Software Requirements Specification

for

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

Version 1.0

Prepared by:

Devin Allen
Stephen Cox
Benjamin Maymir
Zhenyu Yang

10/03/2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The Parson's Problems Appliance (v 1.0) is a stand-alone application that users to import source code from a file and generate quiz or exam questions from the lines of code. Question types include multiple-choice, line reordering, 2D problems, fill-in-the-blank, and bug selection. The system saves these questions to a file of type (.qti) which can be imported into Learning Management Systems (LMS): Canvas, Moodle, or Blackboard.

1.2 Document Conventions

For simplicity, the Parson's Problems Appliance for Learning Management Systems will be denoted as **PPALMS** in most cases. Learning Management System will be denoted as **LMS**. Any mention of an instructor is assumed to be a computer science educator using the system to generate questions. The words 'questions' or 'problems' carry identical meaning and can be used interchangeably. Requirements for the project may derive from previous requirements and will thus be stated as "Related Requirements"; associated use cases will also be given.

1.3 Intended Audience and Reading Suggestions

PPALMS is intended for use by instructors, institutions, and researchers using an LMS at all education levels.

Readers will find hardware, software, and developmental details discussed in the coming sections. Section 2 ("Overall Description") describes the functionality, components, and accompaniments to PPALMS. Section 3 ("External Interface Requirements") discusses the PPALMS interface and possible connections to other systems. Section 4 ("System Features") lists and explains the system requirements. Section 5 ("Other Nonfunctional Requirements") describes system requirements not associated with functionality. Section 6 ("Use Cases") details specific user interactions with the system. The requirements sections (4 and 5) are intended for all readers, however developers and testers may find them the most useful. Users (instructors and others), managers, and marketing staff may wish to consider examining the sections 3 and 4 for a brief overview of the system in its entirety.

1.4 Product Scope

PPALMS allows for easy, simple generation of computer-science-related exam questions by automating the action of modifying or rearranging lines of source code. This relieves users (instructors and organizations) of the manual process and reduces the time and resources necessary to create quizzes for students.

1.5 References

No other documents or sources were referenced within this software specification.

2. Overall Description

2.1 Product Perspective

This stand-alone software will serve as an alternative to the temporary web hosted Parson's pProblems. By creating a stand-alone application, the software will not depreciate in circumstances where grant funding runs out. By formatting the generated problems to be stored in a .qti file, users will not require additional software beyond the LMS system they are already using.

2.2 Product Functions

1. The product will be able to read source code from a provided file.
2. The product will be able to generate reordering problems, multiple choice problems, two dimensional problems, bug selection problems, and fill-in-the-blank problems.
3. The product will be able to export a bank of questions and answers in a .qti file that users shall be able to import into their desired LMS.

2.3 User Classes and Characteristics

There are two primary user classes for the product. The first is instructors that will use the product to autonomously create a large bank of practice questions for their students in order for the students to become more familiar with the style of questions. The second is researchers who want to collect data on how effective Parson's problems are at teaching programming concepts.

2.4 Operating Environment

The product must be run in an environment which has sufficient permissions to execute the provided code as well as creating and modifying files specified by the user.

2.5 Design and Implementation Constraints

Large source code files may inhibit problem generation or output and are limited to a maximum file size of 200 lines of text and 20 kilobytes. All generated problems and sets of problems must be able to be saved in a .qti file, the common file type for use with an LMS. The programming language for the completed system is yet to be determined, as the considerations for problem generation, line-selection, and file input/output may influence the chosen language.

2.6 User Documentation

The product will be shipped with a document detailing each functionality of the software, a list of hashes for all included files, and a full copy of the source code documentation.

2.7 Assumptions and Dependencies

PPALMS shall use the common regular expression libraries and tools (regex) to generate and check correctness of generated problems.

3. External Interface Requirements

3.1 User Interfaces

PPALMS shall feature a graphical user interface (GUI) containing a text box to enter file paths to import source code and an alternative method by clicking a button to open the file explorer. The interface will open the source-code in a clickable text-box where users can select lines of code to be used for problem generation. The interface will have clickable buttons to

1. Modify the number of problems generated.
2. Choose a question type for generated problems.
3. Save the generated questions and export them to a file.

The interface will also feature an error box to report errors and messages back to the user.

3.2 Hardware Interfaces

The system will support both x64 and x86 cpu architecture. Additionally, the system will interact with the computer's long term storage solution, HDD or SSD, when reading in source code files and outputting a set of generated questions to a new file.

3.3 Software Interfaces

The system shall not use any outside libraries, databases, applications, or systems that are unavailable by default.

3.4 Communications Interfaces

PPALMS is a stand-alone system and has no communication to other systems, programs, processes, files, or networks outside of the imported source code by the user.

4. System Features and Requirements

Importing Source Code

Requirement: 1 **Related Requirements:** 3, 4 **Use Case:** 1

Author: Devin Allen **Date:** Oct 2, 2022

Introduction: The first step in generating code questions is obtaining source code from which the questions can draw content.

Rationale: In order for the users to generate questions the system will need the ability to grab source code on the computer's hard drive.

Inputs: A file and its path.

Requirement Description: Users can import a file containing source code by specifying the file's path on the hard drive. The file shall be imported by the system if and only if:

1. The file path points to an existing file.
2. The file contains only text data.

Outputs: N/A

Stand-Alone System

Requirement: 2 **Related Requirements:** 1 **Use Case:** N/A

Author: Devin Allen **Date:** Oct 2, 2022

Introduction: The system will be independent of other programs local or online.

Rationale: Interactions between the system and other programs could introduce unnecessary complexity to the user's experience. Additionally, stand-alone applications typically run faster compared to web applications.

Inputs: N/A

Requirement Description: The system will operate independently; there will be no interactions between the system and other software on the user's computer or over the internet/network.

Outputs: N/A

Limit on Source Code Size

Requirement: 3 **Related Requirements:** 1, 4 **Use Case:** 1

Author: Devin Allen **Date:** Oct 2, 2022

Introduction: The system will have a limit on the size of source code files imported.

Rationale: Since the system will generate multiple possible variations of a given question type, a limit needs to be placed on the size of the imported source code file in order to prevent the system from stalling during question generation.

Inputs: N/A

Requirement Description: The system shall not allow the user to import a source code file containing more than 200 lines of text or 20 kilobytes of memory.

Outputs: N/A

Original Source Code File Will Not Be Modified by the System

Requirement: 4 **Related Requirements:** 1, 3 **Use Case:** 1

Author: Devin Allen **Date:** Oct 2, 2022

Introduction: The system will not alter the contents of the imported source code file it uses to generate questions.

Rationale: In case the file to be imported is important to the user, the system will not alter it. It is also unnecessary to modify the file as the system can simply copy its data prior to question generation.

Inputs: Source code file.

Requirement Description: The system will not alter the source code file imported by the user before, during, or after question generation. Instead, a temporary copy of the file will be made to draw content for question generation.

Outputs: N/A

Selecting Lines from Source Code

Requirement: 5 **Related Requirements:** 1 **Use Case:** 2

Author: Devin Allen **Date:** Oct 2, 2022

Introduction: The user will be able to pick and choose whichever lines of code from the imported file they wish to include in the question generation.

Rationale: In order to generate any meaningful questions the user will need the ability to decide the contents of the questions they wish to create.

Inputs: Source code.

Requirement Description: Users can mark specific lines in the source code to designate them as included for question generation through the graphical user interface. The system will then only generate questions utilizing the indicated lines.

Outputs: N/A

Problem Generation

Requirement: 6 **Related Requirements:** 1, 5 **Use Case:** 3

Author: Zhenyu Yang **Date:** Oct 1, 2022

Introduction: The system can generate a variety of questions based on a chosen type of question and answer method.

Rationale: The question type and answer type that users are looking for might be different. Additionally, students may learn better from answering certain types of questions. The system shall satisfy generating different types of questions.

Inputs: Source code, selected lines of code, selected type of problem, number of generations.

Requirement Description: The system can generate a set of problems for the selected lines of code from a source file and selected type of question:

1. Reordering (unscrambling) problems.
2. 2D (indentation) problems.
3. Multiple choice.
4. Bug-selection.
5. Fill-in-the-blank.

After selection, the system generates mutations on the selected lines of code and generates sets of questions based on the selected type.

Outputs: A set of generated problems based on the user's selection.

Reordering Problems

Requirement: 7 **Related Requirements:** 1, 5, 6 **Use Case:** 3

Author: Zhenyu Yang **Date:** Oct 1, 2022

Introduction: Users can generate reordering / unscrambling questions questions which use tuples (sets of lines) that hold lines of source code.

Rationale: Users may want to test students or others on an LMS by using reordering/unscrambling questions.

Inputs: Source code, selected lines of code.

Requirement Description: The system can split the selected lines of code into tuples and rearrange the sequence of tuples into a new order that students will have to rearrange into the correct order.

Output: Code-reordering problems.

2D Problems

Requirement: 8 **Related Requirements:** 1, 5, 6, 18 **Use Case:** 3

Author: Zhenyu Yang **Date:** Oct 1, 2022

Introduction: When imported source code is of the Python programming language, users can generate 2D Problems (indentation problems) for selected lines of code.

Rationale: Some particular languages have strict rules for indentation (e.g., Python) and the system shall be able to generate possible questions based on that rule.

Inputs: Source code, selected lines of code.

Requirement Description: The system extracts whitespace from the selected lines of source code and generates a set of questions based on the standard indentation and spacing conventions of Python.

Outputs: A set of 2D problems.

Multiple-Choice Problems

Requirement: 9 **Related Requirements:** 1, 5, 6 **Use Case:** 3

Author: Zhenyu Yang **Date:** Oct 1, 2022

Introduction: The system can generate mutations on single lines of source code to create multiple-choice questions.

Rationale: Multiple-choice questions are one of the most commonly used types of questions on quizzes and exams, as they provide a simple alternative to open-ended problems. The system shall be able to generate this type of problem.

Inputs: Source code, selected lines of code.

Requirement Description: Users can generate multiple-choice problems based on single lines of selected source code. The system creates mutations on the line of code and provides those mutations as incorrect answers, coupled with the correct answer, as a multiple-choice problem.

Outputs: A set of multiple-choice problems.

Bug and Error Problems

Requirement: 10 **Related Requirements:** 1, 5, 6 **Use Case:** 3

Author: Stephen Cox **Date:** Sept 30, 2022

Introduction: The system can generate questions that prompt students to select bugs and errors.

Rationale: Computer programming is highly prone to error and bugs are extremely common occurrences. Students should be able to read code and locate mistakes in not only their own code but others' as well. Bug-selection questions are a great addition to the types of questions asked in computer science quizzes.

Inputs: Source code, selected lines of code.

Requirement Description: Users can generate bug-selection problems by selecting lines of code and the bug-selection question type. The system mutates the selected lines of code and returns a set of problems with bugs placed into these lines of code.

Outputs: A set of bug-selection problems.

Fill-in-the-blank Problems

Requirement: 11 **Related Requirements:** 1, 5, 6 **Use Case:** 3

Author: Stephen Cox **Date:** Oct 1, 2022

Introduction: Users can generate fill-in-the-blank questions where students are asked to fill in blanks in code with their own answers.

Rationale: Students should be able to write and change code as necessary. Fill-in-the-blank questions test students' abilities to do so.

Inputs: Source code, selected lines of code.

Requirement Description: Users can generate fill-in-the-blank problems by selecting lines of code and the fill-in-the-blank question type. The system mutates the selected lines of code by removing pieces of text and returns a set of fill-in-the-blank problems:

1. The problem will show the lines of source code with a blank in place of code.
2. The removed code will be stored so that it can be compared to the student's answer.

Outputs: A set of fill-in-the-blank problems.

Maximum Number of Generated Questions

Requirement: 12 **Related Requirements:** N/A **Use Case:** 4

Author: Stephen Cox **Date:** Oct 1, 2022

Introduction: The system shall have a default maximum number of variations (set to 5) when generating questions. This number can be configured by the user in a range from 1 to 25.

Rationale: Users shall be able to generate a specified number of questions from their selected lines of source code. As generating mutations for questions are taxing operations, the user is limited to a maximum number of 25.

Inputs: An integer from 1 to 25.

Requirement Description: Users can change the default maximum number of generated question variations. The method to change this value is based off of the user interface (UI) and is to be decided.

Outputs: N/A

Generated Problems Stored in System

Requirement: 13 **Related Requirements:** 6-11 **Use Case:** 5

Author: Stephen Cox **Date:** Oct 1, 2022

Introduction: The system generates questions and stores them for later output to a file.

Rationale: Users are able to generate multiple sets of questions in one session of using the system. These questions must be stored for later output.

Inputs: A set of questions.

Requirement Description: After the system generates a set of questions, the sets of questions shall be stored and deemed ready for output to a file. The sets of questions are held in memory until the user declares they are finished using the system.

Outputs: Questions stored in a question bank that are ready to be output to a file.

Generated Problems Output to a File

Requirement: 14 **Related Requirements:** 11 **Use Case:** 5

Author: Stephen Cox **Date:** Oct 1, 2022

Introduction: Generated sets of questions that have been stored are output to a .qti file, ready for implementation with an LMS.

Rationale: When users are finished generating questions, the questions must be saved to a file that can be imported into an LMS. A .qti file satisfies the required file type.

Inputs: The stored generated questions, user selects they are done.

Requirement Description: When users select they are finished, the system takes stored sets of questions and saves them together in a .qti file. The system exits after performing this operation.

Outputs: A .qti file containing sets of generated questions.

Incorrect Answers do not Function the Same as Correct Answers

Requirement: 15 **Related Requirements:** 1, 6, 9 **Use Case:** 3

Author: Benjamin Maymir **Date:** Oct 2, 2022

Introduction: When the user selects where mutations will be generated, the generated mutations shall not function interchangeably with the unmutated code.

Rationale: It is important for the sake of clarity that answer choices designed to be incorrect are not correct to ensure that the LMS will never incorrectly grade a problem

Inputs: Source code, selected lines of code.

Requirement Description: The system that generates mutations for bug locating or multiple choice questions shall never create a mutation that is fully functional within the context of the problem.

Outputs: A set of problems guaranteed to contain incorrect code snippets.

Starting Positions for Lines in Reordering Problems

Requirement: 16 **Related Requirements:** 1, 5, 6, 7 **Use Case:** 3

Author: Benjamin Maymir **Date:** Oct 2, 2022

Introduction: Reordering questions will be presented with no more than one of every four lines starting in a correct position.

Rationale: Reordering questions are designed to test the student's understanding of the intended flow of the provided source code. If a substantial portion of the code is already in the correct order, the student loses the opportunity to critically analyze each line of the code or may be confused by the apparent lack of errors within the problem.

Inputs: Source code, a set of potential reordering problems

Requirement Description: When a user generates a set of reordering problems, the system will produce initial states of the problems with no more than one of every four lines of the source code starting in a position that will satisfactorily answer the presented problem.

Outputs: A set of reordering problems.

System Hash Included in Releases

Requirement: 17 **Related Requirements:** 1, 8, 14 **Use Case:** N/A

Author: Benjamin Maymir **Date:** Oct 2, 2022

Introduction: Each released version of the system will be accompanied by a hash of the program files.

Rationale: Since the system must be a stand-alone application, it is imperative that the user is able to trust that the code that they run on their machine is the same as the code that the development team has produced. By providing hashes along with the operating files, the work required to create a malicious program that presents itself as the released software becomes exponentially more difficult.

Inputs: Source code.

Requirement Description: When a user attempts to generate a two dimensional problem for source code that is not written in Python, Make, or Haskell , the system will reject the request and inform the user that only source code written in one of the supported languages can be used to create two dimensional problems

Outputs: Either a set of two dimensional problems or an error visible to the user indicating that the request was rejected.

Usable Languages for 2D Problems

Requirement: 18 **Related Requirements:** 8 **Use Case:** 3

Author: Benjamin Maymir **Date:** Oct 2, 2022

Introduction: Two-dimensional (2D) problems will only be generated for source code written in languages that include syntactically significant whitespace.

Rationale: In order for a problem to assist a student to understand the presented material, the possibility of failure must exist. In languages including but not limited to Java, C, and Javascript, all answers differing only in indentation are equally correct, so generating incorrect answers is impossible.

Inputs: N/A

Requirement Description: When a user receives a copy of the software, they will also receive a list of SHA-256 hashes for every included file allowing them to independently verify the authenticity of the software.

Outputs: A list of SHA-256 hashes for all included files.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

PPALMS allows source code from the common programming languages Java, C, C++, and Python. Source code files must not exceed 200 lines – where each line does not exceed 80 characters – or 20 kilobytes of memory.

5.2 Safety Requirements

The system will not add-to, delete, or modify imported source code in any way. In addition, other files on a user's device will not be affected by question generation through PPALMS. The system also does not have the capability of compiling code and cannot execute commands from source code. Imported source code is assumed to be correct – that it compiles, runs, and executes as the user intends, free of error.

5.3 Security Requirements

PPALMS will always be packaged along with a list of hashes for each file in the product. The user has the ability to verify the authenticity of the program by hashing the files they received and comparing the hashes to the hash list.

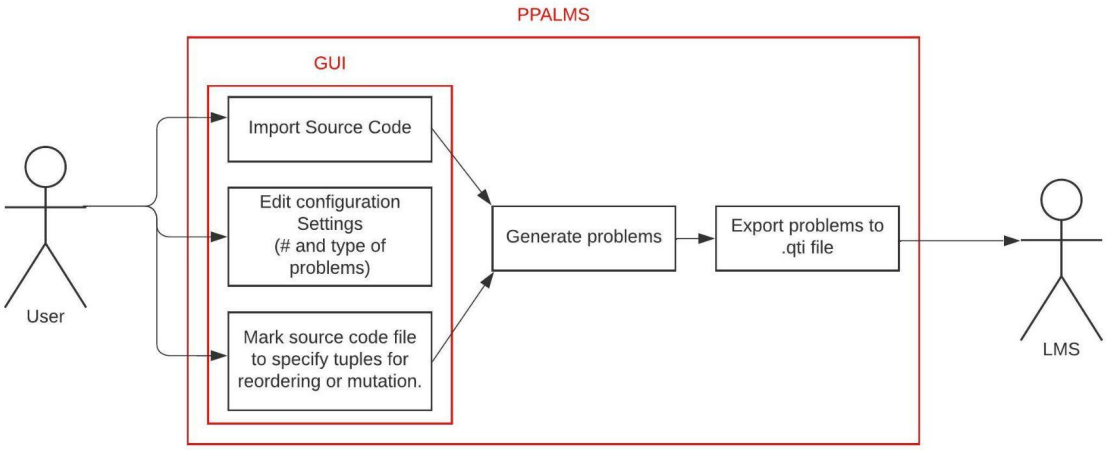
5.4 Software Quality Attributes

PPALMS shall be a flexible, modular system which can be expanded upon to allow source code to contain code from additional programming languages not currently supported. For ease of use, the system will contain a simple method to import source code and select lines for question generation (the exact method is to be decided). PPALMS will generate problems by testing them against the source code for equivalency – to prevent problems from having no correct or incorrect answers.

5.5 Business Rules

Users will only be able to generate problems from imported source code to be exported to an LMS – the system contains no other functionality. The source code will not be verified for compiling or runtime errors and will be assumed to be executable and functional to its full extent. Users should compile their code and make sure it is free from errors before importing to PPALMS.

6. Use Cases



Import Source Code from File

Use Case: 1 **Author:** Stephen Cox **Date:** Oct 2, 2022

Actors: User, PPALMS

Summary: Users choose and import text from a source file to the system.

Basic Course of Events:

1. Users starts the system.
2. User selects the option to import a file through the GUI.
3. System recognizes the existence of the source code file.
4. System checks size of the given file.
5. System proceeds to user line-selection.

Alternative Paths: Step 3: If the file does not exist, user is notified to enter a new file path. Step 4: If the file is larger than 200 lines of text or 20 kilobytes of memory, it is rejected and the user is prompted to enter a new file path.

Exception Paths: None.

Trigger: Start of system – user prompted to select a file.

Assumptions: User has a file of working, correct source code written in Java, C, C++, or Python.

Preconditions: A file has not already been chosen or imported.

Postconditions: Source code file verified to exist and user can move to selecting lines of code for problem generation.

Selecting Lines of Code and Question Type

Use Case: 2 **Author:** Stephen Cox **Date:** Oct 2, 2022

Actors: User, PPALMS

Summary: User selects which lines from their source code from which they would like to generate problems. Then, the user selects a question type from Reordering, 2D, Multiple Choice, Bug and Error, Fill-in-the-blank.

Basic Course of Events:

1. User selects lines of source code through the GUI.
2. User selects the question type through the GUI.

Alternative Paths: None.

Exception Paths: Step 2: If lines of code were not selected, the system relays an error message to the user and prompts them to select lines of code.

Trigger: User must select lines of code and question type to generate questions.

Assumptions: User has imported source code written in Java, C, C++, or Python, and compiles and runs correctly according to the user.

Preconditions: Source code file chosen and imported.

Postconditions: Lines of code selected.

PPALMS Generates Problems

Use Case: 3 **Author:** Stephen Cox **Date:** Oct 3, 2022

Actors: PPALMS

Summary: The system generates problems for the user based on their selected lines of code and their selected question type.

Basic Course of Events:

1. System determines if the selected lines of code can be converted to questions of the given type (e.g., only Python accepted for 2D problems).
2. System generates problems based on selected question type using different methods for each type:
    a. Reordering problems will take tuples of lines and reorder them in the source code.
    b. 2D problems will take away or introduce whitespace where there shouldn't be.
    c. Multiple-choice problems will mutate lines of code and include them as possible answers for a portion of code.
    d. Bug and Error problems will introduce bugs into the lines of code or modify the code so it would not compile or run correctly in a realistic setting.
    e. Fill-in-the-blank problems will extract a portion of code from a line and ask users to type in their own answer, which will be checked against the extracted code.
3. Generated problems checked for correctness and equivalency issues, to ensure problems are solve-able and answerable.

Alternative Paths: Step 3: If problems have no correct answer, the system attempts to generate new problems.

Exception Paths: Step 1: If problems are unable to be generated, the user will be notified with an error message.

Trigger: User wishes to generate problems for their code.

Assumptions: System given source code and lines selected.

Preconditions: User has selected lines of code and a question type.

Postconditions: Problems for the given lines and type generated.

Configuring Maximum Number of Generated Problems

Actors: User, PPALMS

Use Case: 4 **Author:** Stephen Cox **Date:** Oct 3, 2022

Summary: User can adjust the default number of generated problems (5) within the range 1 to 25 – the new number of problems will be generated.

Basic Course of Events:

1. User changes the number of generated problems to a number from 1 to 25 or to the default value through the GUI.
2. The number of generated problems (when choosing to generate) will now be equivalent to the designated value.

Alternative Paths: None.

Exception Paths: User is warned if they try to change the value to a number outside the range 1 to 25 or to the default value.

Trigger: User wants to generate more or less variations of a problem type.

Assumptions: None.

Preconditions: None.

Postconditions: Number of problems generated changed to user's selection.

Generated Problems Output to a File

Actors: PPALMS

Use Case: 5 **Author:** Stephen Cox **Date:** Oct 3, 2022

Summary: All generated problems are saved in a .qti file for later use within an LMS.

Basic Course of Events:
1. Problems generated (see Use Case 3).
2. Problems exported and saved within a .qti file.

Alternative Paths: None.

Exception Paths: Step 2: If problems were not generated, they will not be saved to the file.

Trigger: User has generated problems and wants to export them to an LMS.

Assumptions: User's system can create and save to .qti files.

Preconditions: Problems have been generated for output.

Postconditions: .qti file containing all new and previously-generated problems saved to same folder containing the original source code.

Appendix A: Glossary

LMS: Abbreviation for Learning Management System; a software system that assists educators, for use by both instructors, students, and institutions.

Mutation: Code that has been modified by the system for generated problems.

PPALMS: Abbreviation for the name of the system (Parson's Problems Appliance for Learning Management Systems).

Regex: (1) Regular expression. (2) The regular expression libraries used to check expressions for equivalency and correctness.

Variation: (see Mutation).

Appendix B: Analysis Models

N/A: No pertinent models.

Appendix C: To Be Determined List

1. Testing methods of the system.
2. The base computing language the system will be written using.
    a. Supported operating systems