SE 3XA3: Test Plan
Q-aRt Code

Team 05, Q-aRt QTs
Elton Schiott schiotek
Emilio Hajj hajje
Liam Duncan duncanla

November 4, 2016

Contents

List of Tables

List of Figures

No figures at this time.

Table 1: **Revision History**

Date	**Version**	**Notes**
03/11/16	1.0	System and Proof of Concept tests present. Unit testing to be determined after design phase.

1 General Information

1.1 Purpose

This document is an outline of the procedures intended to be used to verify that the software project currently under development, Q-aRt Code satisfies requirements as specified in the SRS document. This plan and the majority of the tests described within have been constructed prior to the completion of the project implementation. As such, the intention of the document is to be used as a reference during the implementation of the software, and during testing.

1.2 Scope

The majority of the project is encoding binary strings from the input data, therefore, the scope of the testing for this software will largely be ensuring correctness of these generated strings, but also includes readability of the final generated QR code, as well as image quality.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
QR Code	Quick Response Code (image-based data encoding)
URL	Uniform Resource Locator (web address)
GUI	Graphical User Interface
PNG	Portable Network Graphic (most ubiquitous digital image format)

Table 3: **Table of Definitions**

Term	**Definition**
No Outstanding Terms	N/A

1.4 Overview of Document

2 Plan

2.1 Software Description

The software under development for this project, Q-aRt Code, is a QR code generator that has the added functionality of creating artistic QR codes. The modular decomposition of the software is as follows:

- Encode binary strings from the input data .
- Generate error correction codewords from the binary strings.
- Structure data and error correction binary strings, including interleaving.
- Create a matrix using the generated data according to QR code standards.
- Combine generated QR Code with input image to create artistic QR code.

2.2 Test Team

Each member of the test team will be responsible for both the creation of tests and their execution. The members of the test team are as follows:

- Elton Schiott
- Emilio Hajj
- Liam Duncan

2.3 Automated Testing Approach

All automated testing will be done on the modular level. Since the system is restricted from GUI through to external scanner, display colour tool or manual interpretation by a person will be used for system testing.

2.4 Testing Tools

Unit testing will be conducted using the Python unittest unit testing framework.

2.5 Testing Schedule

See Gantt Chart at the following url: https://gitlab.cas.mcmaster.ca/schiotek/Q-aRt_Code/blob/master/ProjectSchedule/GanttChartQ.gan

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Functionality and Fidelity of QR Encoder

QR Scanning App

1. Retrieve URL

   Type: Functional, Dynamic.

   Initial State: Output on hand.

   Input: Display/View of QR Code png.

   Output: Originally encoded URL.

   How test will be performed: With a QR Scanning Application.

2. Max. Character URL Retrieval

   Type: Functional, Dynamic.

   Initial State: Output on hand.

   Input: Display/View of QR Code png.

   Output: Originally encoded URL with the maximum number of character.

   How test will be performed: With a QR Scanning Application.

3. Empty String

   Type: Functional, Dynamic.

   Initial State: Output on hand.

   Input: Display/View of QR Code png.

   Output: No Output, displays ”Empty String” message.

   How test will be performed: With a QR Scanning Application.

4. URL with greater than max. characters

   Type: Functional, Dynamic.

   Initial State: Output on hand.

   Input: Display/View of QR Code png, #808080 grey square image.

   Output: No Output, displays error message.

   How test will be performed: With a QR Scanning Application.

3.1.2 Input Handling

1. ASCII not compatible with URL

   Type: Functional, Dynamic.

   Initial State: Start GUI.

   Input: Characters in GUI text box, no image.

   Output: No Output, displays error message.

   How test will be performed: Pressing ”Create” in start GUI.

2. Attempt to paste chars not within the accepted range: in this case Kanji, which the system will not handle

   Type: Functional, Dynamic.

   Initial State: Start GUI.

   Input: Characters in GUI text box, no image.

Output: No Output, displays error message.

How test will be performed: Pressing ”Create” in start GUI.

/endenumerate

3.2 Tests for Nonfunctional Requirements

3.2.1 Visual and Aesthetic Tests

(a) Colour Fidelity

Type: Non-Functional, Dynamic.

Initial State: Start GUI.

Input/Condition: Standard URL, unspecific single colour square image.

Output/Result: QR code with much of its content the same colour as the square image.

How test will be performed: An on-screen hex eyedrop tool will be used to compare the input colour image with elements of the output image.

(b) Visual Text Legibility

Type: Non-Functional, Dynamic, Manual.

Initial State: Output on hand.

Input: Display/View of QR Code png, square image of some text.

Output: QR code with ”artistic” text appearance.

How test will be performed: Member visually inspects output PNG with text chosen by another member, writes down the text they see and compares it to entered text.

3.2.2 Performance

itemReasonable speed

Type: Non-Functional, Dynamic, Manual.

Initial State: Start GUI

Input: Short URL, #808080 grey square image.

Output: QR code image with "artistic" text integrated.

How test will be performed: "create" button pressed, android stopwatch manually started. PNG output, stopwatch manually stopped.

4 Tests for Proof of Concept

4.1 Functional Testing

1. Maximum String Test

   Type: Functional, Dynamic

   Initial State: Python Shell.

   Input: Max char (13 for the Proof of Concept specifically) input string as shell argument.

   Output: QR code PNG image.

   How test will be performed: Max char string will be input on shell prompt, and the output PNG will be tested with an external scanning app to retrieve the original string.

2. Empty String

   Type: Functional, Dynamic

   Initial State: Python Shell.

   Input: Empty string input string as shell argument.

   Output: Error message.

   How test will be performed: Empty string will be input on shell prompt, and no output should be created, with a warning message.

4.2 Non-Functional Testing

The Proof of Concept is purely focused on functional requirement testing.

5 Comparison to Existing Implementation

A comparison of system tests to the existing implementation is immediately availible since we can input identical strings and images, and analyze the output files with the same set of QR Code scanners.

6 Unit Testing Plan

Unit Testing Plan pending modular design.

6.1 Unit testing of internal functions

Unit Testing Plan pending modular design.

6.2 Unit testing of output files

Unit Testing Plan pending modular design.

7 Appendix

No additional information to show.

7.1 Symbolic Parameters

None yet used.