Software Design Document

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

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 2

4. Data Design 2

4.1 Data Description 2

4.2 Data Dictionary 2

5. Component Design 2

6. Human Interface Design 2

6.1 Overview of User Interface 2

6.2 Screen Images 3

6.3 Screen Objects and Actions 3

7. Requirements Matrix 3

Appendix A: Glossary 3

Appendix B: To Be Determined List 3

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Scott Arnette | 3/19/15 | Initial creation. | 1.0 |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

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

## Product Scope

<Provide a description and scope of the software and explain the goals, objectives and benefits of your project. This will provide the basis for the brief description of your product.>

The Exam Generation Application is a program written in Java to be used by professors. It provides the ability. 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).

## Overview

<Provide an overview of this document and its organization.>

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

## References

<List any other documents or Web addresses to which this SDD refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

# System Overview

<Give a general description of the functionality, context and design of your project. Provide any background information if necessary.>

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.

# System Architecture

## Architectural Design

<Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high level overview of how responsibilities of the system were partitioned and then assigned to subsystems. Identify each high level subsystem and the roles or responsibilities assigned to it. Describe how these subsystems collaborate with each other in order to achieve the desired functionality. Don’t go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together. Provide a diagram showing the major subsystems and data repositories and their interconnections. Describe the diagram if required.>

## Decomposition Description

<Provide a decomposition of the subsystems in the architectural design. Supplement with text as needed. You may choose to give a functional description or an object-oriented description. For a functional description, put top-level data flow diagram (DFD) and structural decomposition diagrams. For an OO description, put subsystem model, object diagrams, generalization hierarchy diagram(s) (if any), aggregation hierarchy diagram(s) (if any), aggregation hierarchy diagram(s) (if any), interface specifications, and sequence diagrams here.>

## Design Rationale

<Discuss the rationales for selecting the architecture described in 3.1 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn’t choose them.>

# Data Design

## Data Description

<Explain how the information domain of your system is transformed into data structures.

Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.>

## Data Dictionary

<Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 3.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.>

# Component Design

<In this section, we take a closer look at what each component does in a more systematic way. If you gave a functional description in section 3.2, provide a summary of your algorithm for each function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudocode. Describe any local data when necessary.>

# Human Interface Design

## Overview of User Interface

<Describe the functionality of the system from the user’s perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.>

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.

## Screen Images

*<Display screenshots showing the interface from the user’s perspective. These can be hand drawn or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)>*

TO DO

## Screen Objects and Actions

<A discussion of screen objects and actions associated with those objects.>

TO DO

# Requirements Matrix

*<Provide a cross reference that traces components and data structures to the requirements in your SRS document. Use a tabular format to show which system components satisfy each of the functional requirements from the SRS. Refer to the functional requirements by the numbers/codes that you gave them in the SRS.>*

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SDD, 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 SDD.>

Appendix B: To Be Determined List

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