Software Requirements Specification

for

Parson's Problems Appliance for Learning Management Systems (PPALMS)

Version 1.0 approved

Prepared by Lucas, Charlie, Harsha

UMN 5801

9/28/22

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Charlie, Harsha, Lucas	9/28/22	Setup of document and basic info completion	1.0
Charlie, Harsha, Lucas	10/3/22	Finishing and cleaning up document	1.1

Lucas Ralph	10/17/22	Fixed use case scenario diagram to be formatted cleaner. Changed use cases document to be more general.	1.2
Charlie Mattoon	10/17/22	Removed unnecessary information from sections 2.2 and 4.1-4.5. Clarified functional Requirements as well as TBD in Hardware and Software interfaces.	1.3
Lucas	11/28/22	Added system feature 4.6	1.4

Introduction

1.1 Purpose

Designing a stand-alone-system for the easy generation of parson's problems as well as other quiz like questions based off the input of source code and highlighting of included/excluded code as well as tuples of code that can be moved around to make functionally different (or non-functional) code. The requirements for Parson's Problem Appliance for Learning Management Systems (PPALMS) are specified in this document.

1.2 Document Conventions

We followed the below conventions when writing the document:

1. 11 Times font, bolded 13.5 section dividers.
2. Revision History, which includes Dates when the document was modified, along with the column, indicating the author who modified the document.
3. We categorized the priority requirements into two categories, Not a Priority, indicating there is enough time and resources to allocate for it, and Priority, indicating it should be taken care of in the current sprint.

1.3 Intended Audience and Reading Suggestions

This document is intended for the users of the LMS systems that will be implementing said proprietary software as well as for the developers of this system to provide guidance and direction for both the implementation and the usage of the PPALMS system. This document has 5 sections excluding the introduction and the overall requirements document can be summarized as below. Section 2 of this document gives the overall description of the system. It includes the product perspective, product functions, user classes, and some more information on the product. Section 3 of this document describes the external interface requirements of the system. Section 4 and 5 describes the functional and nonfunctional requirements of the system.

1.4 Product Scope

PPALMS system is used to create Parson's problems, and our goal is to create an automated system to speed up the generation of creating Parson Problems, and other potential problem possibilities. This software is to be used by professors to increase the number of outputted problems and to reduce time spent creating quizzes/exams.

1.5 References

Given below is the link to the use case document.

Link:

https://docs.google.com/document/d/1BLMDMF4CiGffWRHErqE_i_t4b7WK4-blaDiI8m0LULs/edit?usp=sharing

2. Overall Description

2.1 Product Perspective

PPALMS system is from the family of LMS (Learning Management Systems) with a functionality of Parson Problem Generation. PPALMS systems improves the efficiency of test creation by generating questions based on the permutations of working code.

Use Case diagram

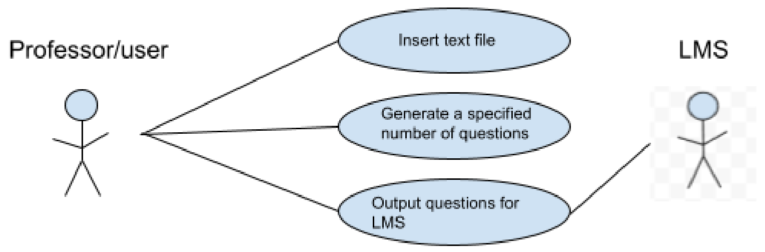

Fig. 1: Flowchart for PPALMS

2.2 Product Functions

- The user will input source code, the user (Instructor) would upload a text document containing source code for Parson's problem generation.
- Choose lines for inclusion/exclusion
- Choose the problem type, the user selects a particular question type for problem generation.

- Choose the LMS system for output.
- Choose the number of questions to be generated, the user can generate as many questions as they intend within a default cap

- Receive the exported problem set, the user can receive the problem set in the form of a text file.
- *more on Product Functions in our use cases

2.3 User Classes and Characteristics

Given below are user classes for our system.

- Professors/instructors/TA's – are the most important users, they are responsible for code generation.

- Other users with access to the system - small subset less important, they are responsible for maintenance, research, etc.

2.4 Operating Environment

We are making the assumption that our system will be designed in a linux environment, and therefore compatible in a linux environment. Our software will be compatible with Canvas, Blackboard, and Moodle. It will be operational Mac os and windows environments, although this is **TBD**.

2.5 Design and Implementation Constraints

TBD - Need to hear more from the user to determine if there are going to be any limitations dependent on the hardware/software/security of the application of our system with a separate LMS system.

2.6 User Documentation

We are planning to include the following documentations along with the software.

- User manual, that guides the user when installing and using the software.
- On-line help, that provides the user with information about the software.
- Tutorials, that provides the user with a step-by-step guide on how to use the software.

We are also planning to deliver the user documentation in the following formats:

- PDF

- HTML

2.7 Assumptions and Dependencies

Our software has to work with most LMS systems such as Canvas, Blackboard, and Moodle. We are assuming that the code uploaded by the user is runnable and working. If there are any external packages that the user might want to make compatible with our system, we can add them in as needed. We are assuming that aside from the main LMS systems there are no other external packages. We are assuming that the user has an operating system installed that is compatible with our system/software. Additional packages and dependencies needed to run our software will be shared with the user later (TBD).

3. External Interface Requirements

3.1 User Interfaces

We are planning to use a Command line-based user interface for all users, and for all functions (Uploading, annotating, selecting question type, LMS selection). The user interface is needed when uploading a text file, for all users (instructors, and students). We are not including admins and the maintenance team.

3.2 Hardware Interfaces

Due to the lack of complexity with this system, we are going to assume that as long as the system uses standard file I/O we do not have to worry about any hardware interactions. **TBD.**

3.3 Software Interfaces

We are solely using text file I/O and out that is it, there is not currently any other software apart from the LMS and our system. We plan to use a command line interface although this can be changed if requested.

3.4 Communications Interfaces

As this system works purely offline, we will have no need for any communication interfaces, this system will simply be downloaded to a computer and run there where it will take in code and output questions without the need for an internet connection of any kind.

4. System Features

4.1. Generate Re-ordering Parson Problems

4.1.1 - Description and Priority: Generate re-ordered Parson problems based on the inputted text file, if Parsons problem is the type of problem that was selected.

4.2. Generate Re-ordering 2D Parson Problems

4.2.1 - Description and Priority: Generate re-ordered 2D Parson problems based on the inputted text file, where indentation matters rather than just re-ordering of lines if Parsons problem is the type of problem that was selected.

4.3. Generate Multiple-Choice Problems

4.3.1 - Description and Priority: Generate 3 incorrect sections of code and output, and the correct code as the 4th option(not necessarily in that order).

4.4.Generate Fill in the Blank Problems

4.4.1 - Description and Priority: Generate as many different variations of the code as possible with the selected lines with lines and output variations for different sections, and prompt the user to fill in said section with the correct and original code.

4.5.Generate Fix the Error Problems

4.5.1 - Description and Priority: Generate a set of incorrect code based closely off of the correct code that was inputted. Generated one line at a time.

4.6.Removing annotations/comments from a file, or linking comments with following lines of code

4.6.1 - Description and Priority: Users are given the option to delete all annotations in the file, or link annotations to following lines of code.

4.1.2 Stimulus/Response Sequences

Inputting text file: Take in text file, to be parsed and reordered by tuples of lines.

Choosing number of questions to generate and type of questions to generate: Will change the defaulted value of the number of questions to be generated, where default is 500. If the requested amount of questions cannot be generated then it will output max number of questions, and display that it was not able to output the amount of questions requested.

4.1.3 Functional Requirements

The system must allow for text files containing valid programs to be inputted.
The system must output questions in a format compatible with the chosen LMS system.
The system must work with all LMS systems.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

We are assuming that this system will be efficient enough to run on a "general school computer" (Specification TBD based on user requirements) in a reasonable amount of time, a couple of minutes or less. Initial testing will be done using the UMN Lab Computers through **ssh** (i7 7700k with 16gb of ram).

5.2 Safety Requirements

Due to the nature of this system it is completely harmless and therefore has no associated safety requirements to be run or used on any system.

5.3 Security Requirements

Based upon our understanding of the user needs, security on this system is unnecessary as there are no negative side effects to accessing this system even if unauthorized, given that this program only works using functional source code there are no risks from student access, in other words this system would not help them gain answers to questions without already having the original source code.

5.4 Software Quality Attributes

This system needs to be simple enough that a professor or TA could sit down and understand it on their first or second use but complex enough to offer the full range of question functionality and output customization that could be generally useful to a user.

5.5 Business Rules

This is outside the scope of our project and as such is NA/**TBD.**

6. Other Requirements

None/**TBD.**

Appendix A: Glossary

LMS: Learning management system, such as; Canvas, Blackboard, or Moodle.

Parson's Problems: A type of problem that asks the student to take a section of code that has had its lines rearranged and fix the order of the lines.

Appendix B: Analysis Models

No analysis models are relevant here, None (Potentially TBD if requested).

Appendix C: To Be Determined List

- Operable operating systems (have not finished developing our product or chosen a language in which to write it in yet).
- Design and implementation constraints .
- User documentation.
- Minimum system requirements for the program to run (3.2)