**Software Requirements Specification**

**for**

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

**Version 0.0.2**

**Prepared by Anthony Narlock, Stephanie Ye, Shen Lua, Jaden Rodriguez**

**Group 2 — CSCI 5801 Section 002 — University of Minnesota**

**October 15th, 2022**

**Table of Contents  
 Table of Contents** ………………………………………………………………..2-3  
 **Revision History** ………………………………………………………………….. 3

**1. Introduction and Description**…….……………………………………… 3

1.1 Purpose ……………………………………………………………. 3

1.2 Document Conventions……….…………………………………… 4

1.3 Intended Audience and Reading Suggestions ……………………...4

1.4 Benefits of PPALMS ……………………………………………….4

1.5 References ………………………………………………………….4

1.6 Product Perspective and Functions……….…………………………4

1.7 Design and Implementation Constraints……….……………………5

1.8 User Documentation……….………………………………………..5

**2. External Requirements** …………………………………………………...6

2.1 User Interface ………………………………………………………6

2.2 Hardware Requirements ……………………………………………6

2.3 External LMS Requirement .……………………………………….6

**3. Use Cases** …………………………………………………………………..7

3.1 Use case Diagram …………………………………………………..7

3.2 Use case #1: Inputting Source Code ………………………………..8

3.3 Use case #2: Selecting the target LMS ……………………………..8

3.4 Use case #3: Selecting a Parson’s problem type ……………………9

3.5 Use case #4: Indicating which sections to annotate ………………..9

3.6 Use case #5: Optional title and description ………………………...10

3.7 Use case #6: Exporting problem to LMS target ……………………10

**4. System Model** ……………………………………………………………...114.1 Context Diagram …………………………………………………...11  
 4.2 Interaction Diagram ………………………………………………..12  
 4.3 Structure Diagram ………………………………………………….13

**5. System Requirements** …………………………………………………….14

5.1 Requirement #1: Inputting source code ……………………………14

5.2 Requirement #2: Selecting a target LMS …………………………..15

5.3 Requirement #3: Selecting a Parson’s problem type ……………15-16

5.4 Requirement #4: Including/Excluding Annotations ……………….16

5.5 Requirement #5: Providing optional title and description …………17

5.6 Requirement #6: Validation for file output ……………………..17-18

5.7 Requirement #7: Generating Permutations ……………………..18-19

**6. Nonfunctional Requirements** …………………………………………….20

6.1 Performance Requirements ………………………………………...20

6.2 Safety and Security Requirements………………………………….20

6.3 Software Quality Attributes………………………………………...20

**Revision History**

| **Version** | **Reason for Changes** | **Date** |
| --- | --- | --- |
| 0.0.1 | N/A - Initial Commit | October 3rd, 2022 |
| 0.0.2 | The document was reorganized to better satisfy IEEE SRS layout and standards after conducting the requirements review. Additions of a context diagram, sequence diagram, structure diagram. Moreover, indications of purpose and audience were made to improve document validity, consistency, and completeness. | October 12th, 2022 |

# **1. Introduction** *1.1 Purpose* This document specifies the software requirements of version 0.0.2 of the Parson’s Problems Appliance for Learning Management Systems (PPALMS). This software requirement specification covers the scope of the PPALMS in its entirety. The specifications require that PPALMS allow users to select a piece of source code from their machine and select from a variety of Parson’s problems types, enable a variety of source code annotation methods for problem creation, generate variations of the selected type from the provided source annotations, and collect the generated questions for export into a chosen Learning Management System.

## *1.2 Document Conventions*

This section’s purpose is to describe the typographical conventions. This document is formatted in Times New Roman 12-point font. Headers of subsections outlined in the Table of Contents are enumerated, enlarged, and italicized. The use cases and system model sections are intentionally put before the system requirements because it is important to understand how to use the PPALMS system before detailing its specifications.

## *1.3 Intended Audience and Reading Suggestions* This PPALMS is intended for instructors who want to create Parsons Type problems quickly. This document serves as a software requirements specification for other PPALMS developers as well as a guide for how the front-end is developed for the use of future backend/LMS developers.

## *1.4 Benefits of PPALMS* The PPALMS must make designing problems and exams easier and must be able to be deployed to learning management systems. It appeals to Universities, Learning Management System businesses, and testing/merit corporations like College Board who specialize in academic testing services. One goal of PPALMS is to make Parson’s Problems more accessible and frequent in academic testing.

## *1.5 References* This document primarily reflects the IEEE SRS Template, abstracting portions that were unnecessary for the PPALMS application. The use cases and requirements labeled in this document were referenced from the Project Description page and the P1 Requirements deliverable page on the University of Minnesota’s canvas page. Images and diagrams represented in this document were prepared by team members.

## *1.6 Product Perspective and Functions* This product is a stand-alone software application meant to streamline Parson’s Problems creation and export. It is meant to act as an interface between a user and their LMS system of choice. Refer to context diagram 4.1 for the context of how PPALMS fits into a larger system.

The core functions of the PPALMS system are to enable annotation and exportation of Parson’s Problems to a target LMS system. User’s should be able to upload source code, choose a target LMS system, choose a problem type, annotate a problem, and export it. Refer to the use case diagram provided in Section 3.1 to observe core functionality, and the following subsections of Section 3 for detailed explanations.

## *1.7 Design and Implementation Constraints* One major limiting constraint will be the problem types supported by each LMS system. Each LMS system has its own supported problem types, and a user must be limited to only be able to annotate problems supported by their target LMS.

## The PPALMS system’s interface language is in English, therefore, the users and developers must be able to speak English. This version also requires developers to implement PPALMS in Java, so both developers and users are expected to have a system which supports the JRE (Version 8 or higher). For developer implementation conversions, the PPALMS application uses standard [Code Conventions for the Java Programming Language](https://www.oracle.com/java/technologies/javase/codeconventions-contents.html).

## The PPALMS system would be expected to have new versions put out regularly as new supported LMS systems are added. Hence, a user may need to update frequently.

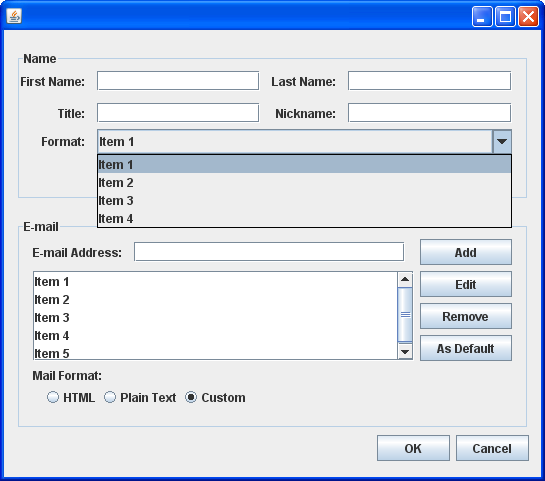
## *1.8 User Documentation* The software is expected to guide the user and be somewhat self-explanatory. That is, text-entry fields and buttons should be labeled, and control flow should be limited to that of what is expected in the general use cases. The software may be delivered with a brief virtual documentation which describes the core functionality and expected UI interactions.

# 

# 

# **2. External Requirements** *2.1 User Interfaces*

The GUI shall have various buttons for different steps in inputting to the PPALMS, one of which is the Upload button for uploading source code. Another button is the Export button for when all required input fields have been filled out and the user is ready to generate the Parson’s Problems and export them to the target LMS. In addition to buttons, GUI shall have drop down menus for selection options at other steps in inputting to the PPALMS. A drop down menu shall be used for selecting the target LMS and also for selecting the Parsons Problem type. Finally, there shall also be text-entry boxes for if the user wants to input an optional title and description for their Parsons Problems and also for where the source code file shall be displayed for annotations to be made. Below is an example image of a very bare bones Java GUI which includes examples for buttons, drop down menu, and text-entry boxes.



A sample user interface utilizing Java Swing

## *2.2 Hardware Requirements* PPALMS shall be able to use source code from the user’s file system. Hence the machine running it shall have some memory-storage apparatus like a hard drive, and a file system. PPALMS shall be implemented in Java, so the machine shall support the Java Runtime Environment (Java 8 or higher). PPALMS shall provide a user-friendly problem annotation interface, so the machine shall have general user peripherals for input like a monitor, keyboard, and mouse. PPALMS shall deploy the designed problems to an LMS, so the machine shall have an internet connection.

## *2.3 External LMS System* The user shall have an intended LMS target to export to in order to use PPALMS. Supported LMS systems are currently to be determined.

# **3. Use Cases** *3.1 Use case diagram*

## 

The use case diagram demonstrates how a user uses the PPALMS application. The user wants to create a Parson’s problem, so they upload source code for their problem. After uploading source code, the user specifies their target LMS. The user then selects their desired type of Parson’s problem. This enables the ability for the user to add optional title and description to their problem. To continue, the user creates annotations for their problem. Once annotations have been finalized by the user, the user can then export their Parson’s problem. The exported Parson’s problem will be used by the target LMS.

*3.2 Use case #1: Inputting Source Code*

**Summary**: The user selects their source code file as input to the PPALMS application (Req 5.1)

**Basic Course of Events**:

1. The user selects the upload file option from the user interface.
2. The user selects their source code file.
3. The user interface displays the uploaded source code and question information form.

**Exception Path**: There may be an instance where the file provided is not a valid programming file. In this case, the user follows steps 1-2 in the basic course of events, but step 3 is replaced with the user interface indicating that the user has selected an invalid file. The user has to re-upload a new source code file to proceed.

**Trigger**: The user has started the application

**Assumptions**: The user has an intended source code file in their file system.

**Precondition**: The user has a file with which they would like to make a Parson’s problem.

**Postcondition**: The source code provided is displayed on the user interface and be used throughout the question form.

**Author**: Anthony Narlock

## *3.3 Use case #2: Selecting a target LMS*

**Summary**: The user selects which LMS the Parson’s Problem will be exported to (Req 5.2)

**Basic Course of Events**:

1. The user has entered a source code file into the user interface
2. The available LMS options are displayed to the user for input
3. The user selects a given LMS

**Exception Path**: None

**Trigger**: The system has verified the user’s source file input

**Assumptions**: The user has an intended LMS target

**Precondition**: The user has provided a valid source code file.

**Postcondition**: The user may proceed with Parson’s Problem annotation.

**Author**: Stephanie Ye

## *3.4 Use case #3: Selecting a Parson’s problem type*

**Summary**: The user specifies the type of Parson’s problem for their given source code (Req 5.3)

**Basic Course of Events**:

1. The user selects a type from a set of Parson’s problem types compatible with their chosen LMS.
2. The user interface displays a problem annotation interface corresponding to the chosen problem type. Under all type selections, the user interface shall have the option to input problem title or description.

**Exception Path**: None

**Trigger**: The system has validated the user’s LMS selection

**Assumptions**: A valid source code file has been inputted to the application. (That is, the use case “Input Source Code” has been successfully completed).

**Precondition**: The source code has been successfully inputted and the target LMS has been chosen.

**Postcondition**: The user interface updates based on the problem type.

**Author**: Anthony Narlock

## *3.5 Use case #4: Indicating which sections to annotate*

**Summary**: To create the Parson’s Problem, the user annotates sections of the code for inclusion/exclusion (Req 5.4)

**Basic Course of Events**:

1. User annotates their problem with the problem annotation interface

**Exception Path**: None

**Trigger**: The user interacts with the problem annotation interface

**Assumptions**: The user has annotations in mind for their Parson’s problem

**Precondition**: The user has selected a problem type

**Postcondition**: The user has the option to indicate problem completion

**Author**: Shen Lua

## *3.6 Use case #5: Providing optional title and description*

**Summary**: The user wants to provide a title and description for their Parson’s Problem (Req 5.5)

**Basic Course of Events**:

1. The user has selected a problem type
2. The user enters a title, description, or both

**Exception Path**: The user does not provide a title, description, nor both

**Trigger**: The user interacts with the interface to enter a title and description

**Assumptions**: The system has acknowledged the user’s problem type selection

**Precondition**: The user has selected their Parsons’s problem type

**Postcondition**: User may proceed to annotate their Parson’s Problem

**Author**: Shen Lua

## *3.7 Use case #6: Exporting problem to LMS target*

**Summary**: The user wants to export their problem to their selected LMS target (Req 5.6 and 5.7)

**Basic Course of Events**:

1. The user has completed selections and annotations
2. The user initiates problem export to target LMS
3. The user is given indication of success by system

**Exception Path**: User does not complete annotations, export is aborted and an error message is displayed saying which fields have not been filled out

**Trigger**: The user indicates that their problem will be exported to a selected LMS.

**Assumptions**: The user has completed annotation

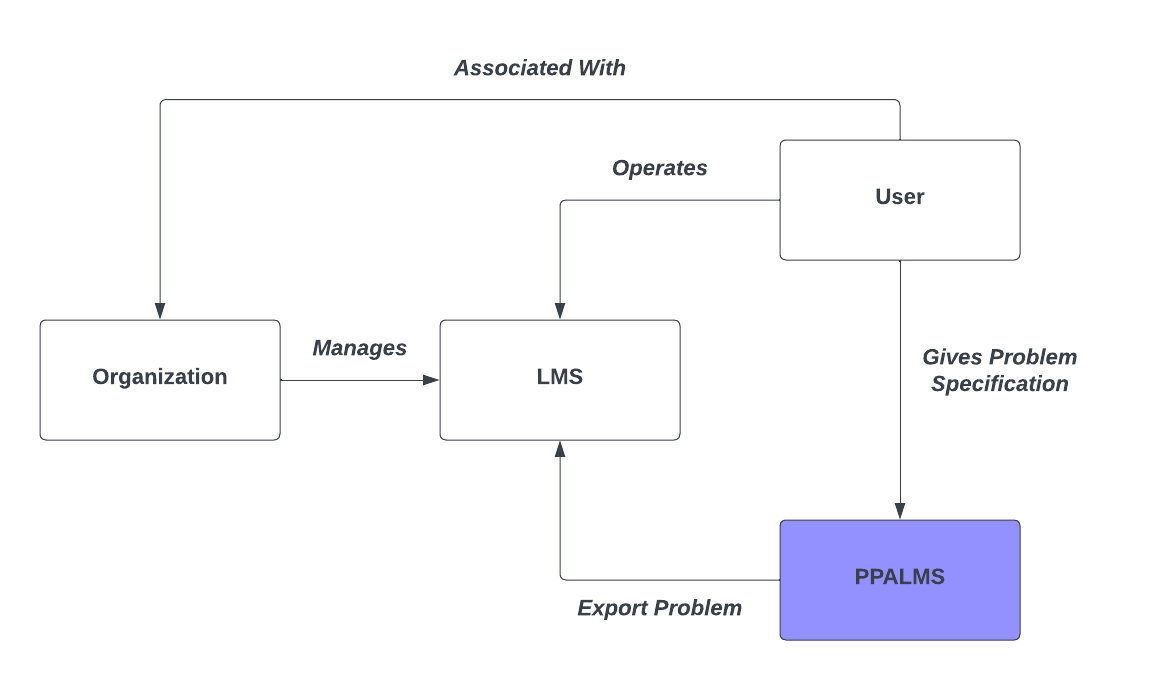
**Precondition**: The user has been allowed to interact with the problem annotation interface.

**Postcondition**: Problem is successfully exported and the system indicates success, or indication of failure if requirements have not been satisfied.

**Author**: Anthony Narlock

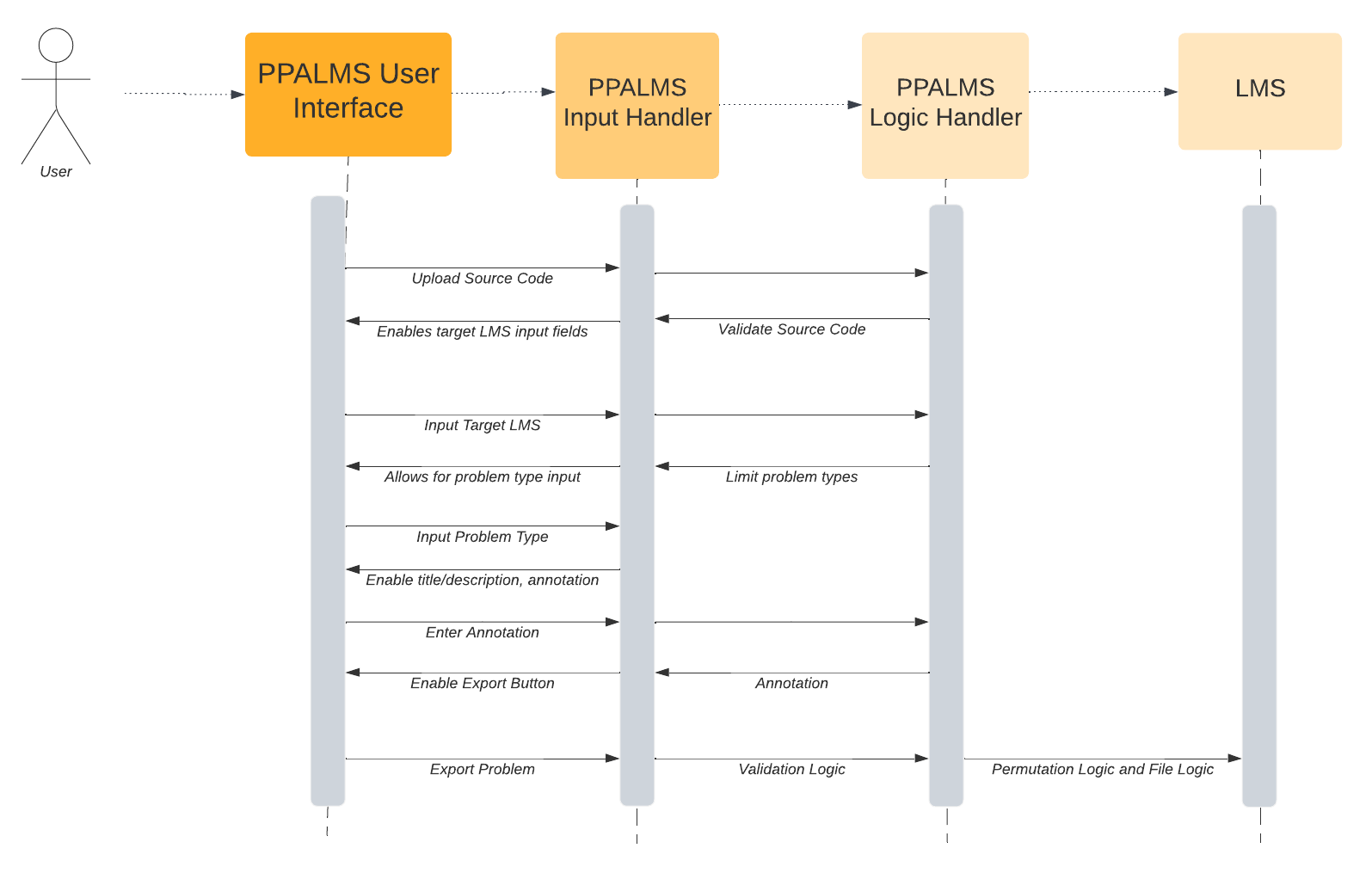
# **4. System Models**

## *4.1 Context Diagram*



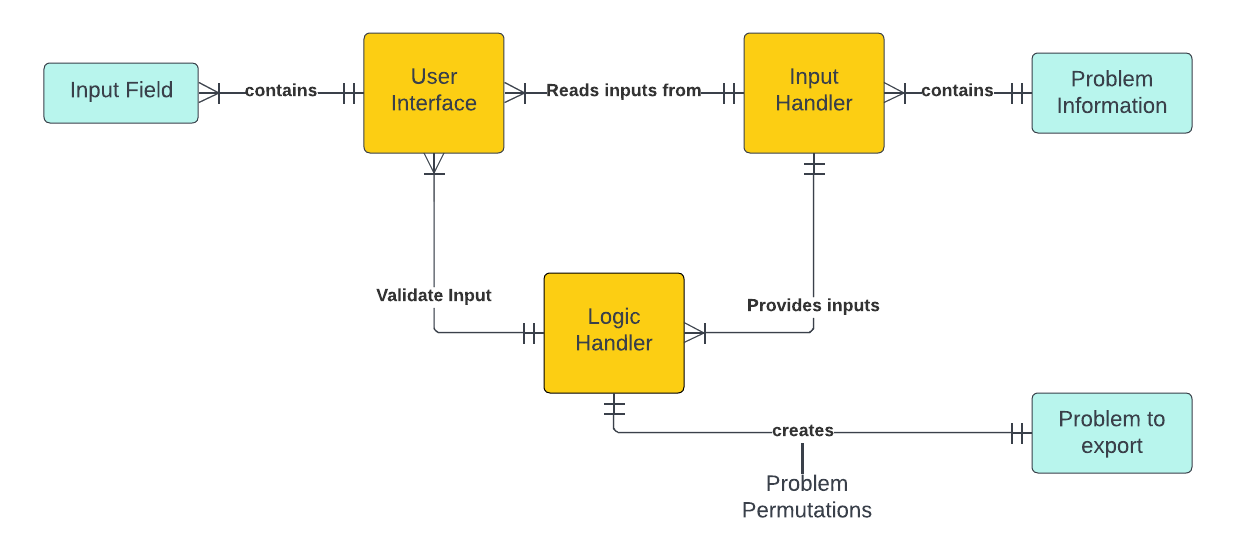
A user leverages the PPALMS application’s purpose to enable Parson’s Problems creation by interaction with its interface to specify problems. The PPALMS application serves its purpose by exporting a user-created problem to an LMS system that the user operates. This context diagram shows the interactions between PPALMS, the user, their chosen LMS system and its associated managing organization.

## *4.2 Interaction Diagram*



The sequence diagram represents the sequence of functions that are performed throughout the usage of the PPALMS system. The process begins by uploading source code through the user interface. The PPALMS system validates the user’s source code and enables the target LMS input field on the user interface. The user selects a target LMS using the user interface. Upon selection, the PPALMS system limits the problem types available for selection in accordance with compatibility of the LMS target. Consequently, the PPALMS system allows the user to select a problem type from those which are compatible. The user selects a problem type using the user interface and the PPALMS system responds by presenting problem editing fields for title, description, and appropriate annotations to the user interface. The user proceeds to apply annotations to their problem through the user interface, which the system obliges and enables the “export” button upon the user finalizing their annotations. Finally, the user clicks on the “export” button, where the problems are validated. Permutations are created and additional file logic is applied. PPALMS then exports the problems for use by the LMS system.

## *4.3 Structure Diagram*



The PPALMS application has three main components: a user interface, which takes manual input from the user, an input handler, which serves the user’s inputs from the user interface, and a logic handler, which serves as a validation and logic service

Users can utilize the user interface’s input fields to provide input to the application. The user interface contains many input fields. The input handler reads the user’s inputs from the user interface and stores them in its problem information entity type. The input handler then delivers the input of the problem information to the logic handler, where the logic handler will validate the input. Upon inputting an export functionality, the logic handler will receive this input from the input handler and create a problem to export. This problem to export contains the problem permutations.

# **5. System Requirements**

## *5.1 Requirement #1: Inputting Source Code*

**Introduction:** In order to generate Parson’s code, source code is provided to the system and is altered by the application.

**Rationale:** Uploading source code directly into the application is faster than typing it out. This also allows the format to be set by the users.

**Authors:** Stephanie Ye

**Inputs:** Source code file

**Requirement Description:** The user shall be able to input a source code file to the application. Once the application reads in a file, it shall present a question information form or error indication

* The application shall accept a file if and only if

1. The file has some text-format file extension
2. The file is between 1 and 50 lines of code

* Upon accepting the file, the system shall present a question information form
* If it does not accept the file, it shall present an indication of error in file format (extension and content)

**Outputs:** If the source code file is inputted successfully, the user interface shall display the question information form where the user can proceed to give information on the inputted problem. If the source code file is not inputted successfully, the user interface shall display a message indicating the error.

**Test Cases:**

* **Success**: User provides valid file and application validates file and information form displays.
* **Failure**: User provides invalid file with incompatible extension, user interface then displays error message indicating wrong file extension.
* **Failure**: User provides file with more than 50 lines, user interface then displays error message indicating too many lines

## *5.2 Requirement #2: Selecting a target LMS*

**Introduction:** Given that one of the functions of PPALMS is to link Parson’s Problems to a LMS, the user shall be able to select their intended LMS target.

**Rationale:** The user wants to export their Parson’s problem to a LMS target. To do this, they need to indicate which LMS they want to send their problem to.

**Authors:** Shen Lua

**Inputs:** Selection from a set of LMS targets

**Requirement Description:** User shall be able to select their intended LMS target

* LMS selection shall be available if and only if:

1. A source code file has been inputted (Req 5.11)

**Outputs:** The LMS selection is displayed to the user interface. The validation status of the LMS selections and other relevant input options are also displayed on the user interface. After an LMS has been chosen, only problem formats which are compatible with the chosen LMS shall be made available.

**Test Cases:**

* **Success Case:** User selects their LMS. The LMS selection is displayed to the user interface with compatible problem types.
* **Failure Case:** User does not select an LMS. The text box for LMS selection continues to display “Select the target LMS”

## *5.3 Requirement #3: Selecting a Parson’s problem type*

**Introduction:** Testmakers require the freedom to select different types of Parson’s problems.

**Rationale:** Freedom to select different types of Parsons Problems allows them to have sufficient variation and reusability.

**Authors:** Shen Lua

**Inputs:** Selection from a set of Parson’s Problem question types

**Requirement Description:** The user shall be able to select the type of Parson’s Problem (matching, ordering, multiple-choice)

* The application shall present a Parson’s problem type for selection if and only if

1. A source code file has been inputted (Req 5.1)
2. The problem type is compatible with the chosen LMS (Req 5.2)

**Outputs:** The selected Parson’s problems type is updated in the user interface. A problem annotation interface corresponding to the chosen type is presented to the user interface.

**Test Cases:**

* **Success Case:** User selects a specific type of Parson’s problems and is presented with the correct corresponding problem annotation interface.
* **Failure Case:** User does not select a Parson’s problem type. The text box for Parson’s problem selection continues to display “Select the Parson’s Problem Type”

## *5.4 Requirement #4: Including/Excluding Annotations*

**Introduction:** The user shall be able to annotate specific sections of the source code for inclusion or exclusion in creating the Parsons Problem.

**Rationale:** The user shall have the option to select which parts of the code they would like to include or exclude in their problem.

**Authors:** Stephanie Ye

**Inputs:** Annotated source code sections

**Requirement Description:** The user shall be able to annotate what section of the source code to include/exclude

* Highlighting sections of the source code is allowed if and only if

1. A source code file has been inputted (Req 5.1)
2. Depending on the type of problem that the user selects, the annotations that are allowed changes accordingly (Req 5.3)
   1. For matching problems and multiple choice problems, the user shall be able to indicate sections they would like to annotate
   2. For ordering problems, the user shall only be able to indicate which lines they would like to annotate

**Output:** Annotated parts of the source code is displayed on the application user interface.

**Test Cases:**

* **Success Case:** The annotated sections are indicated on the user interface
* **Failure Case:** User does not annotate anything in the source code, an error message indicating that nothing has been selected shall be displayed

## *5.5 Requirement #5: Optional title and description*

**Introduction:** Common attributes of any kind of general problem are a title and description. The PPALMS system shall support this as an optional operation.

**Rationale:** Users may desire to describe information about the provided source code to the test takers. This allows users to communicate expectations for a specific problem. It also allows users and test takers to index problems more easily.

**Authors:** Anthony Narlock

**Inputs (Optional):** Problem title, problem description

**Requirement Description:** The user shall have the option to input a title and/or input a description for their annotated Parson’s problem.

* Title and description input appears if and only if

1. User has selected their target LMS (Req 5.2)
2. User has selected their problem type (Req 5.3)

* Any text-based input can be accepted for title and description.
* Title and description are optional. The system shall accept unprovided title and description.

**Output:** The annotated problem is assigned a title or description.

**Test Cases:**

* **Success Case:** User provides no input for either title or description input fields. The program succeeds
* **Success Case**: User provides some text-based input for title. The user interface dynamically updates based on text input.
* **Success Case**: User provides some text-based input for description. The user interface dynamically updates based on text input.
* **Success Case**: User provides some text-based input for title and description. The user interface dynamically updates based on text input.

## *5.6 Requirement #6: Validation for file output*

**Introduction:** The system shall be able to validate and process a problem the user has created and export it to a target LMS.

**Rationale:** The primary purpose of this software is to create and export Parson’s problems. This key feature is a requirement.

**Authors:** Jaden Rodriguez

**Inputs:** Parson’s Problem, Target LMS, (Optional) Title and Description

**Requirement Description:** The system shall be able to validate the problem and output an appropriate file that will be given to the LMS

* This requirement shall do this if and only if

1. The user has chosen a compatible LMS Target to upload to (Req 5.2)
2. The user has specified type of LMS (Req 5.3)
3. The user has indicated an output shall be made with optional title and description (Req 5.5)

* The output file shall follow a format supported by the chosen LMS
* The output of Req 5 shall be incorporated into the output file
* If the requirements are not met, the user shall be notified of which requirements they have not satisfied

**Outputs:** A file describing a valid Parson’s problem

**Test Cases:**

* **Success**: User provides valid Parson’s problem and target LMS; a problem is successfully exported to target LMS.
* **Failure**: User provides valid Parson’s problem but no target LMS; an error message indicating the user shall choose a target LMS is displayed
* **Failure**: User provides target LMS but not a valid Parson’s problem; an error message indicating the parts the user has not completed for validation.
* **Failure**: User provides no target LMS nor a valid Parson’s problem; an error message indicating the parts the user has not completed for validation.

## *5.7 Requirement #7: Generating Permutations*

**Introduction:** Providing different permutations of the answer choices for the same question is essential to providing variation among test takers.

**Rationale:** This requirement allows users to provide a layer of examination integrity while also providing reuse of problems.

**Author:** Anthony Narlock

**Inputs:** Integer representing number of requested permutations for a validated problem (not necessarily by the user, could be implicit by their design choice)

**Requirement Description:** The application shall generate unique permutations of the annotated problem.

* A maximal set of permutations for an input problem shall be constructed if and only if:

1. The system has validated the user’s annotations (Req 5.6)

* The number of permutations of choices shall be limited to 30 to avoid gross memory usage

**Outputs:** This shall simply be something which occurs upon completion before export.

**Test Cases:**

* **Success Case:** User has completed annotation and the system has validated their problem. We expect the system to allow the user to proceed.
* **Failure Case**: If prerequisites are not completed or invalid inputs are entered, the user shall not be able to export and the user interface displays a message to indicate the problem is not annotated
* **Failure Case:** Permutations are generated incorrectly. Users need to double check if the generated Parson’s Problems are what they wanted. It could be outside the limitations of the PPALMS, or could be an error with what they selected for the annotations.

# **6. Nonfunctional Requirements** *6.1 Performance Requirements*

The expectation is that the PPALMS system is to run on hardware described in Section 2.2, so the primary component of performance will lie in the user’s end system. For general user interactions, such as uploading source code and filling out the general form (problem type, target LMS, annotations), the system should respond from this interaction with a delay no longer than 1 second per action. Upon exporting the problem, the system will calculate permutations for each problem. Depending on the speed of the end user’s system, performance may vary. However, creating the permutations is the most costly function of the application, therefore may result in a loading phase that is dependent on the user’s hardware.

# *6.2 Safety and Security Requirements*

Since the PPALMS application will generate the finalized exported problem including permutations, there is loss in data if the application was interrupted (i.e., computer interrupt/crash or user exits the program before completion). If this happens, the user will be required to begin entering their problem from the beginning state, as no internal data of the problem will be stored.

The application exports generated problems to the target LMS. This means that the target LMS is responsible for providing security on data within the problems. This leaves organizations that manage the LMS to decide what to do with the data exported from the system.

# *6.3 Software Quality Attributes*

One of the primary functions of PPALMS is to filter problem types according to a user’s target LMS system and export problems to the target LMS System. Hence, the need to be able to add new LMS targets would be frequent, thus the software must be adaptable and reusable, and perhaps most importantly open to extension.

Another primary function of PPALMS is to make Parson’s Problem creation accessible to a general user. Hence the software needs to be very usable to a general user (even those without development knowledge).