Testing Plan

Group 3

June 8, 2022

Contents

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	Editorialise

1 Introduction

1.1 Purpose

This document will be used to describe the tests that will be carried out when integrating separate modules together to eventually create a final version of Super Pres. It will outline the testing plan we will follow and describe each individual test case for all module integration. Finally, at the end of the document, there will be testing reports, for each test case, that will be filled out once each test has occurred.

1.2 Overview

We will be using the testing method black box testing, so there will be an expected output for each specified input for each test case. If the actual output, from the test, is that same as that of the expected output then the test has passed, if not the test fails. Each test will consist of testing multiple modules functionalities when integrated with one another, specified by the testing plan, to prove that each module is still operational after being integrated to the final version of Super Pres; it is assumed that all unit tests have already been completed and passed prior to carrying out the integration tests in this document.

2 Test Plan

2.1 Overview

The items to be tested in this document are the individual modules coded when they have been integrated together to create Super Pres. For these integration tests we have chosen the 'Top-Down' testing method in which higher level modules are tested first and then the lower level modules. What makes modules higher or lower level is the amount that module contributes towards controlling the operation of the system. This is so that major faults, that would interpret the overall operation of Super Pres running, will be found earlier and can be fixed first such as the presentation playing. Then later down the line features such as a home page, which are needed in the final iteration of the product but aren't necessary for the operation of the main function, will be integrated and tested.

2.2 Testing format

1. Refer to the test case (this document).
2. Familiarise the expected output for the test.
3. Set up the system with all modules required for the test with corresponding input data.
4. Enter the required input from the test case.
5. Determine if the test was a pass or failure using pass criteria from the test case.
6. Fill out the test report.

2.3 Considerations

This method could potentially lead to the lower level modules being tested in a rush and potentially carried out poorly in comparison to the other tests. This is where careful time management for the entire project is important as to prevent this from occurring, however is a potentiality we cannot ignore and need to think of when planning the test cases. Some of the lower level modules have impacts on the higher level modules. The higher level modules are integration tested first and so this means that when integrating the lower level modules some re-testing of the higher level modules may be imperative in making sure that Super Pres still functions flawlessly. Alternatively, careful planning of the test cases to ensure when testing the lower level modules, all the functionalities that impact modules already tested are re-tested there.

3 Test Cases

3.1 Case 1

Description	Combine the slideshow module and the main window, which is just a generic application window currently.
Purpose	As stated, the current main window has no operation and is just a window. Adding the slideshow functionality will give the main window some purpose and is a main feature of the final application; it is controlled by later modules, that will be integrated, so needs to be added first. The slideshow should also follow the presenters controls at this point and so we will test that this occurs when merged with the main window.
Inputs	Opening a presentation Stack
Expected Outcome	The first Card should appear on the main window, displaying all contained content.

3.2 Case 2

Description	Merging the edit mode module and the main window.
Purpose	To see if enabling edit mode will uncover the edit controls and prove that these controls are operational when combined with the slideshow. This means that T1 has to be completed before T2 can take place.
Inputs	Enabling ‘edit mode’.
Expected Outcome	The first Card should appear on the main window, displaying all contained content, and be editable.

3.3 Case 3

Description	Integrate the cards module with the main window.
Purpose	Show that the user can add, edit and stack cards on top of the presentation slides that they are viewing at their discretion. Also, to show that these cards should remain linked to the slide that they're located on top of and only be visible by the user that created them; they shouldn't be visible to anyone else.
Inputs	Create multiple cards. Edit those cards. Stack the cards. Switch slides and return back to the slide the cards were on.
Expected Outcome	The cards should come into existence. Editing the cards should alter the information visible upon them. The user should see contents on the slide they were created and on no other slide. They should remain there when switching between slides.

3.4 Case 4

Description	Integrate the scripting module into the main view window. At this point T1, T2 and T3 have to have already passed.
Purpose	To see if the user can associate a script with their card stack and that their script is recognised by the program which adds the extra functionality into the program and operates what the user adds to the script
Inputs	A script (that already works and has been unit tested) needs to be added and associated with a card stack in the presentation. This script is to be a simple script with one functionality, such as a button. The expected inputs from this script (such as pressing a button) are required for this test.
Expected Outcome	The card should have the extra functionality as specified in the script (if it needs a button press, there should be a button visually displayed). Pressing this button should give the output as described in the script.

3.5 Case 5

Description	
Purpose	To see if the user can associate a script with their card stack and that their script is recognised by the program which adds the extra functionality into the program and operates what the user adds to the script
Inputs	A script (that already works and has been unit tested) needs to be added and associated with a card stack in the presentation. This script is to be a simple script with one functionality, such as a button. The expected inputs from this script (such as pressing a button) are required for this test.
Expected Outcome	The card should have the extra functionality as specified in the script (if it needs a button press, there should be a button visually displayed). Pressing this button should give the output as described in the script.

4 Test Reports

4.1 Information

The test reports in §4.2 co-exist directly with the test cases specified in §3 via their case number. The test reports are exclusively for reporting passes or failures; to see details of the tests see §3.

4.2 Reports