**Requirement #**1

**Introduction:** In order for Parson’s code to be generated, the system needs to be provided with source code that will be 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. It is also an opportunity for input filtering/sanitization.

**Authors:** Stephanie Ye, Anthony Narlock, Shen Lua, Jaden Rodriguez

**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 fill , it should proceed by presenting 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 will present a question information form
* If it does not accept the file, it will present an indication of error in file format (extension and content)

**Outputs:** If the source code file is inputted successfully, the user interface will 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 will display a message indicating the error.

**Test Cases:**

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

**Requirement #**2

**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 will need to indicate which LMS they want to send their problem to.

**Authors:** Shen Lua, Anthony Narlock, Stephanie Ye, Jaden Rodriguez

**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 1)

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

**Test Cases:**

* **Success Case:** User selects their LMS. The LMS selection is displayed to the user interface with compatible problem types.

**Requirement #**3

**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, Anthony Narlock, Stephanie Ye, Jaden Rodriguez

**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 will present a Parson’s problem type for selection

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

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

**Requirement #**4

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

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

**Authors:** Stephanie Ye, Anthony Narlock, Shen Lua, Jaden Rodriguez

**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 1)
2. Depending on the type of problem that the user selects, the annotations that are allowed will change accordingly (Req 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 will display 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

**Requirement #**5

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

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

**Authors:** Anthony Narlock, Shen Lua, Stephanie Ye, Jaden Rodriguez

**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 will appear if and only if

1. User has selected their target LMS (Req 2)
2. User has selected their problem type (Req 3)

* Any text-based input can be accepted for title and description.
* Title and description are optional. The system will 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 will succeed.
* **Success Case**: User has provided some text-based input for title. The user interface will dynamically update based on text input.
* **Success Case**: User has provided some text-based input for description. The user interface will dynamically update based on text input.
* **Success Case**: User has provided some text-based input for title and description. The user interface will dynamically update based on text input.

**Requirement #**6

**Introduction:** The system should 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, Anthony Narlock, Stephanie Ye, Shen Lua

**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 2)
2. The user has specified type of LMS (Req 3)
3. The user has indicated an output should be made with optional title and description (Req 5)

* The output file should follow a format supported by the chosen LMS
* The output of Rdeq 5 should be incorporated into the output file
* If the requirements are not met, the user should 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 should 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.

**Requirement #**6

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

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

**Author:** Anthony Narlock, Stephanie Ye, Shen Lua, Jaden Rodriguez

**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 before

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

**Outputs:** This should simply be something which occurs upon completion before export. Under invalid input (failed to complete prerequisites), the user will not be able to proceed to export and the user interface will display a message to indicate the problem is not annotated.

**Test Cases:**

* **Success Case:** User has completed annotation and the system has validated their problem. We will expect the system to allow the user to proceed.