Software Design Document

for

Exam Generator Application

Version 1.04 approved

Prepared by Bryan Smith

Joe LaCava

Scott Arnette

Derek Ouzia

University of Virginia's College at Wise

Department of Mathematics and Computer Science

3/19/15

Table of Contents

Table of Contents ..... **ii**
Revision History ..... **ii**
1. Introduction **1**
1.1 Purpose 1
1.2 Product Scope 1
1.3 Overview 1
1.4 References 1
2. System Overview **1**
3. System Architecture **1**
3.1 Architectural Design 1
3.2 Decomposition Description 2
3.3 Design Rationale 3
4. Data Design **4**
4.1 Data Description 4
4.2 Data Dictionary 4
5. Component Design **5**
6. Human Interface Design **6**
6.1 Overview of User Interface 6
6.2 Screen Images 6
6.3 Screen Objects and Actions 6
7. Requirements Matrix **6**
Appendix A: Glossary **7**
- **SRS –** Software Requirements Specification. This documents that specifies the requirements for a project. **7**
- **EGA –** Exam Generation Application. The project that this SRS is for. **7**
- **CLI –** Command Line Interface. The user will issue commands using the keyboard to interact with the application. **7**
- **JSON –** JavaScript Object Notation. A format that allows representation of objects in a text format. **7**

Appendix B: To Be Determined List **7**

Revision History

Name	Date	Reason For Changes	Version
Scott Arnette	3/19/15	Initial creation.	1.0
Bryan Smith	3/19/15	Minor fixes, add diagrams, and finish remaining sections. Mostly 3, 4 5, and 6.	1.01
Derek Ouzia	3/19/15	Additions to 3.3	1.02
Bryan Smith	4/28/15	Update Class Diagram with new JSON Parsing changes	1.03
Bryan Smith	5/1/15	Update Class Diagram with Command Line changes	1.04

1. Introduction

1.1 Purpose

The purpose of this Software Design Document is to define the architecture and system design of the Exam Generation Application in its entirety, showing and describing the subcomponents accurately.

1.2 Product Scope

The Exam Generation Application is a program written in Java to be used by professors. The goals of the Exam Generation Application are to provide a simple generator for creating tests based on a bank of various questions (matching, short answer, true/false, multiple choice).

1.3 Overview

This document includes various UML diagrams to both define and describe how the Exam Generator Application shall function and how the components interconnect.

1.4 References

No references used.

2. System Overview

The Exam Generation Application is a new project allows the ability to generate an exam from two separate input JSON files, one featuring the sets of questions and the other guiding how many questions of each type should be selected. Questions will be one of four types; short answer, matching, true/false, or multiple choice.

3. System Architecture

3.1 Architectural Design

The Exam Generation Application shall contain 3 packages. A default package containing the required (by the Java programming language) main class. The second package is the CommandLine package, which will contain classes relating to user input. Lastly, the third package is the JsonParsing package which shall handle the parsing and construction of the data read in by the application. The main class will contain an instance of a single CommandLine package class. The main class will use the CommandLine package classes to take input from the user and to load and save files. The JsonParsing package classes will be ran from the CommandLine classes and will be used to parse and generate data read in from the JSON files.

3.2 Decomposition Description

Figure 3-A below is a use case diagram, and Figure 3-B is a sequence diagram for the Exam Generator Application.

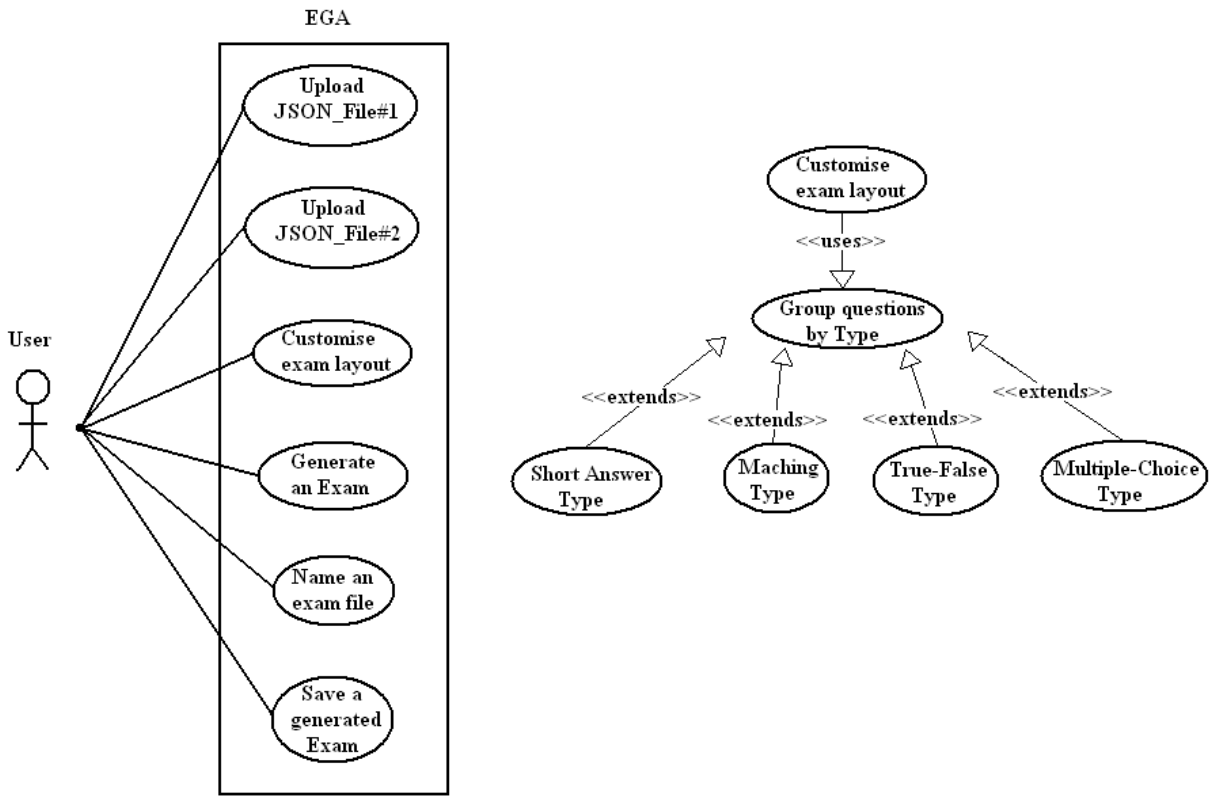

Figure 3-A

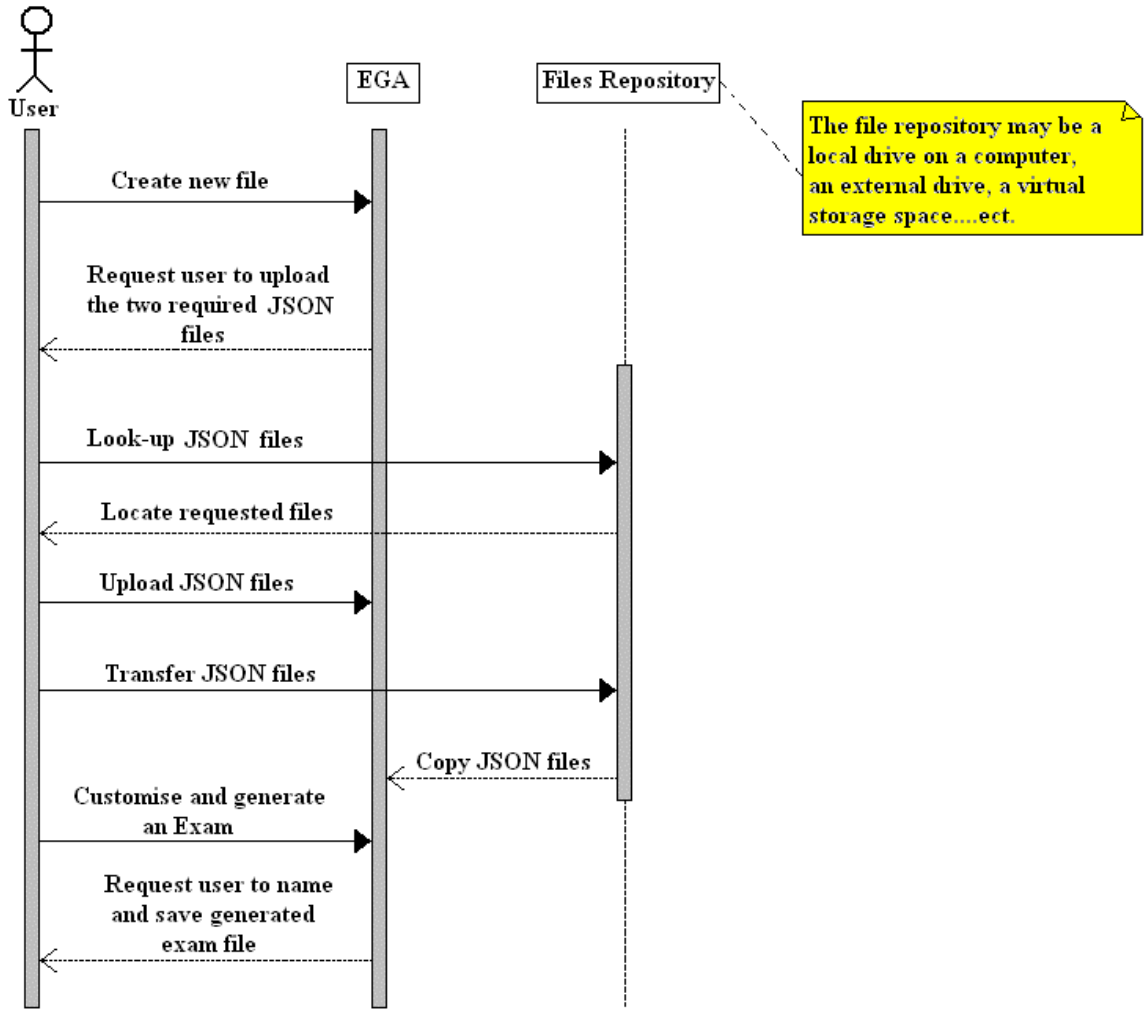

Figure 3-B

3.3 Design Rationale

The object oriented design approach was the architecture of choice for implementing the Exam Generator Application because it was very suitable for modelling the system components and their interactions. The OO design architecture simplifies any complexities with the application through decomposition and functional representation and provides modifiability, maintainability, portability, and testability. The design keeps the parsing and file reading separated as well as both packages from the main class. Overall, the design is not complicated and the application performs a straightforward task.

4. Data Design

4.1 Data Description

Data is read in via JSON files by the application and by using the third party library org.json, the JSON key-value pairs are loaded into the appropriate classes as seen in Section 5. The data will be validated as a valid JSON format using functions provided by the library, then if valid, the data will be loaded into appropriate instances of the Questions and Answers classes. Once this data is stored, the questions and answers based on the criteria specified by the user in the second JSON file will be written to a specified output text file.

4.2 Data Dictionary

Below is a data flow diagram for the Exam Generator Application (Figure 4-A):

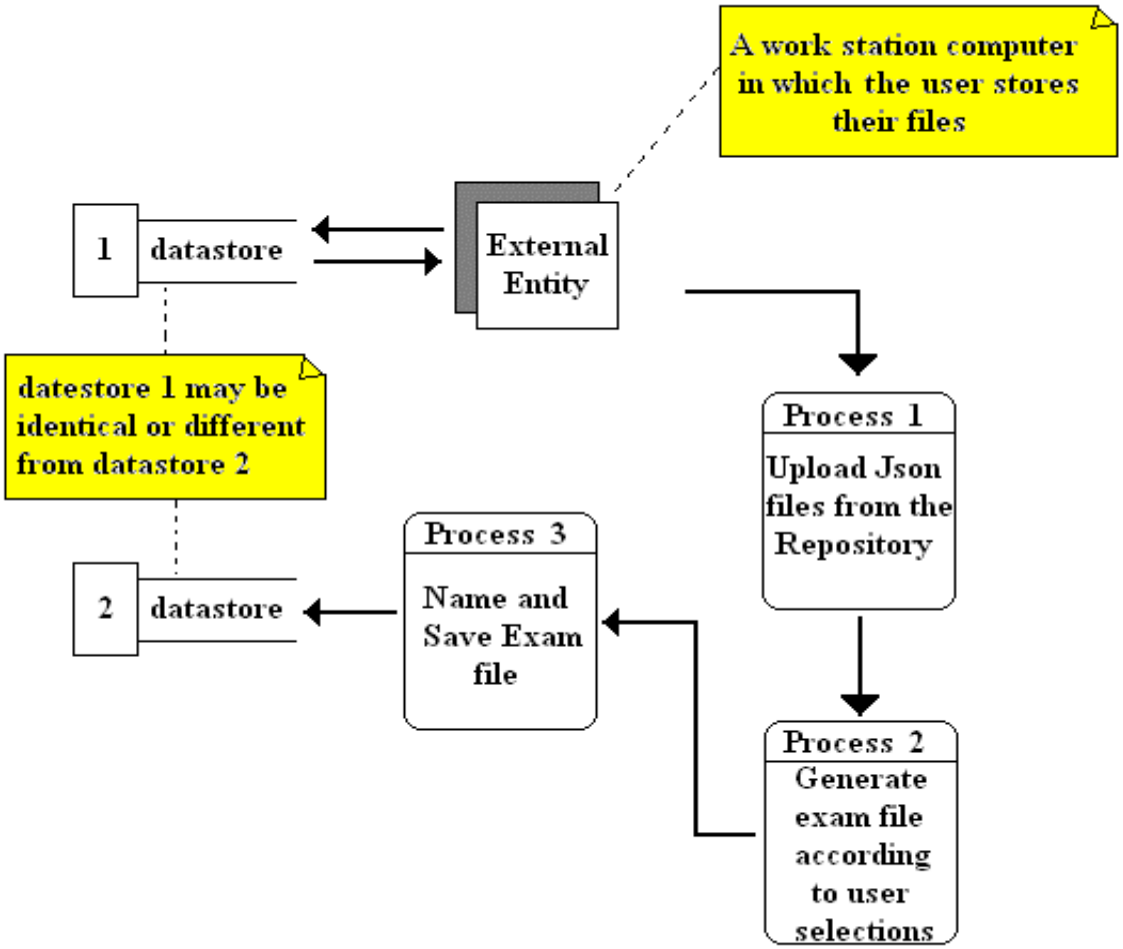

Figure 4-A

5. Component Design

Below is a class diagram for the Exam Generator Application (Figure 5-A):

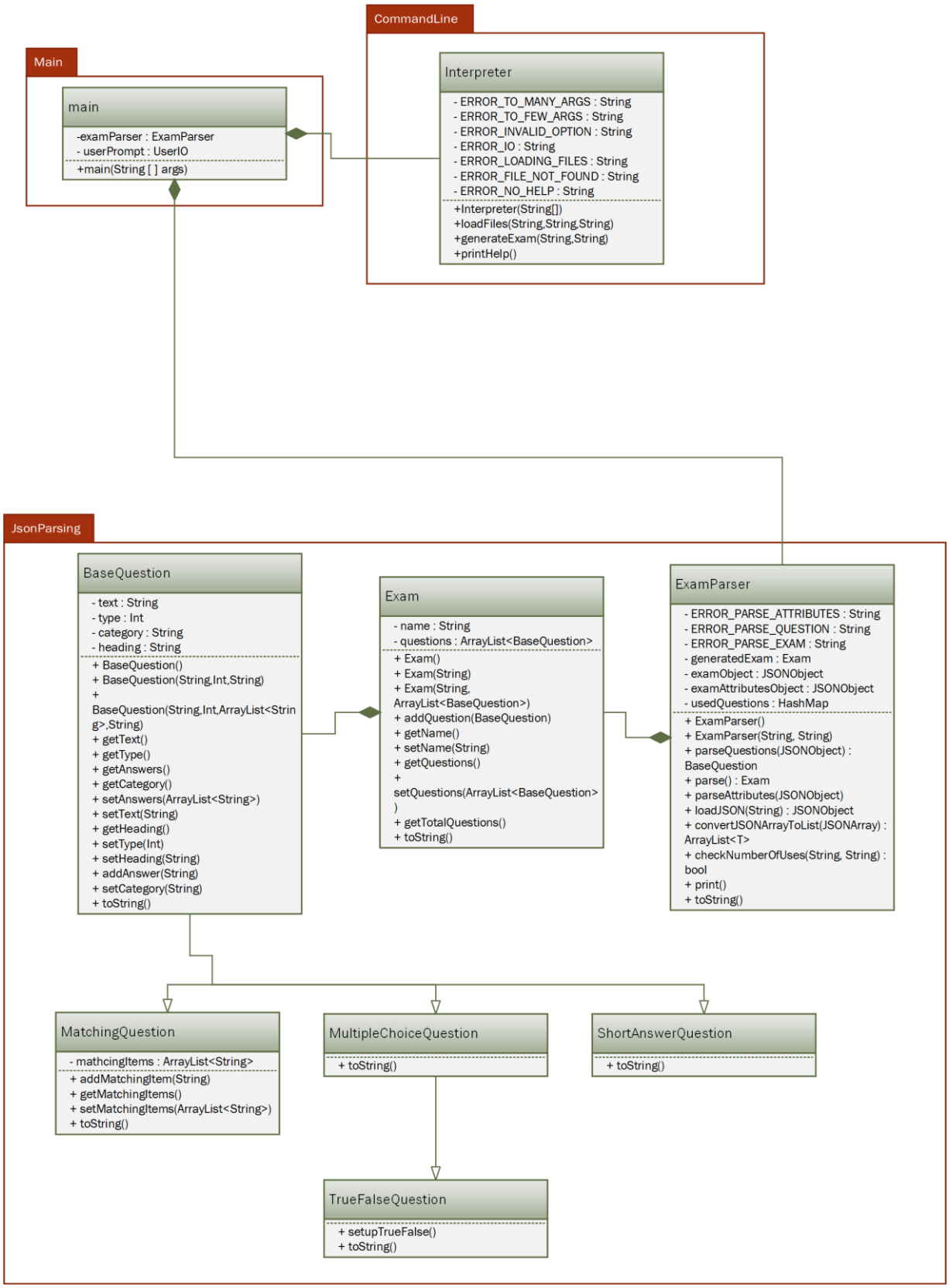

Figure 5-A

6. Human Interface Design

6.1 Overview of User Interface

The Exam Generator Application will not feature a graphical user interface (GUI), but will instead expect the files to be located within a predetermined directory. If the files needed are not present, the user will be displayed an error message. If the files required are present and formatted correctly, the application will continue operation.

6.2 Screen Images

The Exam Generator Application shall only use a CLI interface. Any GUI parts are strictly related to the host Operating System and not required for the application to function properly. Below (Figure 6-A) is a representation of how the CLI may look on modern GUI based operating systems.

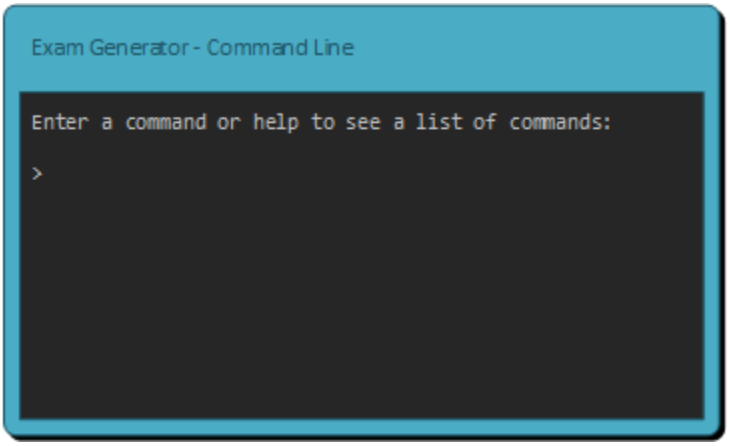

Figure 6-A

6.3 Screen Objects and Actions

Not applicable.

7. Requirements Matrix

	4.1	***4.1.1***	***4.1.2***	***4.1.3***	***4.1.4***	***4.2***	***4.2.1***	***4.2.2***	***4.2.3***
Main Package									
CommandLine Package	*X*	*X*	*X*	*X*		*X*		*X*	*X*
JsonParsing Package	*X*	*X*	*X*		*X*		*X*		

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: To Be Determined List