Software Test Plan

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

5/1/15

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Scott Arnette	5/1/15	Initial Creation	1.0
Joe LaCava	5/7/15	Added section 6	1.01
Bryan Smith	5/7/15	Added section 8, 9, 10 and 11	1.02
Scott Arnette	5/7/15	Final compilation and finished remaining sections.	1.1

1. Introduction

The project that this test plan exists for is the Exam Generation Application, a Java based program to be used by professors or other instructing staff for the generation of tests based on a bank of various questions. Testing of the Exam Generation Application will consist of unit tests and system tests to ensure the full-intended functionality of the application is present and no unintended or malicious defects are present.

1.1 Objectives

Activities that will occur for the testing of the Exam Generation Application will be unit tests to ensure each module of the application is working as expected. As the units pass testing, they will be integrated and system tests will be performed to ensure full functionality is present. This test plan will deliver summaries of all tests performed, sample data that will be used during testing, and the procedures used during testing. Major milestones include the success of all unit tests and the success of final system tests.

1.2 Testing Strategy

For the Exam Generation Application, two types of testing will be used. Unit testing will be performed on each component as it is completed. System testing will be performed to ensure the system's full functionality is present and works as intended. The features that will be tested for the Exam Generation Application will be its ability to generate exams based on various inputs. Additionally, error handling will be tested to ensure that the application can handle input data that is not expected. Each member of the team will perform some unit tests on components that he was not responsible for creating.

1.3 Scope

Any changes made to the Software Test Plan will be viewable through GitHub (the repository being utilized). Additional scheduled updates are not expected due to the scope of this project; unscheduled updates will be viewable as a change to this document in the repository.

1.4 Reference Material

This Software Test Plan may reference the Software Design Document and the Software Requirements Sheet of the Exam Generation Application.

1.5 Definitions and Acronyms

STP: Software Test Plan
SCM: Software Configuration Management

2. Test Items

Here, the items that will be tested shall be listed.

2.1 Program Modules

The testing activity will include the Exam Generation Application on Windows based platforms, to ensure all functionality and requirements are met. What is not to be tested includes the Exam Generation Application on other desktop operating systems (Mac OS, Linux, etc.).

2.2 User Procedures

Included in the Exam Generation Application is a document that explains commands to be used. All commands will be tested in the test plan.

2.3 Operator Procedures

No specific testing procedures will be performed to ensure the applications use in a production environment. The application is intended for personal use only.

3. Features

3.1 Features to be Tested

Features of the EGA that will be tested include its ability to generate various exams based on different inputs (different inputs will result in different outputted exams). Additionally, the application's ability to handle incorrect inputs will be tested.

3.2 Features Not to be Tested

As of creation, all intended functionality of the Exam Generation Application will be tested through unit and system testing.

4. Approach

Listed here is how testing will be performed by the team. Tests will include component testing, integration testing, and acceptance testing.

4.1 Component Testing

Component testing will ensure that all modules accept inputs correctly, perform their intended function and create appropriate outputs. Component testing will be performed to ensure all units are acceptable and complete before integration is performed.

4.2 Integration Testing

Integration testing will be performed once all modules are created and component tests completed. The objective during integration testing is to ensure that all functionality is working as intended. Testing will include using appropriate inputs to ensure correct outputs are generated, using incorrect inputs to ensure errors are thrown and the end-user notified, and ensuring the performance of the entire system is within satisfactory boundaries.

4.3 Interface Testing

As the Exam Generation Application is a stand-alone system with no external interfaces, no interface testing will be performed.

4.4 Regression Testing

As only one integration occurs, no in depth regression testing will occur, as all functionality will be maintained in the system test.

4.5 Acceptance Testing

The performance of acceptance testing will be to ensure that the correct output files are being generated from the Exam Generation Application. This will be similar to system testing being performed.

4.6 Beta Testing

For the Exam Generation Application, no outside beta testing will be performed.

5. Pass/Fail Criteria

Here, the criteria for the status of test items will be detailed.

5.1 Suspension Criteria

If a test item fails to complete its specified test for any reason, it will be suspended for correction. A bug report will be created and a developer assigned to investigate the issue.

5.2 Resumption Criteria

Once a developer has marked a bug as in progress and then has “fixed” the bug, the test item will be moved to resumption status.

5.3 Approval Criteria

Only if all unit and system tests have been passed will an item be approved and has successfully passed testing.

6. Testing Process

Since the scope of this project was relatively simple and the size of the project was relatively small, we elected to only perform the testing methods as follows: dynamic testing, and mostly white-box testing. The process employed was primarily done after each module was completed, in which, a test sample of data was autonomously generated and use to the test that current module. Once all the pieces were connected, we generated another set of data autonomously to test the system as a whole. The only acceptance criteria required, was the ability for the program to operate in manner expected.

6.1 Test Deliverables

The only deliverables that were requested for this project include this testing document and the corresponding test cases. Test Input and Output data will be provided along with the Test cases themselves. No further testing and corresponding testing documentation were requested by the client.

6.2 Testing Tasks

The only set of tasks and skills required to test the system fall to an individual's proficiency with the Eclipse IDE. Since the software would be written and subsequently tested within the same IDE it would go to reason that the individual testing the system would have a working knowledge of both Java, the language the software system is encoded, and Eclipse IDE, the software in which the system is executed. There were no testing report systems that were used in this testing process.

6.3 Responsibilities

Since we are a development group consisting of four individuals the tasking was shared mutually, four way split. The roles of each individual were not clearly defined and subsequently each member has fulfilled the requirements of each role as some point during development. The project scope is too small, and the development is too small for a clearly defined partition.

6.4 Resources

No resources needed to be allocated as the systems used for the testing and the man hours associated were already provided and were not included in the project budget. In fact, there was no project budget. There were no automated tools used in the testing process.

6.5 Schedule

The schedule used was based upon module delivery deadlines. Testing was done on a unit scale focusing on the completed modules since most modules consisted of 2-3 functions. The module testing was done prior to reporting task completion and since the development team was the testing team, any errors found were corrected at the time of testing.

7. Environmental Requirements

7.1 Hardware

No specific hardware is required for the performance of testing. No specific network connection is required to complete testing activities.

7.2 Software

Software used to complete testing activities included a Windows based operating system with Java 1.7 or higher installed. Unit tests can be performed in the Eclipse IDE for Windows.

7.3 Security

No security requirements are needed for the testing environment in which the Exam Generation Application is tested.

7.4 Tools

The only software tool that is needed for testing purposes is the Eclipse Java IDE. This tool allows for unit testing on demand.

7.5 Publications

The readme document included with the application can be used to test all commands in the application.

7.6 Risks and Assumptions

Risks of testing include time constraints, depending on the completion of the Exam Generation Application, creating scheduling strains on the completion of the project.

8. Change Management Procedures

For Software Configuration Management (and subsequently Change Management) we took advantage of our source code repository hoster's features. GitHub overlays an array of features on top of repositories and the version control system Git. These features, such as pull requests, allow changes to be sent to the repository (even by developers not on the team, to facilitate open source projects), reviewed by the team or a designated reviewer, and then merged into the repository. This same mechanism can be used by team members as well, and in some cases has to be used in order to submit changes if a team member does not have repository write access (which is often the case, only admins would).

Users and developers can 'clone' the repository, make their changes to the codebase or documentation, and then get ready to initiate a pull request (called a pull request, because it 'pulls' their changes into the main repository). Changes are initiated by users or developers by submitting their changes (actual changes to the code, not just a theory or proposal) to the repository in a pull request, a team member with permissions can approve or deny the change or comment on it with any pertinent requests or information. If approved, the changes are merged into the specified repository branch.

9. Plan Approvals

X

Bryan Smith
Test Plan Approval

X

Scott Arnette
Testing Plan Approval

10. Test Procedures

Test Procedure Number:	1		
Date Tested:	4/30/15		
Test Performed By:	Joseph LaCava		
Project Name:	Exam Generator (Command Interpreter)		
Software Version:	1.0		
Related Requirements:	4.1.3, 4.2.2, 4.2.3, 5.2.1-5.3.4		
#	Test Step Description	Expected Result	Passed
1	Test using no arguments	Display help file	Yes
2	Test loading with no file paths specified	Display “You seem to have too few arguments.”	Yes

Test Procedure Number:	2		
Date Tested:	4/30/15		
Test Performed By:	Bryan Smith		
Project Name:	Exam Generator (Exam Parser)		
Software Version:	1.0		
Related Requirements:	4.1.1, 4.1.2, 4.1.4, 4.2.1, 5.4.3-5.4.6		
#	Test Step Description	Expected Result	Passed
1	Test with valid input	Display generated exam	Yes
2	Test loading with blank JSON	Display “Error loading JSON data…”	Yes
3	Test loading with malformed JSON	Display “Error loading question JSON data…”	Yes

Test Procedure Number:	3
Date Tested:	4/30/15
Test Performed By:	Derek Ouzia
Project Name:	Exam Generator (Questions Data Structures)
Software Version:	1.0
Related Requirements:	4.1.1.1, 4.1.1.2

#	Test Step Description	Expected Result	Passed
1	Test MultipleChoiceQuestion creation	Display a valid Multiple Choice Question	Yes
2	Test TrueFlaseQuestion creation	Display a valid True False Question	Yes
3	Test ShortAnswerQuestion creation	Display a valid Short Answer Question	Yes
4	Test MatchingQuestion creation	Display a valid Matching Question	Yes

Test Procedure Number:	4
Date Tested:	5/1/15
Test Performed By:	Scott Arnette
Project Name:	Exam Generator (System Test)
Software Version:	1.0
Related Requirements:	4.1.0-6.0.0

#	Test Step Description	Expected Result	Passed
1	Running without parameters	Display a help file	Yes
2	Use help command	Display a help file	Yes
3	Use exit command	Exit the application	Yes
4	Use load command with valid file paths to test input data and a valid save path	Generate expected exam and save to specified location in a text file.	Yes
5	Use load command with valid file paths to test input data and no save path	Generate expected exam and save to current working directory in a text file.	Yes
6	Use load command with valid file paths to test input data and a save path with invalid characters	Generate expected exam and save to current working directory or specified location in a text file. Replace invalid characters with an underscore.	Yes
7	Use load command with invalid file paths to test input data and a valid save path	Display "Error loading JSON data…"	Yes
8	Use load command with invalid file paths to test input data and no save path	Display "Error loading JSON data…"	Yes
9	Use load command with invalid file paths to test input data and a save path with invalid characters	Display "Error loading JSON data…"	Yes

11. Requirements Matrix

	Functional Requirements									
Test Procedure	4.1.1.1	4.1.1.2	4.1.2.1	4.1.2.2	4.1.2.3	4.1.3	4.1.4	4.2.1	4.2.2	4.2.3
1						X			X	X
2	X	X	X	X	X		X	X		
3	X	X								
4	X	X	X	X	X	X	X	X	X	X