Testing Document

Table of Contents

Introduction

To ensure that out application is running to what we expect, we created a series of testing cases that we fell are sufficient in order to determine the performance of the application. A lot of our program revolved around a Graphical User Interface and therefore can be hard to test through normal testing practices such as JUnitCases. We decideded that the best course of action to test our application is visually with an expected to actual test situation. With this we describe what should happen and what did happen. We will also do a coverage case to verify that our application has good coverage.

Below will go through what we believe are the key core tests. We encourage any user who experiences a bug to report it to my personal email (khallman@my.yorku.ca) and we will investigate it right away.

Implemented Tests

1. **Creating a new scenario**

Expected	**Actual**
In the main GUI you will click on the New button. A smaller window will appear with options to enter the file name, number of cells and number of buttons. Here we expect to enter the name of the file and the cells and buttons. If the cells and buttons are not integers we also expect an error message to occur. After selecting the save and continue button a new window should appear with the scenario builder GUI. The file will also save to the directory. The last thing is	When clicked the new window appears with all the options as stated. You can enter the name of the file and the number of cells and buttons. When you enter something that is not an interger into the cell or button field a pop message asking you to change the value occurs. After clicking save and continue the new window opens and adds the cell and button information to the display. The file is also created in the saved scenario folder. The drop box also contained

that the drop down box that shows what answer to set the button to contains the right amount of buttons as the user inputted.	the same amount of buttons the user inputted.

Pass or fail

We consider this a passing case. It did exactly as followed and had no issues.

Importance of test case

The test is important since it is the fundamental of the whole application. Without it the file would not be saved nor would the user be able to create new scenario files.

2. **Editing a scenario**

Expected	Actual
When the edit button is clicked on the main menu a file choose should open up. The file chooser should default to the saved scenario files. From there the user should select the scenario file they want and it will open in a new window with all the information that is on the file displayed.	When clicked the file chooser opens up and the user can select the file. It only shows the options of .txt files as well. When the file is selected and the user clicks ok, it is uploaded into the scenario editor and the user has full functionality of the file.

Pass or fail

We consider this a passing case. It did exactly as follow and had no issues.

Importance of test case

The test is important since it is also a fundamental of the whole application. Without it the user would not be able to edit existing files that have created in the past.

3. Running/Testing a new scenario

Expected	Actual
When clicked on run or test button a file chooser menu will appear. With the file chooser menu you can select a file from the saved scenario files. The simulation will then appear.	When clicked on the button a the file chooser opens and allows you to select a file. When the user hits ok and error has occurred and the simulation does not open, however the error is wrote to the errorlog file.

Pass or fail

We consider this a failing case. The file chooser did open correctly however we have found that some simulation files are not being read properly. In the next release (V3.0) we will fix this issue.

Importance of test case

The test is important since it allows the user to run and try their file.

4. Shortcuts and Hotkeys

Expected	Actual
The application offers functions that can be accessed using short cuts. We expect that for every short cut we have made in the program the proper execution was done. To test we will click every shortcut and insure that all functions were done.	When testing all shortcuts they all did their desired actions. They also followed all conditions such as the finish condition on the scenario builder.

Pass or fail

This is a passing test. All shortcuts behaved the way we wanted them too and did so even with error handling.

Importance of test case

The test is not to important however for people who may be visually impaired it allows them to access common buttons with ease.

5. **Clicking on the finish button**

Expected	Actual
The application will add all fields entered into the display and also add the files into the queue of the editor class. There should also be a behaviour for the finish button in which the minimum amount of information required is to set the cells to display something. To test the editor class we used an output statement for the line. We also tested it by using the save button, which will be talked about later on, and checked the file.	We added in no information and nothing was added to the display and editor and received a popup message saying we require the display input field. We then added information to the fields and clicked the finish button. When clicked the information is put on the display and a series of print statements on the console is executed with the information. After clicking the save button and checking the file the information lines up with what the user entered.

Pass or fail

This is a passing test. The finish button had the correct error handling and added everything to the file/editor accordingly.

Importance of test case

The test is important since it is what allows the users to enter their own information build up the scenario file.

6. **Clicking on the Insert button**

Expected	Actual
The application will add the field located next to it where the user inputs the text. It is then displayed on the display and added to the editor queue. To test the editior queue we used an output statement.	When clicked the information is added succsufully to the application. An output statement from the editor also occurs with the correct information. The display also has the correct information.

Pass or fail

This is a passing test. The inset button behaved properly and added the text to the file.

Importance of test case

The test has a little bit of importance as it gives the user the option to enter in text-speech to the file, however it is not a test in which if it failed it would stop the functionality of the application.

7. **Clicking on the Audio button**

Expected	Actual
The application will display a popup with asking the user if they would like to insert existing audio or create their own. If the user creates their own they are directed to a new window that as audio creation options (a test for these will be later). If they choose existing a file choose opens up with the directory being under Audio Files and the user can select a wav file. The window then closes and the file is added to the application.	When clicked the popup is displayed correctly and shows the proper option. If the user clicks the create new a file, a new window opens with options to create a file. If add existing audio is selected, then a file chooser will open up under the correct directory. After you selected the audio and hit ok the audio is added to the display and editor queue.

Pass or fail

This is a passing test. The add audio button behaved the way it was suppose to and had no issues.

Importance of test case

Similar to the insert text-speech this button is somewhat important as it allows the user to add an audio, whether created or added, to the project. It is an important feature but does not prevent you from not being able to create or edit the scenario files.

8. Clicking on the Save button

Expected	Actual
The application will save all the files that are in the display to the file itself. Then will display a popup saying save successful. To test the editor we used a series of output statements to insure it was being wrote as well as checking the file it self.	When clicked saved the popup message happened. At the same time you could see all the outputs being displayed in the terminal correctly. When looking at the file it was saved and updated successfully. Important to note that it overwrite the old information and updated it rather then appending it which is what we want.

Pass or fail

This is a passing test. The information on the display was added to the file successfully and all cases.

Importance of test case

This test is very important as without it the information that the user wrote about would not be displayed correctly the user would not be able to test their information.

9. Clicking the edit button

Expected	Actual
The edit button should be disabled into a selection is made on the display. Once an item is on the display the edit button is now clickable. When clicked a popup message will occur that entails specifically to the element you clicked. Within the popup you change what you have inputed to fix it. After selecting ok it will add it to the display and the editor class, with the proper position.	When clicked a popup message occurred. I tested it for all possible inputs available from the scenario builder. Each displayed their own unique popup. The information was then changed and ok was selected. The display was updated with the correct information as well as ant parts of the builder i.e. the button selector was also changed. I used an output command to verify that the edit has been updated to the new input and it was.

Pass or fail

This is a passing test. The information inputted was updated properly to the display and the editor class. Also every type of input had a unique popup.

Importance of test case

This test is very important as it allows the user to fix mistakes they may have inputted into the builder.

10. Clicking the delete button

Expected	Actual
When the delete button is pressed the display should delete the element and the editor should also remove it from the list.	When the button is pressed the currently selected field on the display is deleted. I verified that it was removed from the editor class by using an output command and checking that it is deleted. Since the element was deleted I had to check the surrounding elements and to see if it had been deleted. Also if

	you check the file after hitting save it is deleted.

Pass or fail
This is a passing test. The information was deleted properly and verified by checking the file and the editor using output commands

Importance of test case
This test is very important as it allows you to delete unwanted lines of for your code.

Overall Testing

While doing the main focused testing like we did in the test cases we also observed more smaller things to insure the application was functioning. Some of these small things include errors popping up when something can't be done, like delete a cell or button field. It was also testing to see if when you edited something it remained consistent in the class and all variables that are updated should be updated. For example changing the number of buttons should result in the field where you select the button to also change. It was important to us that we are able to identify smaller issues and test them as well as the larger issues talked about above. Another smaller test that we felt was important was checking the accessibility functionality. We used NVDA to test that our code that we implemented for the accessibility was working, which it was. This we felt was a small but important test as we want to insure that any user can use our application.

Coverage Testing

We set the goal of our application having a minimum test coverage of 50 percent using the Eclemma in eclipse. We found that our test coverage was around 64 percent. We tested multiple times to insure we were able to cover as much as possible. The average was around 64 percent. For us the importance of the coverage test was to insure that we tested every button and interaction. One issue that come with the converge testing was jumping back to the main menu after doing one task which made it hard to continue testing. For the next deployment we are going to focus more and creating a better flow in our application that makes a smooth transition from one method to the next. Ideally we would like to have our coverage around 75-80 percent with the next deployment (V3.0).