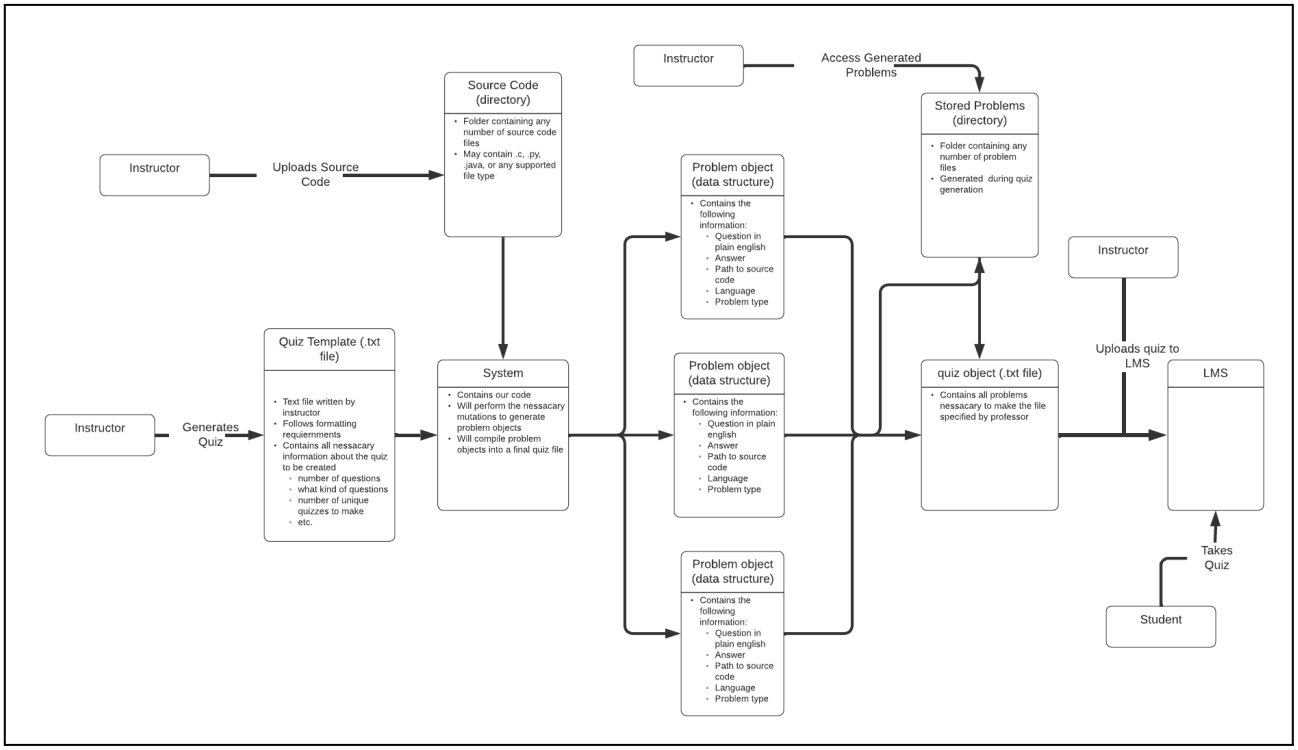

Figure 01: PPALMS Use Case Diagram

Appendix C: To Be Determined List

1. 2.4 - Operating Environment
2. 2.6 - User Documentation
3. 2.7 - Assumptions and Dependencies
4. 3.2 - Hardware Interfaces