Software Requirements Specification

for

Exam Generator Application

Version 1.1 approved

Prepared by Bryan Smith

Joe LaCava

Scott Arnette

Derek Ouzia

University of Virginia's College at Wise

Department of Mathematics and Computer Science

2/10/15

Table of Contents

Table of Contents ii
Revision History iii
1. Introduction 1
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 1
1.5 References 1
2. Overall Description 1
2.1 Product Perspective 1
2.2 Product Functions 2
2.3 User Classes and Characteristics 2
2.4 Operating Environment 2
2.5 Design and Implementation Constraints 3
2.6 User Documentation 3
2.7 Assumptions and Dependencies 3
3. External Interface Requirements 3
3.1 User Interfaces 3
3.2 Hardware Interfaces 3
3.3 Software Interfaces 3
3.4 Communications Interfaces 3
4. Functional Requirements 3
4.1 The application shall read two JSON input files. 3
4.2 The application shall produce one output document for each interaction 4
5. Other Nonfunctional Requirements 4
5.1 Performance Requirements 4
5.2 Safety Requirements 4
5.3 Security Requirements 4
5.4 Software Quality Attributes 5
6. Other Requirements 5
Appendix A: Glossary 5
- **SRS –** Software Requirements Specification. This documents that specifies the requirements for a project. 5
- **EGA –** Exam Generation Application. The project that this SRS is for. 5
- **CLI –** Command Line Interface. The user will issue commands using the keyboard to interact with the application. 5
- **JSON –** JavaScript Object Notation. A format that allows representation of objects in a text format. 5

Appendix B: Analysis Models 5
See figure 5-A in the Software Design Document 5
Appendix C: To Be Determined List 5
None 5

Revision History

Name	Date	Reason For Changes	Version
Bryan Smith	2/10/15	Initial creation.	1.0
Scott Arnette	2/11/15	Additions, 1.2, 1.3	1.01
Bryan Smith	2/12/15	Complete the left over sections (Section 2.3, 2.5, 3.1 and the Appendices)	1.02
Bryan Smith	3/16/15	Explained more what the criteria is for the input file for 4.1.2. Numbered the non-functional requirements for section 5. Changed requirements relating to GUI, as there will no longer be a GUI for the application.	1.0.3
Bryan Smith	5/7/15	4.1.3 change. User will now specify input data locations.	1.1

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification is to define the requirements for the Exam Generation Application in its entirety.

1.2 Document Conventions

This document does not feature any fonts or highlighting that will signify any special circumstances. All high level requirements will be represented by additional detailed requirements, with priorities implemented when all known requirements have been listed and this document established as a baseline.

1.3 Intended Audience and Reading Suggestions

This document is intended to be read by all stakeholders of this project, including developers, testers, and users. The remainder of this document provides information about the system being developed, such as its description, intended functionality, operational environment, and any interfaces it may use. Finally, this document will detail the requirements this system shall adhere to, such as functionality or any nonfunctional requirements that the system shall follow.

1.4 Product Scope

The Exam Generator Application will provide professors the ability to generate exams from a specified dataset of questions and answers. These questions will have the ability to be different variations such as multiple choice, true or false, matching, and short answer. The professor can specify the types of questions wanted and the categories needed. The resulting exam will be generated with questions grouped by type.

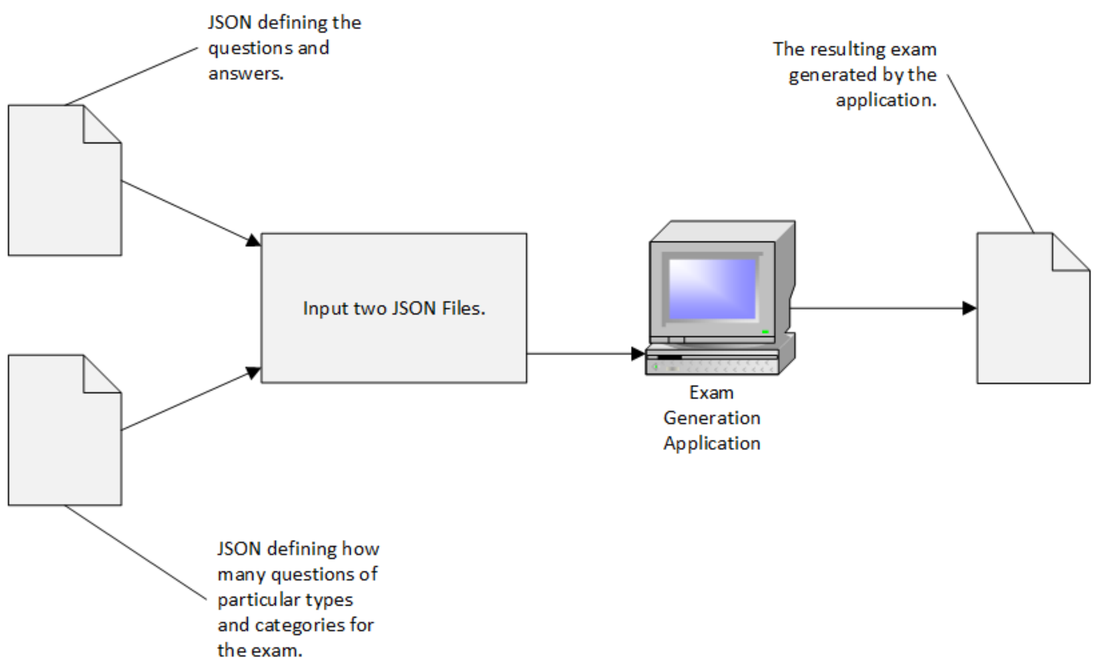
1.5 References

1.5.1 JSON Parser Library (org.json) http://www.json.org/java/index.html

2. Overall Description

2.1 Product Perspective

The Exam Generator Application project is a new, self-contained project. Below is a diagram showing the overview model of the system.


JSON defining the questions and answers.
Input two JSON Files.
Exam Generation Application
The resulting exam generated by the application.
JSON defining how many questions of particular types and categories for the exam.

2.2 Product Functions

Application that allows a professor to create an exam from a set of questions in JSON format.

- There will be two separate input JSON files:
    - One for the set of questions.
    - One to guide the application as to how many questions of particular types and categories should be selected for the resulting output file.
- Each question is an element in the JSON file.
- Each question has the following attributes:
    - Type = Short Answer/ Matching / True-False / Multiple Choice
    - Question
    - Answer
    - Category
- The resulting output file should be a plain text file that lists the questions grouped together by type.

2.3 User Classes and Characteristics

Due to the simplistic nature of the Exam Generator Application, the user classes will now be distinct enough to have individual user classes. The users will all be performing the same actions and there will not be any security levels. This simply leaves the only user class as “User”.

2.4 Operating Environment

The Exam Generation Application shall run on typical desktop PCs and Laptops that are running Microsoft Windows or a distribution of Linux, which has Java installed.

2.5 Design and Implementation Constraints

Design and Implementation of the Exam Generation Application will only be limited by the use of third party libraries or code, with the exclusion of a parser for the JSON data.

2.6 User Documentation

Due to the simplistic nature, no User Documentation will be produced outside of in-application guidance.

2.7 Assumptions and Dependencies

- JSON Parser Library org.json (See Section 3.3)

3. External Interface Requirements

3.1 User Interfaces

Exam Generation Application will use a command line interface (CLI). The CLI will provide the user with commands appropriately allowing them to read in input files and generate an output exam file. All custom criteria for the exam output will be specified by the user in the second input file. (See 4.1.2)

3.2 Hardware Interfaces

The Exam Generator Application will run on a device, which shall have a keyboard and mouse (or equivalent) for input.

3.3 Software Interfaces

- The application will run using Java and the Java Virtual Machine.
- The application will use native operating system functions for reading and writing files.
- The application will use the org.json library for JSON parsing. (See reference 1.5.1)

3.4 Communications Interfaces

The Exam Generator Application is only ran locally and files are only read and saved locally. No external or network connections are made.

4. Functional Requirements

4.1 The application shall read two JSON input files.

4.1.1 The first input file shall contain the set of questions.

4.1.1.1 Questions shall include Type, Question, Answer, and Category.

4.1.1.2 Each Question in the output file shall be one of the following types: Short Answer / Matching / True-False / Multiple Choice.

4.1.2 The second input shall contain the criteria to be used when making selections.

4.1.2.1 Criteria for selection will be the number of questions to use.

4.1.2.2 Criteria for selection will be the type of questions to use. (See 4.1.1.2)

4.1.2.3 Criteria for selection will be the category of questions to use.

4.1.3 The input JSON files location shall be specified by the user.

4.1.4 The two JSON input files shall be valid JSON files.

4.2 The application shall produce one output document for each interaction

4.2.1 The output file shall contain the selected questions grouped by type.

4.2.2 The application shall request the user to provide a name for the output document

4.2.3 The application shall request the user to provide a save location for the output document.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 The application shall load in less than 1 second.

5.1.2 The application shall generate an exam output no slower than 1.5 sec after generation started.

5.2 Safety Requirements

5.2.1 The application will never be allowed to read, write, or delete any files other than those that are created by the application or files that the user specifies for input.

5.2.2 The application, regardless if user specified, should never interact with files belonging to the operating system.

5.3 Security Requirements

5.3.1 The application may not read any files other than JSON files supplied by the user from the same folder as the application.

5.3.2 The application may not write any files other than the output file specified by the user.

5.3.3 The application may not overwrite any files that were not created by the application.

5.3.4 The application will never be allowed to delete files not created by the application itself.

5.4 Software Quality Attributes

5.4.1 The application will be portable by being self-contained in an .exe or .jar file.

5.4.2 The application will be portable by saving any settings in an .ini file in the local directory of the executable.

5.4.3 The application shall notify the user in case the input files are empty.

5.4.4 The application shall notify the user in case the content of the input files cannot be read.

5.4.5 The application shall notify the user in case the content of the output file cannot be written.

5.4.6 The application will inform the user if the JSON files are invalid.

6. Other Requirements

- The application shall be developed in the Java Programming Language.

Appendix A: Glossary

- **SRS –** Software Requirements Specification. This documents that specifies the requirements for a project.
- **EGA –** Exam Generation Application. The project that this SRS is for.
- **CLI –** Command Line Interface. The user will issue commands using the keyboard to interact with the application.
- **JSON –** JavaScript Object Notation. A format that allows representation of objects in a text format.

Appendix B: Analysis Models

See figure 5-A in the Software Design Document

Appendix C: To Be Determined List

None