Software Requirements Specification

for

Parson's Problems Appliance for Learning Management Systems

Version 2.0 approved

Prepared by Aidan Mahlberg, Hannah Cheng,

Oliver Barbeau, Zhijie Xu

Group 10

10/18/2022

Table of Contents

1. Introduction

1.1 Product Purpose

Generally to be used by instructors that distribute material to their students by a LMS (Learning Management System). This product is a program that allows for users to upload code sample files, read it as text and produce variations of Parson's problems and return those in a file to the instructor that they can use with a LMS platform. Parsons problems are used to help new programmers understand how to write code by manipulating the arrangement of code lines. Parson's problems task students to select the arrangement of code lines, and in this product code "blocks".

1.2 Product Scope

The software was invented for the purpose of making the quiz or exam questions efficient. The software takes in a user uploaded file, reads it and generates different variations of questions based on the given file. Instructors will benefit from this software because it saves time and is easy to use. To demonstrate, instructors can upload any code sample file to the software. After the code sample file is being read, instructors can change the questions type based on their choices. For example, if the instructors want certain questions to be short answer questions, they could change the original format that was provided by the software to other formats and they can also do any annotations as they expect. The produced question files can be stored in the local or cloud server. That feature supports the reuse of the produced question files. It will be convenient for instructors to produce and edit the questions, which can be directly implemented in the target LMS platforms.

2. Product Description

2.1 Product Functions

- The system supports uploading any code file, and produces a question bank.
- The system supports annotation and editing on the produced question.
- The system supports the selection of question types(Matching/Multiple choice/Ordering).
- The system supports generation of all possible variations of code matching the question type, and limits the number of variations on demand.
- The system supports potential reuse of all generated questions, an output file that can be stored.

- The system supports the output of question bank files appropriate for different LMSs.

2.2 User Classes and Characteristics

The main intended user class of this product is instructors that are teaching their students a programming language. Users could be professors, teaching assistants, or any person that intends to use a quiz bank to generate quiz questions in the format of Parson's Problems for LMS platforms. The user should have an intended LMS quiz-format as the target file output. A notable side-effect of unintended use by this user would be: If the user (ex. an instructor) uses this program on a code file with bugs, the answer of the questions produced will have that exact code, present in the quiz questions produced.

2.3 Operating Environment

The software does not provide public access to downloads, and only verified registrants are allowed to use this software. It is guaranteed to run on the user's personal computer, the official computers in schools or educational institutions, and work computers in companies. It can run on major operating systems that are still officially supported, including Windows and Linux. It must not be blocked or removed by the security management software that comes with different operating systems.

2.4 Design and Implementation Constraints

Generating variations of an unlimited number of blocks of code leads to an enormous number of variations, n lines, n factorial number of block variations. The user needs to be limited to inputting a set number of blocks to form these variations within reasonable bounds. For the sake of not reaching these expensive cases, we limit this number to 16 (Req #08). Within the reasonable expectations of an instructor user to have 16 answers to a quiz question.

2.5 Assumptions and Dependencies

- This document defaults to the user having obtained an official license.
- It is assumed that the file provided is of a working type, and is intended to be read as a text file. This file is assumed to be a working program of any standard programming language, decided by the instructor. The program file must be able to be read as text characters, which can be read by the program in that generalized format.
- When the software is running, the software is not interrupted by other external factors, including unexpected power failure, physical damage to computer, system bugs and software viruses.

- The type of output file meets the requirements of the target LMS platforms. The types of LSM platforms acceptable files do not change or are not updated when using the latest version of the software.

3. External Interface Requirements

3.1 User Interfaces

- The user-oriented software interface consists of three parts: Status bar, toolbar and main area.
    - The far right side of the status bar contains three buttons to control minimizing, zooming in and out, and closing the interface.
    - The toolbar includes two menus: "File" and "Help".
        - "File" includes two options "Open" and "Save" to help users reuse the previously stored files and save the produced question file.
        - "Help" includes "Guide" and"Update" to help users to get instructions and get the latest version of the software.
    - The main area include one button"Upload"
        - By clicking the button, users can choose the local code file.
        - If uploading successfully, there will be a text box showing "Upload Successful". Otherwise, there will be a text box showing "Upload Failed".
- After uploading, there is the second main area in the interface, which provides the options for users to limit the total number of variations and the number of different types. After those options, users click the button "Run" and the software will automatically produce the new questions under users' limit.
- After the new questions are generated, users can browse each question. There is the third main area, which offers tools for users to edit the questions. Users can add annotations to each question and edit the contents of questions. After those steps, there is a button "Choose Platforms" to go into the next main area.
- In the last main area, there will be different buttons representing mainstream LMS platforms. After clicking the targeted LAM platform button, users can get the produced question file with appropriate format and store it into their local address or cloud server.

3.2 Hardware Interfaces

Hardware of the computer should not be less than ten years old and is compatible with Windows and Linux systems, which are still officially supported and maintained. The

software completes the entire process with computer cursor clicks, which can be done with the computer mouse/touchpad/touchscreen.

4. Primary System Feature: Produce Parson's Problems Variation from File Input

4.1 Description and Priority

The focus of this project is on this function. A user inputs a file that can be interpreted as text; this process allows the user to select blocks of code whose order will not be changed in context to one another; create variations of Parson's problems quiz questions from this file by the user's specification, and return a file containing those questions in the user specified format.

4.2 Stimulus/Response Sequences

1. Users select a code file.
2. Users will choose generation options, annotate their file, and choose output format , outlined in 4.3 Req 2 - 4.3 Req 5.
3. An output file will be produced for the user, a quiz bank file for their LMS of choice.

4.3 Functional Requirements

Req 1. Users upload (code) files that can be read and interpreted as text, one or many
   a. Output: File of specified target LMS question bank file type.

Req 2. Users can annotate generated questions, attaching a message to each question variation produced from that question generation.

Req 3. Users can group or"block" code lines together so in reordering of code samples, those selected blocks will have the same order, in reference to the lines of that block. That block will be moved as one unit when the code is then used to produce a problem variation.

Req 4. Users can choose from 3 different types, and any combination of those types to be produced as questions:
   a. Matching: Fill in the blank with the correct code
   b. Multiple Choice: Choose the missing line
   c. Ordering: Choose the order of the lines

Req 5. User will choose how many blocks of the input file to scramble.

Req 6. Users will choose a particular number of answers to be associated with a question that is produced in a multiple choice format.
   a. LMS platforms may have a limit on the number of answers to a multiple choice question, (4/5/a finite number)

Req 7. Output LMS file type: QTI, GIFT, XLXS, AICC

Req 8. From file input, the number of variations of questions produced will be limited by allowing no more than 16 blocks to be input after user file parsing.

Req 9. Windows MDS or Linux operating system needed to run this project's program. OS has been updated past primary distribution within 5 years.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Req 1. Software should return the question file in under 1 minute (user file is at maximum of 1 GB/gigabyte). If the uploaded file exceeds the maximum file limit, the software shall return the error message to users.

5.2 Security Requirements

Req 1. When using/logging in the software, the software will ask for the instructor ID and password. If the entered password is incorrect three times, software will ask the user to reset their password and send a link to their email that was used to register the software account.
Req2. The password should be at least 15 characters long, contain at least one number, at least one capitalized character, at least one special character (i.e.#, $, %).
Req3. The text file uploading software must not be compromisable by a student or a third party. The original file will not be accessible by anyone other than the instructor who uploaded it.

5.3 Business Rules

All instructors, teacher assistants and students have access to view the quizzes, but at certain times. For instructors, they are able to view the quizzes at the beginning since they upload the file and edit the quizzes format. For teacher assistants, they are able to view the quizzes after the quizzes are generated and edited by the instructor. Students are able to view and take the quizzes after it is published by the instructor or teacher assistants. Students do not have access to edit the quizzes or view the quizzes before publishing. Teacher assistants do not have access to edit the quizzes.

6. Other Requirements

6.1 Legal Requirements

Req1. The source code imported into the software must come from legitimate sources, including user-generated code, licensed code, and open source code.
Req2. The exported files must only be used on certified and legitimate LMS platforms by local government and laws.

6.2 Reuse Requirements

Req1. When users get new questions from the system, generated questions will be stored locally and there will be an option for users to store them to the cloud server.
Req2. When users try to create a new quiz or exam, they can quickly call the stored questions from local or cloud.

7. Functional Requirement Specifications

Requirement #: 01 **Requirements Type:** Functional **Use Case:** 1 **Date:** 10/16/2022

Shorthand: *Sec 4.3 Req 1:* Users select (code) files that can be read and interpreted as text, one or many

Output: File of specified target LMS question bank file type.

Introduction: The core use case of this system relies on the user being able to select a code file and generate Parson's problems quiz questions for their target LMS in the format specified by that LMS.

Rationale: We can generalize a code file to any file that can be interpreted as text, covering all programming languages in the process.

Author: Oliver Barbeau
Source: Elicitation Session 1
Inputs:

1. A (code) file that can be interpreted as text.
2. User required and optional selections

Requirement Description: The user shall be able to select a code file to generate parson's problem questions.

User must:

1. Select output file format for target LMS (Req # 07)
2. Select one to all types of quiz questions to generate (Req #04)

3.

User can:

1. Annotate the code file, as a message or hint to be displayed alongside the question produced. (Req #02)
2. Select lines of their input file to be grouped together as "blocks". Each line is otherwise interpreted as a block of one line. (Req #05)
3. Select number of code blocks to rearrange at a maximum. (default of 4, max 16) (Req #08)

From these selections and the input file, the generation algorithm can operate:

1. If the user selected to generate Matching question type:
    a. Produce questions where the missing fill in the blank answer is exact text of a code block
2. If the user selected to generate Multiple Choice question type:
    a. Produce multiple choice questions where the answer to a given question is the correct arrangement of blocks according to the input file. The potential answers for a question are blocks selected from the file.
3. If the user selected to generate Ordering question type:
    a. Up to the number of maximum answers M, take M blocks of the input file and permute arrangements of these lines. The correct answer to this type of question is the correct arrangement of code blocks matching the original input file.

After Questions are generated, they will be formatted according to the specifications of the target LMS file-format, with a formatting algorithm according to each particular format.

Outputs: Quizbank of questions in the target file format selected by users.

Persistent Changes: None

Related Requirements: Req #02, #03, #04, #05, #06, #07

Conflicts: None

Support Materials: Resources and references in the appendices of this document.

Test Cases: TBD

Requirement #: 02 **Requirements Type:** Functional **Use Case:** 1 **Date:** 10/16/2022

Shorthand: *Sec 4.3 Req 2:* Users can annotate generated questions, attaching a message to each question variation produced from that question generation.

Introduction: The software supports the annotation function. Users can add annotations to the generated questions in every position for each question. The annotation can be a hint for the generated question before answering or an explanation of the correct answer after answering. This will help the students to understand and master the programming.

Rationale: System feature

Author: Zhijie Xu

Source: Elicitation Session 1

Inputs: The generated questions after the software process. The questions include all variations of the original code file as the users set. The all generated questions will be shown in the main area of the interface.

Requirement Description:

1. Users have access to all generated questions in the interface.
2. After the users get into one of the questions, users can select specific parts of the question by clicking on them.After that, there will be two options for users to choose. One option is "Before answering" and the other is "After answering".
3. After choosing the expected option, there will be an input box for users to enter their annotations.
    a. If choosing "Before answering", the annotations will be displayed before answering the question.
    b. If choosing "After answering", the annotations will be displayed after answering the question.

Outputs: A question with annotations in a specific order as users expect

Persistent Changes: None

Related Requirements: Req #01, #02

Conflicts: None

Support Materials: Resources and references in the appendices of this document.

Test Cases: TBD

Requirement #: 03 **Requirements Type:** Functional **Use Case:** 1 **Date:** 10/16/2022

Shorthand: *Sec 4.3 Req 3:* Users can "block" code lines together so in reordering of code samples, those selected lines will have the same order, in reference to the lines of that block. That block will be moved as one unit when the code is then used to produce a problem variation.

Introduction: After the questions are generated, users have access to all generated questions. Users can edit the codes in the questions to meet their expectations. Users can block the code lines together and move them into any place users expect. The order in the selected block does not change. This will help users to handle the question in a more convenient way.

Rationale: System feature

Author: Zhijie Xu

Source: Elicitation Session 1

Inputs: The generated questions after the software process. The questions include all variations of the original code file as the users set. The all generated questions will be shown in the main area of the interface.

Requirement Description:

1. Users have access to all generated questions in the interface.
2. After the users get into one of the questions, users can select a specific block with lines together.
3. Users can move the block to every place. The order in the block does not change.

Outputs: A question with order of some codes changing as users expect

Persistent Changes: None

Related Requirements: Req #01, #03

Conflicts: None

Support Materials: Resources and references in the appendices of this document.

Test Cases: TBD

Requirement #: 04 **Requirements Type:** Functional **Use Case:** 1 **Date:** 10/16/2022

Shorthand: *Sec 4.3 Req 4:* Users can choose from 3 different types, and any combination of those types to be produced as questions:

1. Matching: Fill in the blank with the correct code
2. Multiple Choice: Choose the missing line
3. Ordering: Choose the order of the lines

Introduction: There are a few types of questions that should be produced. Each question type has slightly differing, and overlapping constraints.

Rationale: None

Author: Oliver Barbeau

Source: Elicitation Session 1

Inputs:

1. Choice, one to many:
    a. Matching
    b. Multiple Choice
    c. Ordering

Requirement Description:

1. Users can choose any given question format options
2. after choosing the options, the system will generate a question bank based on options selected and return the question bank to the user

Outputs: a question bank with selected question format
Persistent Changes: None

Related Requirements: Req #02, #04

Conflicts: None

Support Materials: Resources and references in the appendices of this document.

Test Cases: TBD

Requirement #: 05 **Requirements Type:** Functional **Use Case:** 1 **Date:** 10/16/2022

Shorthand: *Sec 4.3 Req 5:* Choose how many blocks of the input file to scramble

Introduction: Making permutations (n^2) is an expensive computation with large file sizes.

Rationale: Most LMS platforms wont have options for matching past 10, and it is unlikely that an instructor user would want more than that many options for any question type.

Author: Oliver Barbeau
Source: Project Assignment 1
Inputs:

1. Integer number 4-16

Requirement Description: To restrict the question permutations made from a single code file, the system shall limit the number of quiz answers, and thus permutations that need to be made on a single file.

Outputs: Operation parameter - Req #01
Persistent Changes: None

Related Requirements: Req #01, #05

Conflicts: None

Test Cases: TBD

Requirement #: 06 **Requirements Type:** Functional **Use Case:** 1 **Date:** 10/16/2022

Shorthand: *Sec 4.3 Req 6:* Users will choose a particular number of answers to be

associated with a question that is produced in a multiple choice format.

Introduction: Software supports the function of choosing a particular number of answers that are associated to the multiple questions that are generated from the software.

Rationale:

Author: Hannah Cheng
Source: Elicitation Session 1
Inputs:

1. one of the generated multiple choice questions

Requirement Description:

1. LMS platforms may have a limit on the number of answers to a multiple choice question, the system shall limit the answers to each multiple choice question.
2. the user can choose a particular number of answers, the system will have notification when the number of questions is at maximum.

Outputs: multiple choice question with number of answers with it
Persistent Changes: None

Related Requirements: Req #01, #02, #04

Conflicts: None

Support Materials: Resources and references in the appendices of this document.

Test Cases: TBD

Requirement #: 07 **Requirements Type:** Functional **Use Case:** 1 **Date:** 10/16/2022

Shorthand: *Sec 4.3 Req 7:* Output LMS file type: QTI, GIFT, XLXS, AICC

Introduction: One goal of this system is to generate the question file types that correspond to different LMS platforms. This will be convenient for users to create questions or exams of Parson's problem.

Rationale: The different LMS platforms have different acceptable file types.

Author: Zhijie Xu
Source: Elicitation Session 1
Inputs: A (code) file that can be interpreted as text.

Requirement Description:

1. Users can choose a code file, which can be converted into text type. This type is readable by the software.
2. After the process of generating the expected questions types, there will be different options for users to choose the target LMS platforms.
3. After choosing the required platforms, the software will produce the question file of the corresponding type.

Outputs: A question file which meets the uploading requirements of the selected LMS platform. The type of the question file will be one of those types: QTI, GIFT, XLXS, AICC
Persistent Changes: None

Related Requirements: Req #01, #07

Conflicts: None

Support Materials: Resources and references in the appendices of this document.

Test Cases: TBD

Requirement #: 08 **Requirements Type:** Functional **Use Case:** 1 **Date:** 10/16/2022

Shorthand: *Sec 4.3 Req 8:* From file input, the number of variations of question produced will be limited by allowing no more than 16 blocks to be input after user file parsing.

Introduction: Generating variations of an unlimited number of blocks of code leads to an enormous number of variations. For the goal of this system, it is used to be convenient for users and helpful for people who answer that. We limit the number to 16. Every question has at most 16 options to reach an answer.

Rationale: Implement constraints

Author: Zhijie Xu
Source: Elicitation Session 1
Inputs: A (code) file that can be interpreted as text.

Requirement Description:

1. Users can choose a code file, which can be converted into text type. This type is readable by the software.
2. Users can limit the number of variations and blocks of every question before generating the questions variations.

Outputs: A question file with questions having a bounded number of options and variations.
Persistent Changes: None

Related Requirements: Req #01, #08

Conflicts: None

Support Materials: Resources and references in the appendices of this document.

Test Cases: TBD

Requirement #: 08 **Requirements Type:** Functional **Use Case:** 1 **Date:** 10/16/2022

Shorthand: *Sec 4.3 Req 9:* Windows MDS or Linux operating system needed to run this program. OS has been updated past primary distribution within 5 years.

Introduction: to use this software, the hardware operating system should be either Windows MDS or Linux and should be updated within five years.

Rationale: None

Author: Hannah Cheng

Source: Elicitation Session 1

Inputs: eligible hardware and operating system

Requirement Description:

1. hardware that is used for the software must either have Windows MDS or Linux operating system, and should be updated within five years.

Outputs: None

Persistent Changes: None

Related Requirements: Req #07

Conflicts: None

Test Cases: TBD

Appendix A: Glossary

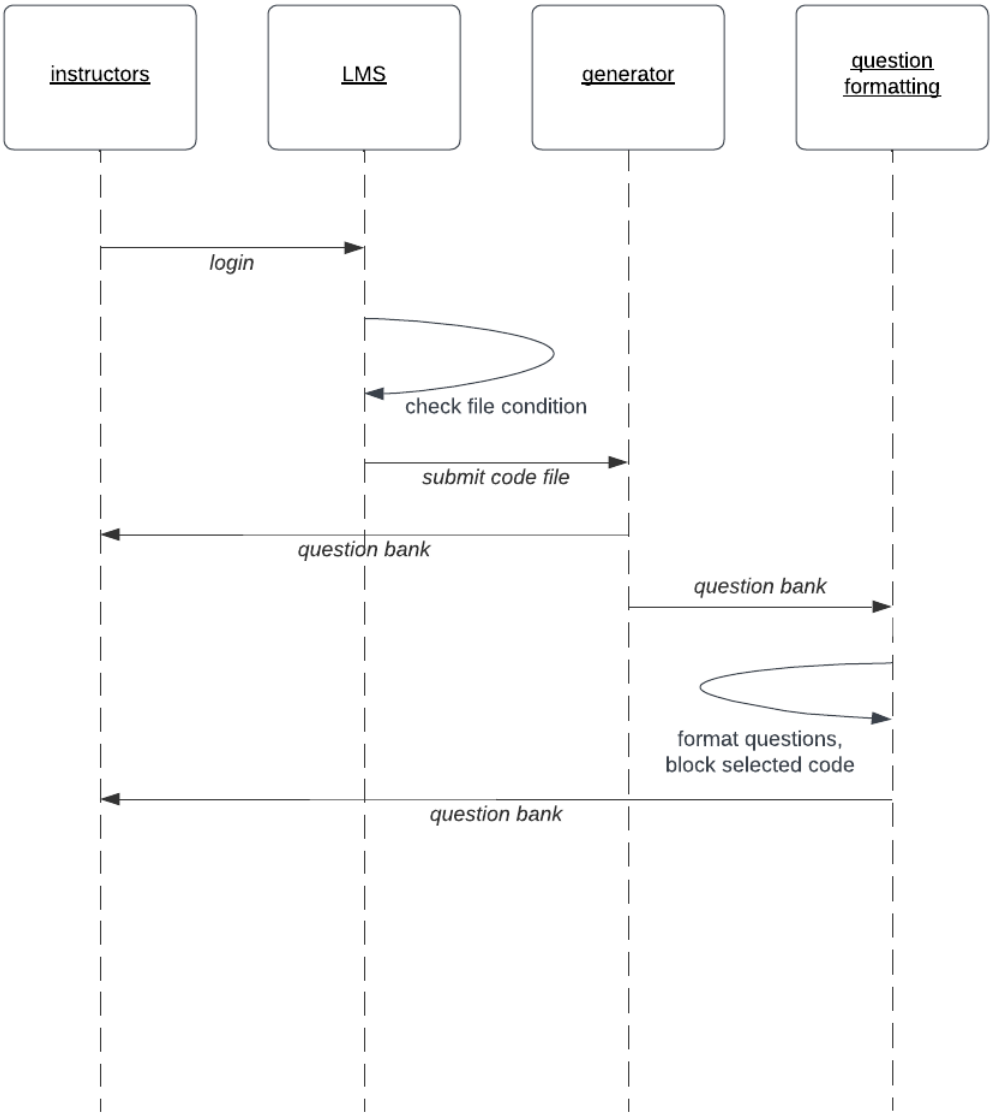
- PPALMS - The acronym of Parson's Problem Appliance for Learning Management System
- LMS - The acronym of Learning Management System

Appendix B: Reference Links

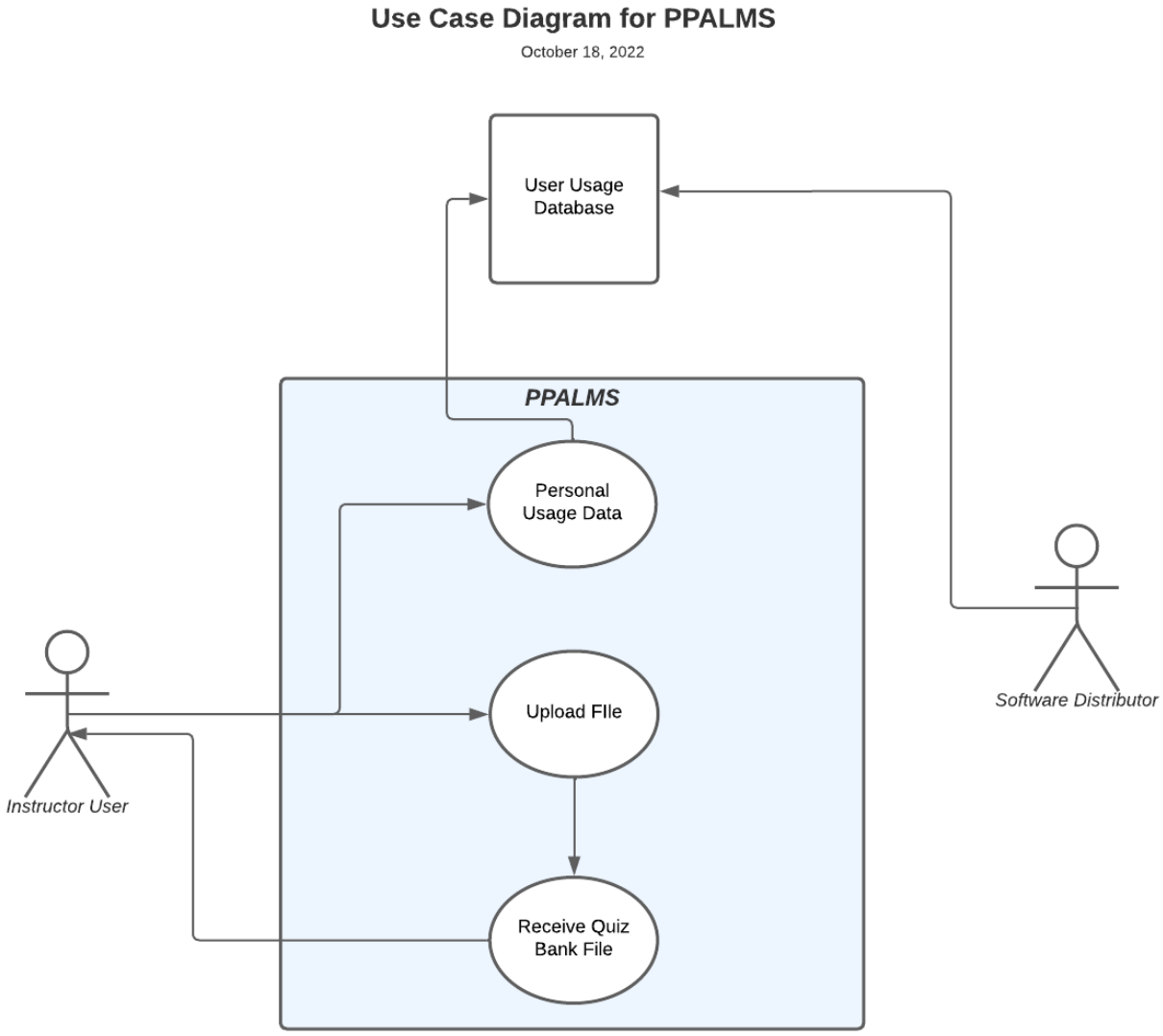
- Parsons Puzzle http://parsons.problemsolving.io/
- https://en.wikipedia.org/wiki/Parsons_problems
- Original Paper by Dale Parson
  https://dl.acm.org/doi/10.5555/1151869.1151890

Appendix C: Models and Diagrams

Interaction Diagram(Sequence Diagram)



Descriptions: Instructor users have two sets of interactions that matter to the scope of this project: With our system and with their LMS. The product is related to the LMS that the instructor user uses by a necessity to match file format. In this diagram our product is labeled "generator" The generator does not directly interact with the LMS, but through the instructor uploading the file produced by this product's primary feature. Instructors select a file to be used by the generator, along with their required and optional selections, to produce an output quiz bank file in a format that the instructor user's target LMS can interpret. The generator makes the questions, while the question formatting process gets those questions into the file format of the target file format.

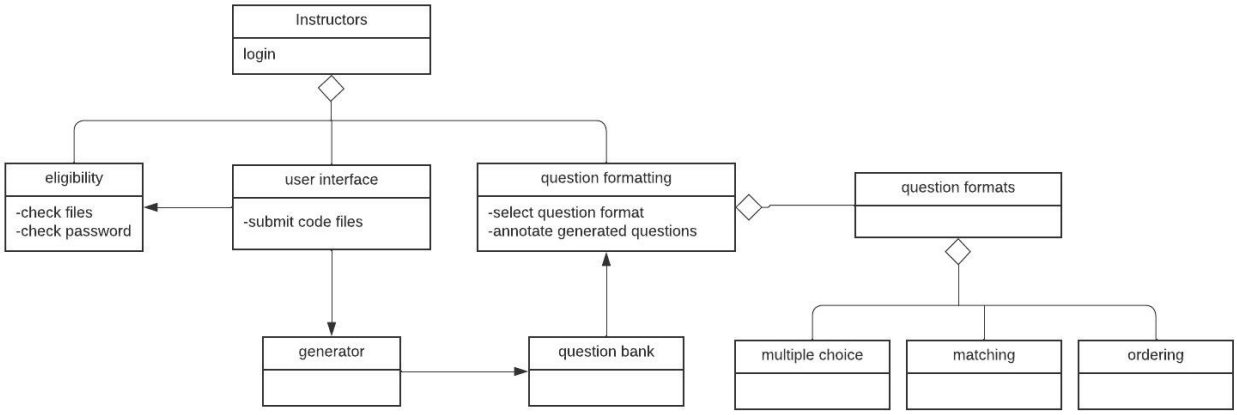


Descriptions: There are two individuals that can interact with the system.

Basic users/instructors are able to upload a file and have returned a quiz bank file, as well as view their own personal usage data, such as: type of questions they have previously generated, how long they have been using PPALMS, and which code files they have previously submitted.

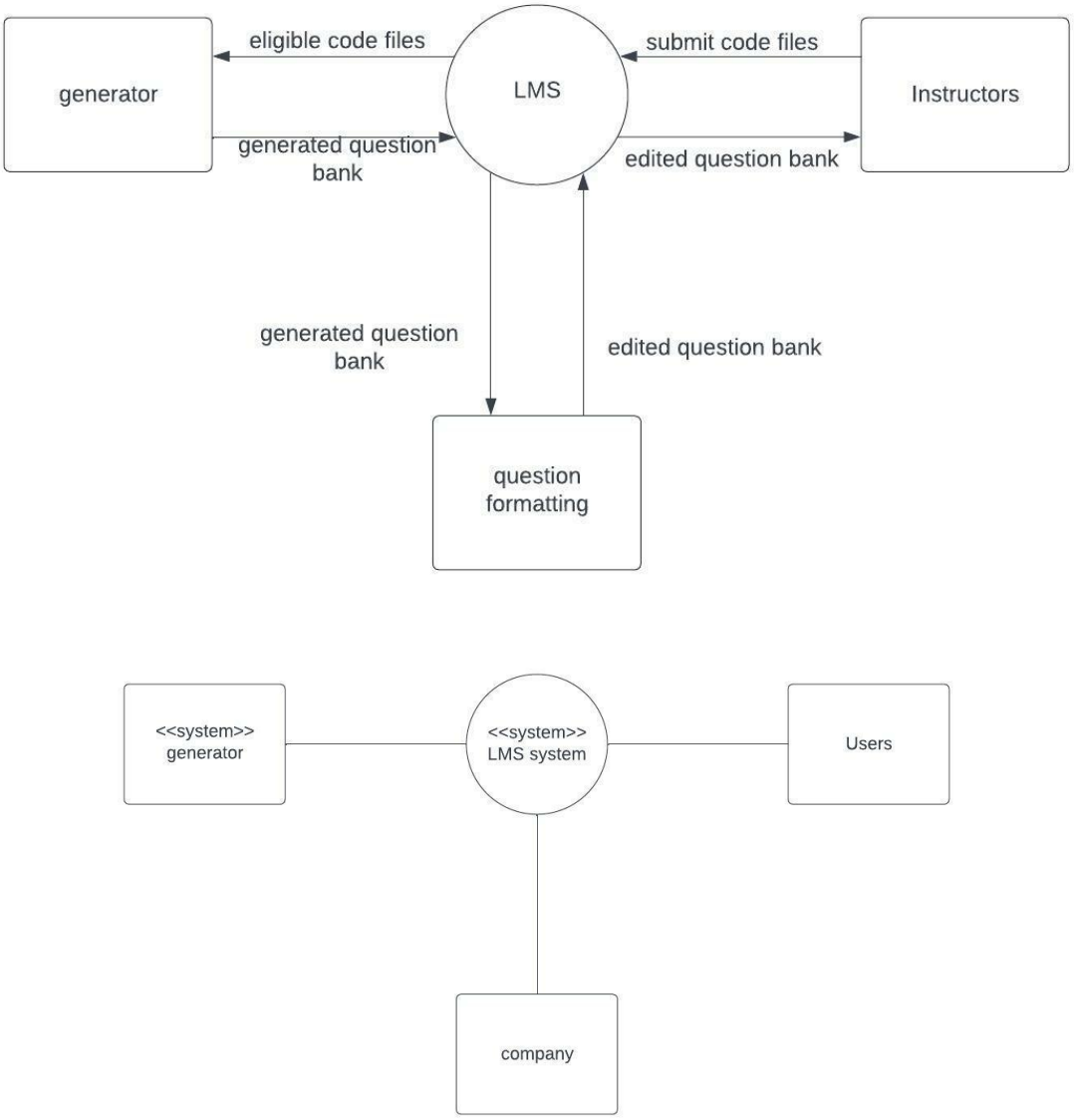
The software distributor is able to access usage data for all users.

Structure Diagram



Descriptions: This model is the class model for the software. When users log into the system, it will check if the passwords match. After logging in successfully, users can submit the code files to the software system. After the files are checked eligible, the system generates a question bank based on the files. Users can either choose to keep the question bank that is generated, or annotate and edit the question banks. Users are able to edit and annotate multiple times and the system will save the edit history. Users are able to choose from three question formats that are multiple choice, matching, and ordering. They are able to have a combination of the formats. After choosing the formats, the system will generate a new question bank based on the selected formats.

Context Diagram



The first is an internal relation diagram for the system. The second is the context diagram for our system. Users have access to the system. They can upload a code file and get the corresponding LMS platforms question file of Parson's problem. The company represents the provider of the software, and they are responsible for authorization, maintenance upgrades, providing data storage and analysis. Users can directly upload and use the generated question file in the LMS platform.

Revision History

Date	*Author*	*Description*
10/3/22	Group	Original draft
10/4/22	Group	Added requirements
10/17/22	Oliver	Original P2 outline created, enrich the review report and refine the requirements
10/17/22	Hannah, Aidan, Zhijie	Added requirements, use cases and diagrams to the document
10/18/22	Group	Finished the P2 document

(Detailed revision information is shown in the review report.)