SRS DOCUMENT

Team Actimel
Go team Actimel.

Department of Engineering
School of Informatics & Engineering
Institute of Technology, Blanchardstown
Dublin 15.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Document overview

The project is to produce a product using TinyXML, the product is a program that will generate a multiple choice quiz to the user, who will have only ten attempts. This product is aimed at students wishing to use the program as a learning or study aid.

There will be two types of user, student and administrator. Administrator will have access to the quiz bank in order to edit the multiple choice questions list, can take the quiz for testing purposes and also access the class reports.

The student can simply access their personal reports and start or restart the quiz.

The requirements of the client is to receive a product that does all of the above with ease of editing for administration.

Verification is done through series of testing methods such as the following White Box and Black Box testing:

In relation to testing, for White Box testing the Software is tested using 'Acceptance Testing', this is basically running the code and seeing if it preforms the way it should, for example:
Can the users log into their profiles?
Can the quiz be started?
Does 10 question appear on the screen?
Can the questions be completed?
Are results returned?
Can the profile be viewed?
Can admin access the quiz bank and edit the questions?
Can the quiz be retaken?
Does it generate ten attempts?
Etc...

For Black Box testing, The Software is tested using Google test harness (G Test).
Google Test is a unit testing library for the C++ programming language, based on the xUnit architecture. The library is released under the BSD 3-clause license. It can be compiled for a variety of POSIX and Windows platforms, allowing unit-testing of C sources as well as C++ with minimal source modification.

The product covers all aspects of the client's needs, a full range of detailed analysis and design documentation was produced to ensure full compliance of the client's wishes.

The team involved in this project fully understood the importance of the client's needs and requirements in relation to producing a reliable and easy to use product. Taking into consideration the open-closed Principle which is basically to write the modules so that they can be extended, without requiring them to be modified, in other words, we want to be able to change what the modules do, without changing the source code of the modules.

1.2 Abbreviations and Glossary

1.2.1 Abbreviations

SRS – Software requirements Specifications
G Test – Google Test Harness
I – Inspection
A – Analysis
D – Demonstration
T – Test

1.2.2 Glossary

TinyXML: Light-weight xml parser used for serialization.
White Box: Initial testing method used to see if software works, does what it's meant to do.
Black Box: Final testing method used to test functions are working correctly.
Inspection (I): control or visual verification
Analysis (A): verification based upon analytical evidences
Demonstration (D): verification of operational characteristics, without quantitative measurement
Test (T): verification of quantitative characteristics with quantitative measurement

1.3 References

1.3.1 Project References

#	Document Identifier	Document Title
[R1]	TinyXML Tutorial	https://shilohjames.wordpress.com/2014/04/27/tinyxml2-tutorial/ Tutorial on how to use TinyXML
[R2]	Git Hub	https://github.com/PaulKi/quiz Project file sharing

2 REQUIREMENTS

2.1 States

There are only two basic states in this program:
Waiting to see if user wants to take the quiz or not.
In a quiz.

2.2 Functionalities and Performance

- Class User
  - Function 1.1 getName: Access function to return the name variable of class user
  - Function 1.2 getID: Access function to return the name variable of the class user
  - Function 1.3 Login: Login
  - Function 1.4 Answer: To send an answer
- Class student public user – inherited User class
  - Function 2.1 viewStudent: To retrieve and display student profile
  - Function 2.2 restartQuiz: To restart the quiz
- Class Admin public user – inherited User class
  - Function 3.1 view: To retrieve and display student profile
  - Function 3.2 remove: To remove a question from question database
  - Function 3.3 add: To add a question from question database
- Class Quiz - public
  - Vector 1.1 qQuestions: A vector used to store 10 questions as text
  - Vector 1.2 qAnswers: A vector to store the 10 correct answers to the questions stored in qQuestions
  - Function 4.1 makesQs: To fetch 10 random questions from database
  - Function 4.2 sendQ: Print question to screen and request answer from database
  - Function 4.3 verifyQs: Check user entered answer versus correct answer
  - Function 4.4 report: generate and print report
  - Function 4.5 save: save quiz result & report

Requirement ID	**SRS-XXX-010**
Title	*Function 1.1 getName*
Description	Access function to return the name variable of class user
Version	V1.0

Requirement ID	**SRS-XXX-020**
Title	*Function 1.2 getID*
Description	To return the name variable of the class user
Version	V1.0

Requirement ID	**SRS-XXX-030**
Title	*Function 1.3 Login*
Description	Login
Version	V1.0

Requirement ID	**SRS-XXX-040**
Title	*Function 1.4 Answer*
Description	To send an answer
Version	V1.0

Requirement ID	**SRS-XXX-050**
Title	*Function 2.1 viewStudent*
Description	To retrieve and display student profile
Version	V1.0

Requirement ID	**SRS-XXX-060**
Title	*Function 2.2 restartQuiz*
Description	To restart the quiz
Version	V1.0

Requirement ID	**SRS-XXX-070**
Title	*Function 3.1 view*
Description	To retrieve and display student profile
Version	V1.0

Requirement ID	**SRS-XXX-080**
Title	*Function 3.2 remove*
Description	To remove a question from question database
Version	V1.0

Requirement ID	**SRS-XXX-090**
Title	*Function 3.3 add*
Description	To add a question from question database
Version	V1.0

Requirement ID	**SRS-XXX-100**
Title	*Vector 1.1 qQuestions*
Description	A vector used to store 10 questions as text
Version	V1.0

Requirement ID	**SRS-XXX-110**
Title	*Vector 1.2 qAnswers*
Description	A vector to store the 10 correct answers to the questions
Version	V1.0

Requirement ID	**SRS-XXX-120**
Title	*Function 4.1 makesQs*
Description	To fetch 10 random questions from database
Version	V1.0

Requirement ID	**SRS-XXX-130**
Title	*Function 4.2 sendQ*
Description	Print question to screen and request answer from database
Version	V1.0

Requirement ID	**SRS-XXX-140**
Title	*Function 4.3 verifyQs*
Description	Check user entered answer versus correct answer
Version	V1.0

Requirement ID	**SRS-XXX-150**
Title	*Function 4.4 report*
Description	Generate and print report
Version	V1.0

Requirement ID	**SRS-XXX-160**
Title	*Function 4.5 save*
Description	Save quiz result & report
Version	V1.0

2.3 Safety, security, and privacy protection

Requirement ID	**SRS-XXX-030**
Title	*Personal profile data*
Description	XXX ensures that the displayed personal profile data is the same as read in the input files. The personal data is: • Name • Student Number • Results • History
Version	V1.0

2.4 User maintenance

Requirement ID	**SRS-XXX-040**
Title	*Application logs*

Description	XXX generates a log file containing:
	• The User logs in
	• User defined as either Student or Admin
	  ○ Admin
	  ▪ Access class profile
	  ▪ Access quiz bank
	  ▪ Edits quiz bank
	  ▪ Takes quiz
	  ▪ Log out
	  ○ Student
	  ▪ Access personal profile
	  ▪ Starts quiz
	  ▪ Access personal results
	  ▪ Retake quiz x ten attempts
	  ▪ Log out
	• The possible error logs, if any

Version	V1.0

2.5 Usability and human-factors engineering

Human interaction with the software is simply by starting the program and interacting with the questions by answering them and waiting for a report to generate.

Requirement ID	**SRS-XXX-050**
Title	*Menu items*
Description	XXX software has the following items: • Text based interface
Version	V1.0

2.6 System environment

The Software is not to be used as part of an external system, it is designed to be a stand-alone program created specifically for a determined closed user group.

2.7 External interfaces

The system features a text based program window for all user input and outputs.

2.7.1 Data exchange

The system takes advantage of an XML parser such as TinyXML2, for file management, user credentials and database management.

2.8 Resources

2.8.1 Hardware resources

Requirement ID	**SRS-XXX-080**
Title	*Hardware configuration*
Description	XXX shall run with any modern computer
Version	V1.0

2.8.2 Software resources

Requirement ID	**SRS-XXX-090**
Title	*Software configuration*
Description	XXX runs in the following software environment: • Windows
Version	V1.0

2.9 Internal data

This program uses XML for internal data.

2.10 Adaptation

No specific requirements necessary to run this program apart from Windows OS.

2.11 Verification

Verification is done through series of testing methods such as the following, White Box and Black Box testing:

For final verification Black Box testing is preformed, The Software is tested using Google test harness (G Test).

Google Test is a unit testing library for the C++ programming language, based on the xUnit architecture. The library is released under the BSD 3-clause license. It can be compiled for a variety of POSIX and Windows platforms, allowing unit-testing of C sources as well as C++ with minimal source modification.

2.12 Personnel and training

The program is designed to be as simple as possible, however the Administrator may require basic training in relation to accessing the quiz bank/database in order to edit the questions.

Requirement ID	**SRS-XXX-USR-010**
Title	*E-learning*
Description	XXX is delivered with e-learning module.
Version	V1.0

2.13 Packaging and installation

Requirement ID	**SRS-XXX-PAK-010**
Title	*Packaging*
Description	Zip file
Version	V1.0

3 VERIFICATION METHODS

Discard this section if you don't want to have verification methods attached to your requirements.

The verification methods of the requirements are defined below:

- Inspection (I): control or visual verification
  - Control of the physical implementation or the installation of a component. The control verifies that the implementation or the installation of a component is compliant with the requirements of diagrams.
  - Control of the documentation describing a component. The control verifies that the documentation is compliant with the requirements.
- Analysis (A): verification based upon analytical evidences
  - Verification of a functionality, performance or technical solution of a component by analyzing the data collected by tests in real conditions, by simulation of real conditions or by a analysis report.
  - Analysis of test data or of design data is used as appropriate to verify requirements.
  - The verification is based upon analytical evidences obtained by calculations, like modeling, simulation and forecasting.
  - Analysis is used when an acceptable level of confidence cannot be established by other methods or if analysis is the most cost-effective solution.
- Demonstration (D): verification of operational characteristics, without quantitative measurement
  - Verifying a requirement by demonstration implies that the required functionality specified by a requirement is complete.
  - Demonstration is used when quantitative measurement is not required for verification of the requirements
  - Demonstration includes the control of the technical solutions specified by the non-functional requirements.
- Test (T): verification of quantitative characteristics with quantitative measurement
  - Verifying a functionality, performance or technical solution of a component by executing testing scenarios in predefined, controlled and traceable testing conditions.
  - Tests require the use of special equipment, instrumentation, simulation techniques, or the application of established principles and procedures,
  - Data produced during tests is used to evaluate quantitative results and compare them with requirements.

For each requirement of the SRS, a verification method is defined. Method is abbreviated I, A, D or T.

Requirement ID	Requirement Title	Method
REQ-001	Whitebox test: The product does what its meant to do The system follows the requirements set out according to the analysis and design documentation?	I
REQ-001	The system follows the requirements set out according to the analysis and design documentation? The system has passed the Acceptance test. (White Box)	A
REQ-001	The system has passed the Acceptance test. Whitebox testing, upon demonstration the system functions correctly and easily	D

	generated as required by the client brief.
	
	• The User can log in
	• User is defined as either Student or Admin
	  o Admin
	  ▪ Access class profile
	  ▪ Access quiz bank
	  ▪ Edits quiz bank
	  ▪ Takes quiz
	  ▪ Log out
	  o Student
	  ▪ Access personal profile
	  ▪ Starts quiz
	  ▪ Access personal results
	  ▪ Retake quiz x ten attempts
	  ▪ Log out
	• The possible error logs, if any	

REQ-001	White Box testing: (Acceptance testing) The system does what it is meant to do. Black Box testing: G test harness test has been designed in order to test out each function of the program, ensuring it is working correctly without error.	T

4 REQUIREMENTS TRACEABILITY

SRS Req.	Req Title	Functional Req.	Req. Title
SRS-REQ-001	Reading ECG values	FUN-REQ-00A	ECG post treatment
SRS-REQ-002	Writing results	FUN-REQ-00A	ECG post treatment

5 CRITICAL REQUIREMENTS

Requirement ID	Requirement Title	Origin
REQ-001	Alarm when value out of range	Risk Analysis
REQ-002	Do not open file if no patient name	Risk Analysis
REQ-003	Display negative values in red color	Human factor engineering