Software Requirements Specification

for

Exam Generator Application

Version 1.0 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 ii

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 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

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. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

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

## 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.

## 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.

## 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.

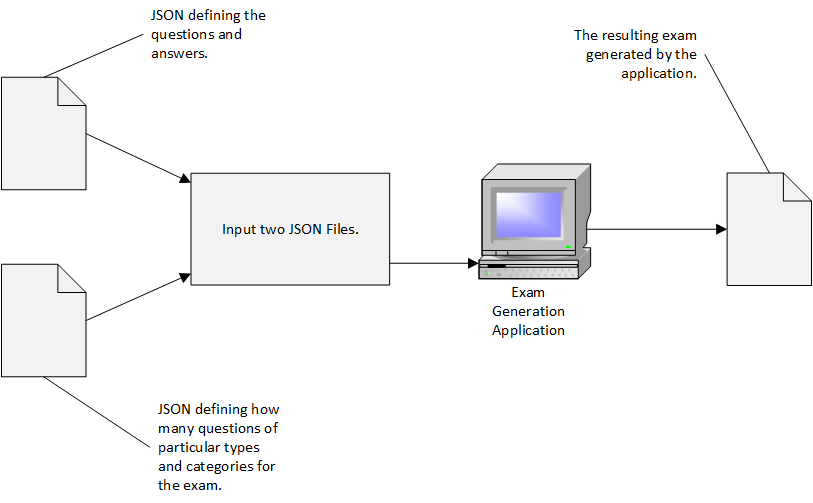
## References

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

# Overall Description

## Product Perspective

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



## Product Functions

Application which 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.

## User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

## 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.

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

## User Documentation

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

## Assumptions and Dependencies

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

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

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

## 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 )

## 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.

# Functional Requirements

## The application shall read two JSON input files.

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

#### Questions shall include: Type, Question, Answer, and Category.

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

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

### The input JSON files shall be located in the application folder.

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

## The application shall produce one output document for each interaction

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

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

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

# Other Nonfunctional Requirements

## Performance Requirements

* The application shall load in less than 1 second.
* The application shall generate an exam output no slower than 1.5 sec after generation started.

## Safety Requirements

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

## Security Requirements

* The application may not read any files other than JSON files supplied by the user from the same folder as the application.
* The application may not write any files other than the output file specified by the user.
* The application may not overwrite any files that were not created by the application.
* The application shall never be allowed to delete files not created by the application itself.

## Software Quality Attributes

* The application will be portable by being self-contained in an .exe or .jar file.
* The application will be portable by saving any settings in an .ini file in the local directory of the executable.
* The application shall notify the user in case the input files are empty.
* The application shall notify the user in case the content of the input files cannot be read.
* The application shall notify the user in case the content of the output file cannot be written.
* The application will inform the user if the JSON files are invalid.

# Other Requirements

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

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>