Software Requirements Specification

for

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

Version 1.1

Prepared by:

Devin Allen
Stephen Cox
Benjamin Maymir
Zhenyu Yang

10/17/2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Devin Allen Stephen Cox Benjamin Maymir Zhenyu Yang	10/12/2022	Requirements Review. Updated Use Cases and Requirements to reflect proper requirements. Added system models and diagrams to Appendix B.	1.1

1. Introduction

1.1 Purpose

The Parson's Problems Appliance (v 1.0) is a stand-alone application that allows 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 are used interchangeably. Requirements for the project may derive from previous requirements and will thus be stated as "Related Requirements"; associated use cases shall 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 shall serve as an alternative to the temporary web hosted Parson's Problems. 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 shall read source code from a provided file.
2. The product shall generate reordering problems, multiple choice problems, two dimensional problems, bug selection problems, and fill-in-the-blank problems.
3. The product shall 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 class consists of instructors that shall 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 class consists of researchers who want to collect data on how effective Parson's problems are at teaching programming concepts.

2.4 Operating Environment

PPALMS shall operate on current versions of Windows, Linux, and Mac computers, in an environment in which the user can create, modify, and execute. Since PPALMS is a stand-alone system, there are no compatibility requirements or concerns with other software.

2.5 Design and Implementation Constraints

Large source code files may inhibit problem generation or output. PPALMS shall require the computer to contain enough memory and storage to allow for problem generation and creation of a .qti file (the common file type for use with an LMS) containing the sets of problems. PPALMS shall be developed in a programming language with considerations towards problem generation, line-selection, and file input/output, which may influence design and implementation choices.

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 utilize a graphical user interface (GUI) to handle user input, configuration of settings, and presentation of error messages. The interface shall feature a simple, clear, and easy-to-use layout.

3.2 Hardware Interfaces

The system shall support both x64 and x86 cpu architecture. Additionally, the system shall 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

PPALMS interacts with the user's file system to import source code and generate a .qti file containing a set of generated problems. The system shall not use any outside libraries, databases, applications, or systems that are unavailable by default. The system does not require the use of other programs to perform its base functionality, apart from the GUI system and the user's file system.

3.4 Communications Interfaces

PPALMS is an offline locally-run system. The system outputs a .qti file that shall be compatible with the user's choice of LMS (Learning Management System): Canvas, Blackboard, or Moodle.

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 shall need the ability to grab source code on the computer's hard drive.

Inputs: A file and its path.

Requirement Description: The user shall 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.

Otherwise, an error message shall be displayed to the user.

Outputs: If the file path does not exist or the file is empty, the system shall output an error message to the user stating the respective reason.

Warning on Line Selection

Requirement: 2 **Related Requirements:** 1, 3, 4 **Use Case:** 3

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

Introduction: The system shall warn users about possible performance issues when selecting large amounts of lines for problem generation (greater than 200).

Rationale: Since the number of variations of a given question type increases exponentially as the number of lines selected increases, the user should be warned of potential performance issues should they select a large number of lines.

Inputs: Source code file.

Requirement Description: The system shall issue a warning message to the user when selecting a large number of lines.

Outputs: If the file follows the 200-line restriction, no output shall occur. If the file contains more than 200 lines of text, a warning message shall display to the user stating performance could be impacted.

Original Source Code File Will Not Be Modified by the System

Requirement: 3 **Related Requirements:** 1, 2 **Use Case:** 1

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

Introduction: The system shall 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 shall 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 shall not alter the source code file imported by the user before, during, or after question generation.

Outputs: N/A

Selecting Lines from Source Code

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

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

Introduction: The system shall enable users to select specific lines in their imported source code to designate they are included in 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: The system shall allow users to to select lines from within the source code which will be the target for problem generation.

Outputs: A set of line numbers from which the system can generate problems.

Selecting the Intended LMS

Requirement: 5 **Related Requirements:** 6 **Use Case:** 2

Author: Stephen Cox **Date:** Oct 14, 2022

Introduction: Users shall select an intended LMS for output, which will determine what types of problems can be generated. Users shall select from Canvas, Blackboard, or Moodle.

Rationale: Canvas does not support reordering problems. Therefore, if the user selects Canvas, they are not able to produce reordering problems using PPALMS.

Inputs: The desired LMS: Canvas, Blackboard, or Moodle.

Requirement Description: The system shall allow the user to select the intended LMS target, which shall accept the output .qti file after problem generation.

1. The user selects the intended LMS.
2. PPALMS disallows reordering problem-generation if the user selects Canvas.

Outputs: N/A

Problem Generation

Requirement: 6 **Related Requirements:** 1, 4, 5 **Use Case:** 2, 4

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

Introduction: The system shall 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 shall 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, 4, 5, 6 **Use Case:** 4

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

Introduction: Users shall generate reordering / unscrambling 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 shall allow users to 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, 4, 5, 6, 18 **Use Case:** 4

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 shall allow the extraction of 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, 4, 5, 6 **Use Case:** 4

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: The system shall generate multiple-choice problems based on single lines of selected source code. The system shall generate both incorrect answers and correct answers based on the code provided.

Outputs: A set of multiple-choice problems.

Bug and Error Problems

Requirement: 10 **Related Requirements:** 1, 4, 5, 6 **Use Case:** 4

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: The system shall generate bug-selection problems by selecting lines of code and the bug-selection question type. The system shall generate both incorrect answers and correct answers based on the code provided.

Outputs: A set of bug-selection problems.

Fill-in-the-blank Problems

Requirement: 11 **Related Requirements:** 1, 4, 5, 6 **Use Case:** 4

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: The system shall generate fill-in-the-blank problems by selecting lines of code and the fill-in-the-blank question type. The system shall mutate selected lines of code 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:** 6-11 **Use Case:** 2, 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: The system shall allow users to change the default maximum number of generated question variations using the GUI.

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 while the user is still 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:** 13 **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: The system shall take stored sets of questions and save them together in a .qti file. when users select they are finished. The system shall exit 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 **Use Case:** 4

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 shall generate 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, 4, 6, 7 **Use Case:** 4

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

Introduction: Reordering questions shall be generated with no line in a correct starting position. By requiring that every line be repositioned, the question requires a student to understand the functionality of every line in the problem.

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 shall produce initial states of the problems with no line starting in a correct position.

Outputs: A set of reordering problems.

System Hash Included in Releases

Requirement: 17 **Related Requirements:** 1, 8 **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 receives a copy of the software, they shall also receive a list of hashes for every included file allowing them to independently verify the authenticity of the software.

Outputs: A list of hashes for all included files.

Usable Languages for 2D Problems

Requirement: 18 **Related Requirements:** 8 **Use Case:** 3, 4

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

Introduction: Two-dimensional (2D) problems shall 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: 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 shall 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.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

PPALMS allows source code (as a text file) that contains correct, compilable programming language to be imported and used to generate problems. Without a limit to the amount of problem mutations, the system could produce a near infinite amount given certain lines of text. The number of mutations shall be limited to a maximum of 25 per problem – configurable by the user, with a default value of 5. The number of selected lines may impact system performance; if the user selects a large number of lines for problem generation, a warning shall be displayed indicating that performance of the system shall be impacted.

5.2 Safety Requirements

The system shall 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.

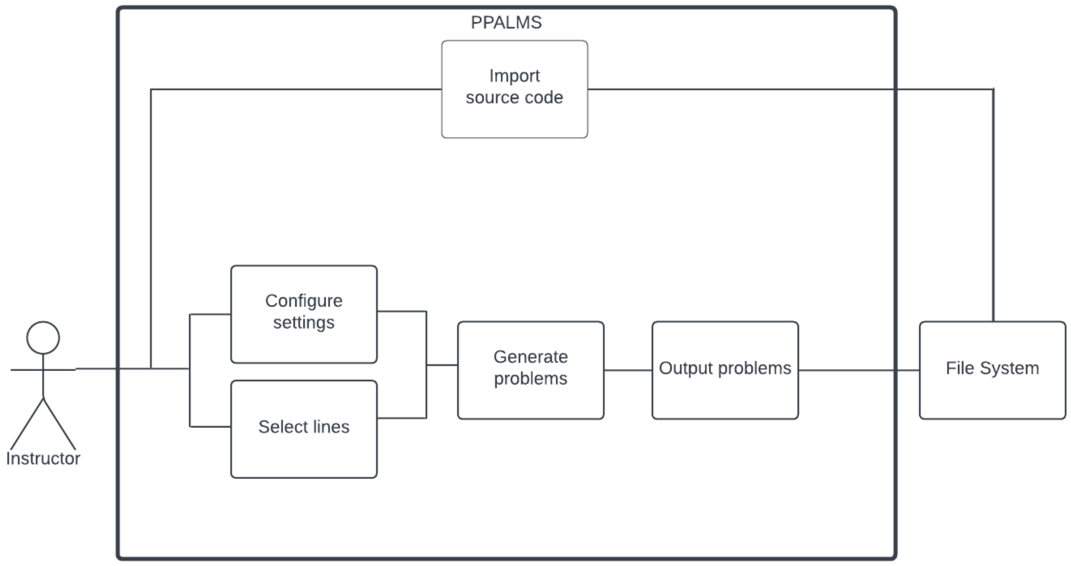
5.4 Software Quality Attributes

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 shall 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: Instructor, File System

Summary: Instructors 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 proceeds to user line-selection.

Alternative Paths: Step 3: If the file does not exist, the instructor is notified 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 a programming language.

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.

Configuring Initial Settings

Actors: Instructor

Use Case: 2 **Author:** Stephen Cox, Benjamin Maymir **Date:** Oct 3, 2022

Summary: The instructor can adjust the default number of generated problems (5) within the range 1 to 25 – the new number of problems will be generated. The instructor can also specify what type of LMS the problems will be generated for, from Canvas, Moodle, and Blackboard. Then, the user selects a question type from Reordering, 2D, Multiple Choice, Bug and Error, Fill-in-the-blank.

Basic Course of Events:
1. Instructor changes the number of generated problems to a number from 1 to 25 or to the default value through the GUI.
2. Instructor changes the LMS target to Canvas, Moodle, or Blackboard through the GUI.
3. Instructor changes the type of problems to generate.
4. The system settings will now reflect the choices made by the instructor.

Alternative Paths: Step 1: If the instructor enters a value for how many problems to generate that is not within the specified range, an error message is displayed and the user is prompted to enter a value that is valid. Step 3: If the instructor selected Canvas as the LMS target then the 2D problem type shall not be available.

Exception Paths: If the instructor exits the settings interface without having entered valid values, such shall be set to their defaults.

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

Assumptions: None.

Preconditions: None.

Postconditions: System settings are consistent with the instructors preferences.

Selecting Lines of Code

Use Case: 3 **Author:** Stephen Cox **Date:** Oct 2, 2022

Actors: Instructor

Summary: User selects which lines from their source code from which they would like to generate problems.

Basic Course of Events:

1. User selects lines of source code through the GUI.

Alternative Paths: Step 1: If the user selects more than 200 lines, the system shall output an error message to the user stating that performance is impacted by selecting a large number of lines.

Exception Paths: Step 1: 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 a programming language that compiles and runs correctly.

Preconditions: Source code file chosen and imported.

Postconditions: Lines of code selected.

PPALMS Generates Problems

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

Actors: Instructor

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 solvable 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 shall 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.

Generated Problems Output to a File

Actors: Instructor, File System

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

Exception Paths: Step 2: If problems could not be saved to a file, the system shall output an error message to the user stating the reason.

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 the 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.

Parson's Problems: Computer programming problems based on modifying, reordering, or selecting lines of code.

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

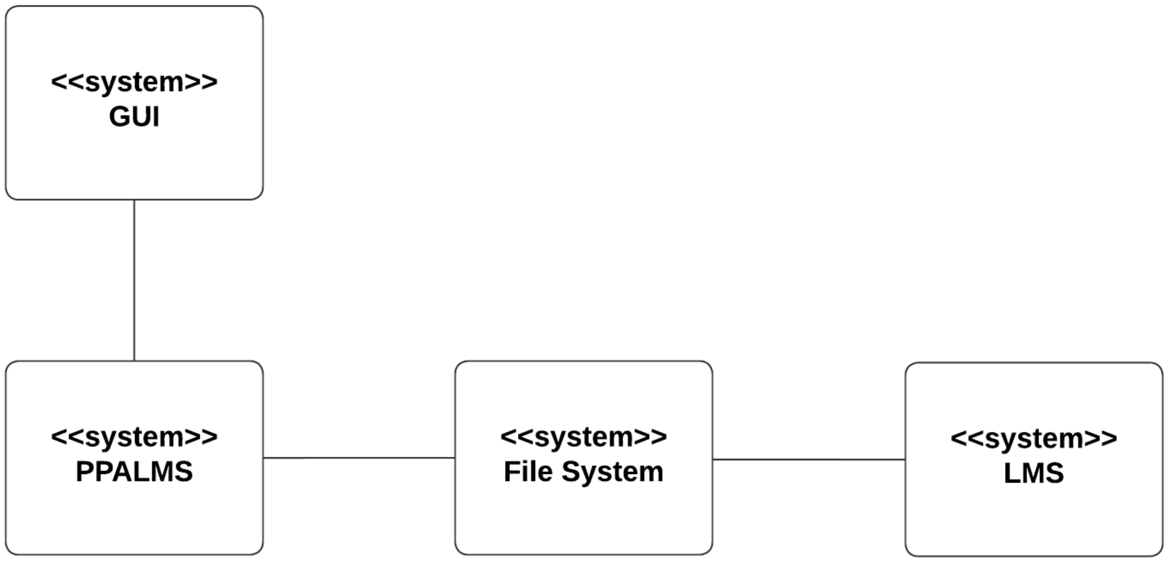
Regex: (1) Regular expression. (2) The regular expression libraries used to check expressions for equivalency and correctness.

.qti: (Question and Test Interoperability) A file type containing generated problems, able to be imported into an LMS.

Variation: (see Mutation).

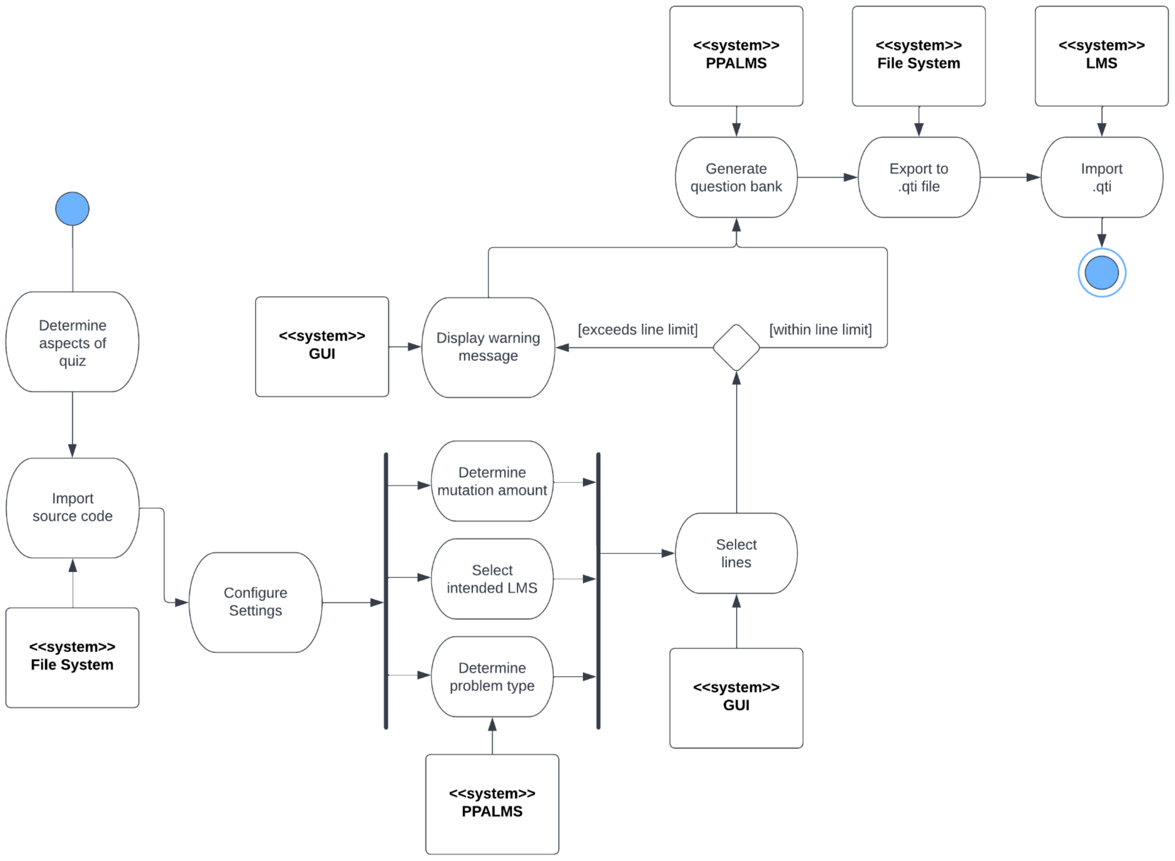
Appendix B: Analysis Models

i. Context Model



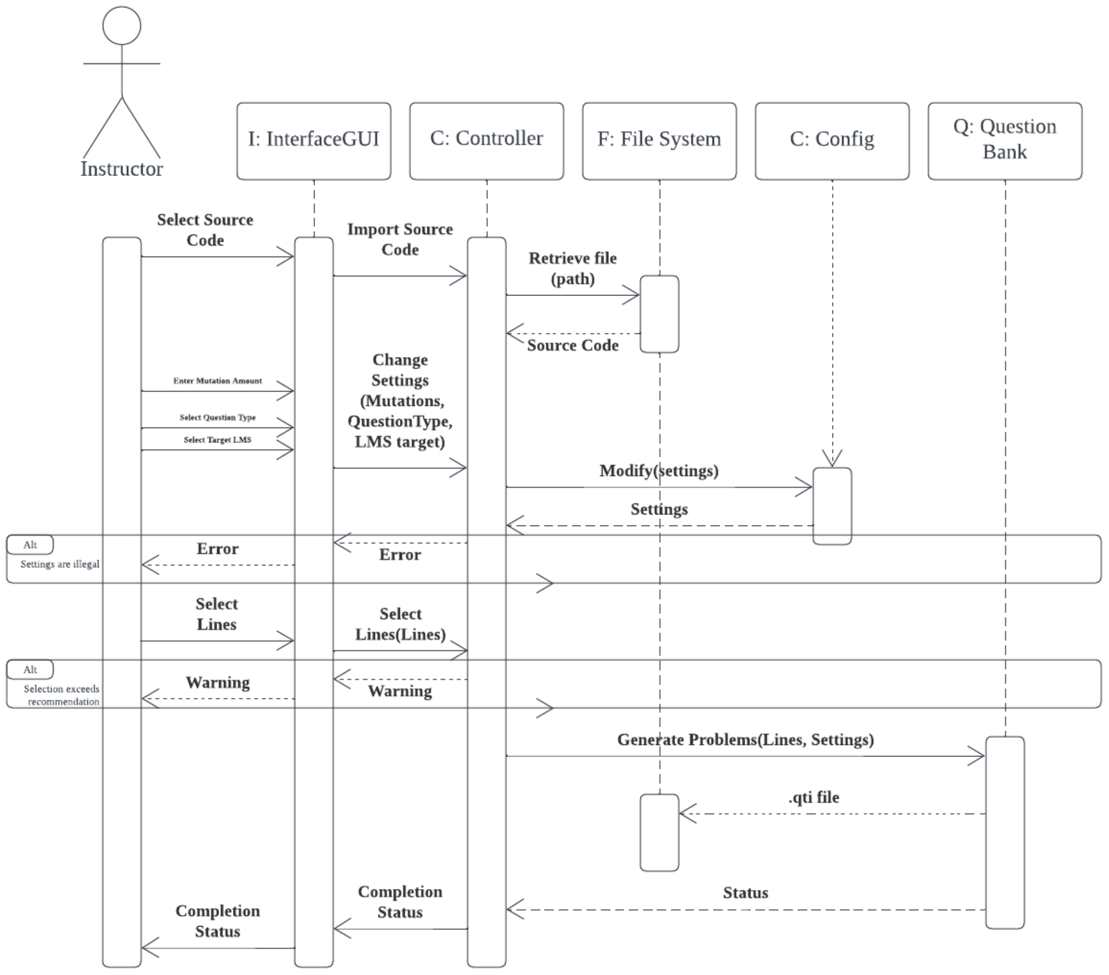
The diagram above is a context model illustrating the standard environment in which the PPALMS system shall operate in. As shown above the PPALMS system is connected to a graphical user interface (GUI) system and a file management system dependent on the underlying operating system, which is also connected to a learning management system (LMS). The PPALMS system utilizes the GUI system to handle instructor input when designating lines for inclusion in the source code file and to output error messages. PPALMS uses the file management system to import source code and export .qti files containing the generated question bank. Lastly, the LMS utilizes the output files generated by PPALMS.

ii. Business Process Model



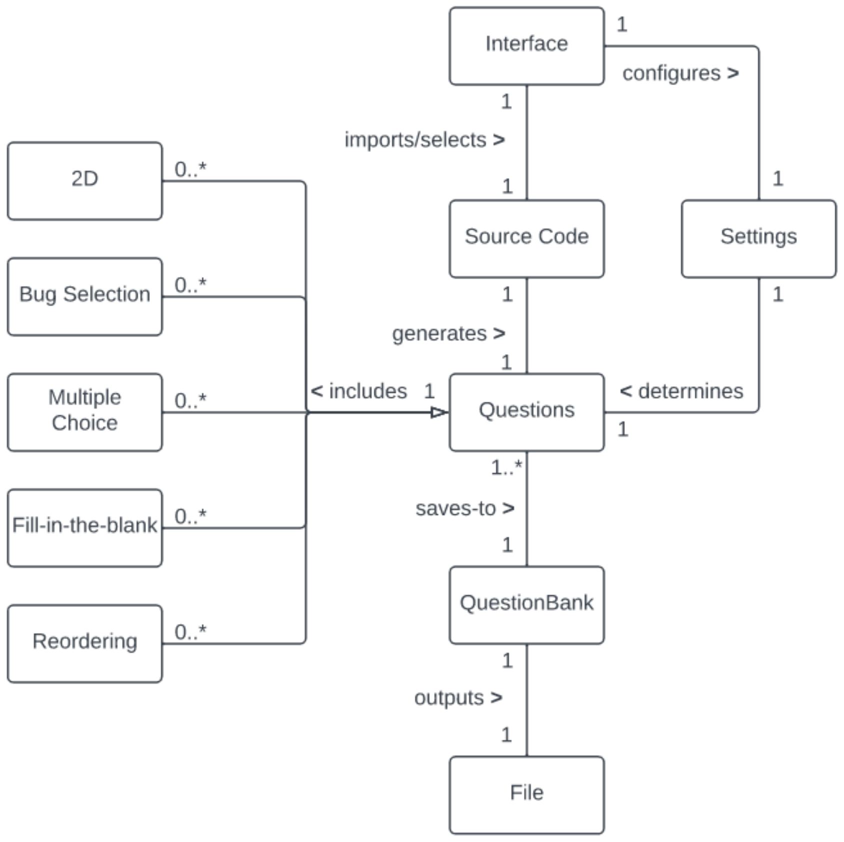
The diagram above is a business process model intended to be used with the context model to demonstrate the interactions between the PPALMS system and those in its operating environment. The user imports the source code into the file system and sets the configuration settings, which consist of three parts: the amount of mutations, the intended LMS target, and the problem type. After configuring the settings, the user will be asked to select lines through the GUI, and the system will determine if the number of lines selected exceeds the recommended line limit. If it does, a warning message shall display and continue to generate problems without performance consideration; if it does not, the system will generate questions and export to .qti file. Once the questions have been exported, the user can then import the .qti file into their intended LMS.

iii. Sequence Diagram



The Instructor shall interact with PPALMS at three distinct points. During the first interaction, the instructor shall specify the location of the source code file using the Graphical User Interface. The controller shall then retrieve a copy of the source code from the file system for later usage. The instructor shall then specify the number of mutations to be generated, the type of problem to be created, and the LMS for which the problems will be created. If any of these settings conflict with the capabilities of PPALMS, an error will be displayed and the system will exit. Finally the instructor shall specify the lines from which the problems will be generated. If the size of the selected lines exceeds the recommended size, the instructor will see a warning but execution shall continue. At this point PPALMS shall generate problems and output them as a .qti file. The exit status of the generation will be shown to the instructor and PPALMS will then exit.

iv. Class Diagram



The system shall feature a graphical user interface (Interface) which allows users to import programming code from a source file (Source Code) and configure settings for problem generation (Settings): maximum number of mutations per problem, intended LMS target, and the type of problem to generate. Both the Source Code and Settings shall be used in creation of the problems (Questions), which can be one of five types (2D, Bug Selection, Multiple Choice, Fill-in-the-blank, Reordering). The number of problems generated is determined by the user and may be zero for a specific type.

After the system creates a set of questions, they shall be saved (QuestionBank) and the user shall be allowed to continue generating additional questions through the Interface. Once the user has finished, the questions shall be output to a .qti file (File) in the user's file system.

Appendix C: To Be Determined List

1. Testing methods of the system.