Testing Plan

Group 3

June 7, 2022

Contents

1 Introduction **4**
1.1 Overview 4

2 Test Plan **4**
2.1 Overview 4
2.2 Testing Format 5
2.3 Unit Testing 5
2.4 Considerations 5

3 Functional Tests **5**
3.1 Message Display 5
3.2 2D Graphics Display 5
3.3 Image Display 6
3.4 Video Display 6
3.5 Tool List Display 6
3.6 Element Properties Display 6
3.7 Document Parse 6
3.8 Document Output 7
3.9 Event Handling 7
3.10 Script Engine 7
3.11 Tool Parse 7
3.12 Tool Handling 7

4 Test Records **8**
4.1 Information 8
4.2 Reports 8

Revision History

Revision	Date	Author(s)	Description
0.1.0	14.03.22	SSP526	Doc created in GDocs
0.1.1	14.03.22	DM1306	Fit to LaTeXtemplate
0.2.0	14.03.22	DM1306	Add further information to all sections. Refocus on product goals. Change overall test strategy to bottom-up approach.

1 Introduction

1.1 Overview

This document describes our Testing Methodology that will be used through the development cycle of our product. It will define Functional Tests for User Stories as found in the Functional Specification document, and a broad overview of our methodology for generating and applying automated Unit Tests.

2 Test Plan

2.1 Overview

At the highest level, our codebase can be split into two super-modules: The UI Controller and the Engine. This is a practical distinction, with the two running on seperate threads to prevent heavy processing blocking the UI; the two super-modules may be divided further into several modules each.
The UI Controller may be seen to minimally consist of:

- Message Display
- 2D Graphics Display (Including Text and Tables)
- Image Display
- Video Display
- Audio Player Display
- Tool List Display
- Element Properties Display

The Engine may be seen to minimally consist of:

- Document Parse
- Document Output
- Event Handling
- Script Engine
- Tool Parse and Handling

These modules may even be split several times further into their component parts; we shall start our testing with these, employing the common JUnit Test framework to run localised, automated Unit Tests - enusring that we have confidence in these parts as they become available prior to further high-level Functional testing. In a word, our strategy is “bottom-up”.

2.2 Testing Format

1. Refer to the appropriate Test if available.
2. Ensure that all testable dependancies have passed Unit Testing.
3. Build the software with the module to-be-tested included.
4. Enter the required input for the Test.
5. Compare the expected outcome with the actual outcome.
6. Record the result.

2.3 Unit Testing

Automated Unit Testing shall be applied to "lower-level" modules. Every deterministic and non-static public method of a Class should be tested for correctnes of operation through our Unit Test suite using a combination of random invalid and valid inputs, through a combination standard Unit Testing and "fuzzing" techniques.

2.4 Considerations

"Lower-level" modules have impacts on those above, and so flawed Unit Testing has the potential to invalidate further Functional Testing. This means that our Unit Test suite must be near-complete with high code coverage, to provide confidence in our semi-automatic and manual Functional Tests.

3 Functional Tests

3.1 Message Display

Description	Post individual blocking and non-blocking messages to the UI for display.
Purpose	Users require notification about certain events within the program. Test this function.
Inputs	Blocking message (Action-required message). Non-blocking message (Information message).
Expected Outcome	Messages containing the input text shall be displayed in the correct manner.

3.2 2D Graphics Display

Description	Post valid and invalid 2D graphics objects to the UI for display.
Purpose	The User may require that a certain 2D Graphical element be displayed. Test this function.
Inputs	Random 2D graphical objects.
Expected Outcome	Valid objects should display correctly on the Card/Page.

3.3 Image Display

Description	Post valid and invalid images to the UI for display.
Purpose	The User may require that a certain image be displayed. Test this function.
Inputs	Random images.
Expected Outcome	Valid images should be displayed correctly. Images at invalid sources should display "not found".

3.4 Video Display

Description	Post valid and invalid videos to the UI for display.
Purpose	The User may require that a certain video be displayed. Test this function.
Inputs	Selected Valid and Invalid Video sources.
Expected Outcome	Valid video should be displayed correctly. Videos at invalid sources should not be displayed.

3.5 Tool List Display

Description	Post tools to the UI for display.
Purpose	The User requires access to tools from the Tools menu. Test this function.
Inputs	Selection of Tools.
Expected Outcome	Entered tools are displayed in the Tools menu.

3.6 Element Properties Display

Description	Click on a visual element to show its' properties and Click off of the element to hide them.
Purpose	The User shall select a visual element, which should reveal its' properties in the Properties menu. Test this function.
Inputs	Mouse clicks.
Expected Outcome	An object's properties are displayed in the Properties menu on click on the object.

3.7 Document Parse

Description	Attempt to parse Valid and Invalid documents.
Purpose	The User shall open a document, and the engine will attempt to parse it. Test this function.
Inputs	Valid and Invalid presentation XML documents.
Expected Outcome	The parsed result of a valid document is returned. Invalid documents return nothing.

3.8 Document Output

Description	Try to save a presentation Stack Document.
Purpose	The User shall edit a document and then save it. Test this function.
Inputs	Graphically edited document.
Expected Outcome	Valid presentation file is written to the User's specified location.

3.9 Event Handling

Description	Post valid and invalid events to the engine.
Purpose	The User shall perform an action and the engine should respond. Test this function.
Inputs	User actions.
Expected Outcome	The Engine and Scripts should show the correct response to a valid event.

3.10 Script Engine

Description	Ensure that the Script Engine can execute scripts with the correct access to program data and methods.
Purpose	The User shall trigger an event associated with a script, which should execute as expected. Test this function.
Inputs	Test Scripts for each supported Langauge engine.
Expected Outcome	Scripts execute correctly.

3.11 Tool Parse

Description	Attempt to Parse valid and invalid tool documents.
Purpose	The User shall open the application which shall attempt to load a tool file. Test this function.
Inputs	Valid and Invalid tool documents.
Expected Outcome	Valid documents are correctly parsed. Invalid documents return nothing.

3.12 Tool Handling

Description	Post valid and invalid actions for tools.
Purpose	The User shall select a tool from the UI and use it, triggering an action. Test this function.
Inputs	User tool input.
Expected Outcome	Tool scripts are run correctly, having the correct effect on the UI.

4 Test Records

4.1 Information

4.2 Reports