# TalkBox Testing Document

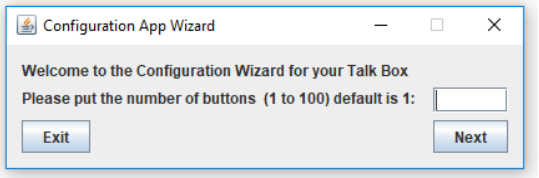
Implemented Test Cases:

1) 1 Button GUI:

This test case is designed to determine how the simulator will behave if only one button exists in the configuration. The main emphasis of the test case was to see how the image and button were affected and proportioned across the frame, and if the sound functionality was maintained.

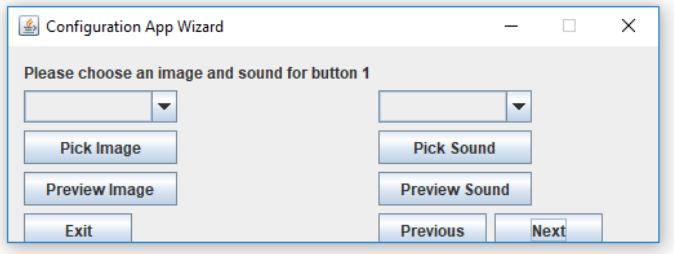
**Testing Process:**

1. Run the MainConfiguration app. The configuration window opens:



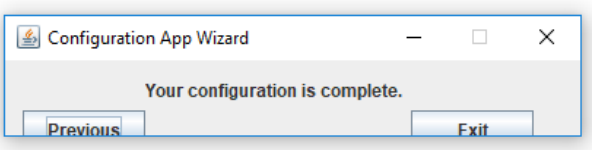

2. The user specifies in the text field the number of buttons desired. For this test case it is 1.

3. The following window appears after clicking “Next”. This is where the user can specify the images and sounds for the specific button. This window will appear as many times as the user specified in the first step.




4. Here the user can upload their image or pick any sound. The user can also record the sound. To make sure that the image and sound were successfully uploaded we preview both:


5. We then click on next and this final window appears since we only chose to configure one button:



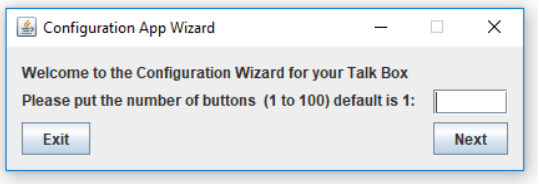

6. Then we run the simulator frame to make sure that the test was successful, we obtain this frame, and the audio that the user picked plays upon clicking the button:


2) 4 Buttons GUI:

This test case is designed to determine how the simulator will behave in a “default” scenario, with four buttons existing in the configuration. The main emphasis of the test case was to confirm that the frame supported the default case we had designed for our TalkBox. Note that the default case of 4 buttons is formed from the TalkBox originally shown during the project introduction.

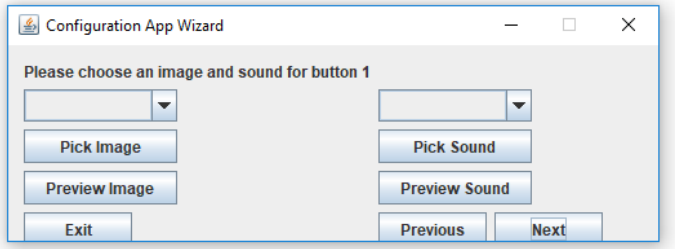
Testing Process:

1. Run the MainConfiguration app. The configuration window opens:

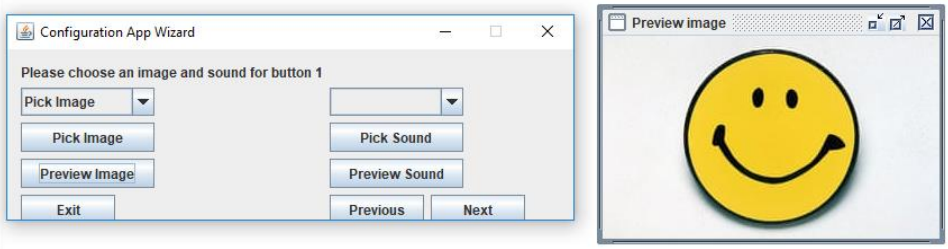



2. Second, the user specifies in the text field the number of buttons desired. For this test case it is 4.

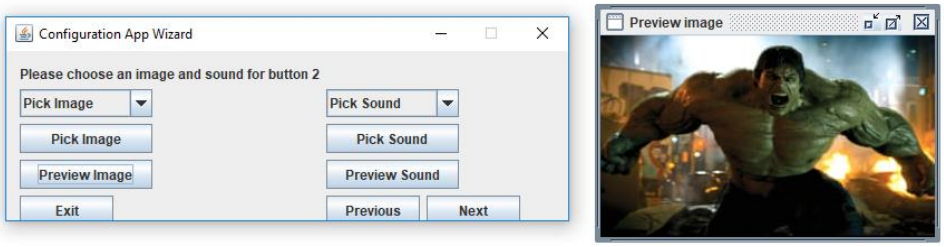
3. The following window appears after clicking “Next”. This is where the user can specify the images and sounds for the specific button. This window will appear as many times as the user specified in the first step.



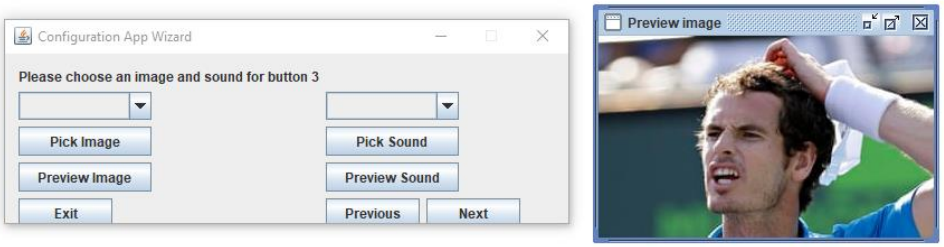
4. Here the user can upload their image or pick any sound. The user can also record the sound. To make sure that the image and sound were successfully uploaded we preview both:



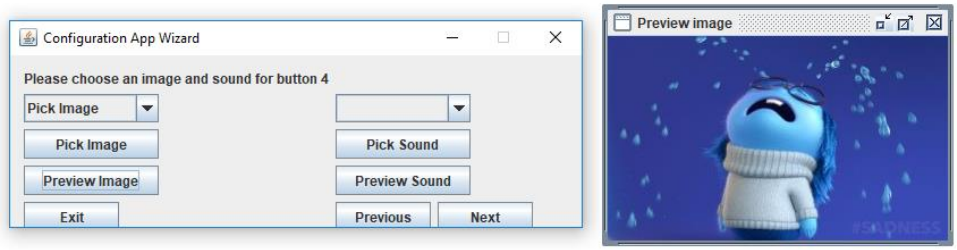

5. After clicking next, the user will be taken to the exact same window to configure the next button. This will occur four times until all buttons are configured. These are the results:
   a. Second Button:



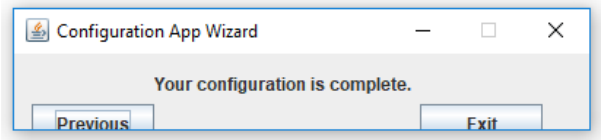

   b. Third Button:




   c. Fourth Button:




6. We then click on next and this final window appears, since we finished configuring all buttons:



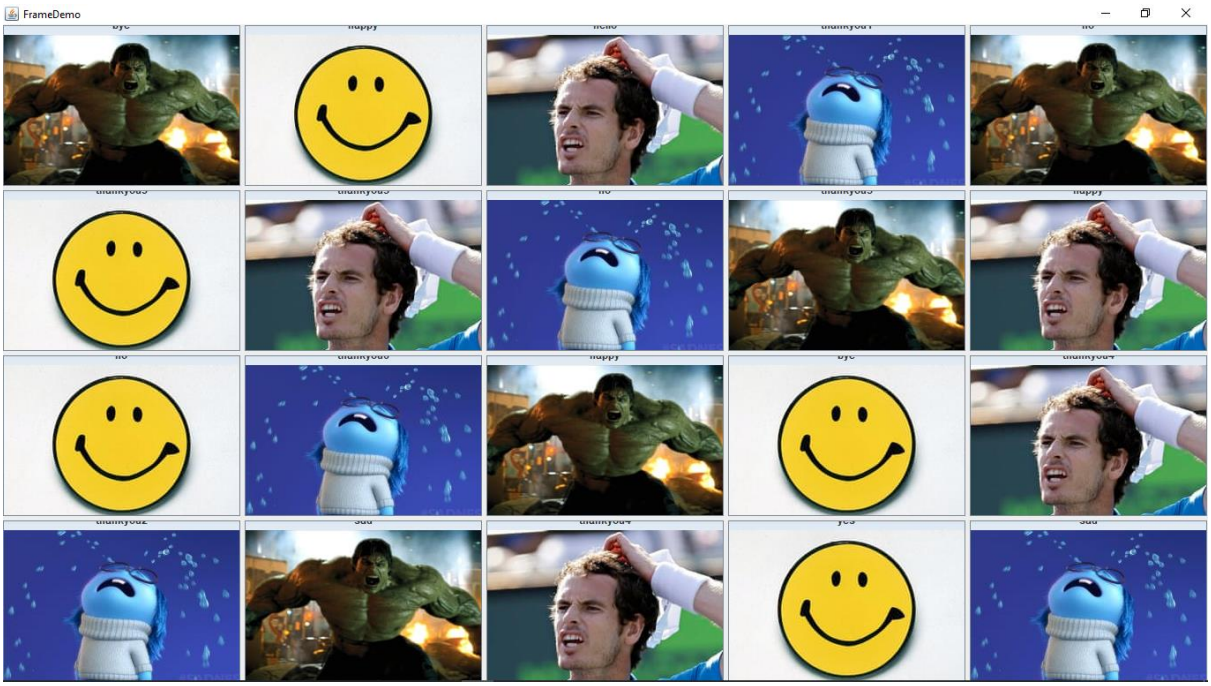
7. Then we run the simulator frame to make sure that the test was successful. We obtain this frame, and the audio that the user picked plays upon clicking the buttons:


3) 20 Buttons GUI:

This test case is designed to determine how the simulator will behave with an extreme case, namely 20 buttons existing in the configuration. The main emphasis of the test case was to see how the image and button duo were affected and proportioned across the frame when so many buttons must fit in it. It was also important to see if the sound functionality was maintained when so many buttons required to have it in the frame at once. Finally, it was vital that the simulator maintained productivity despite the grandiose addition of functioning visuals and interactions.

Testing Process:

1. We follow the same process for the second requirement, however this time during the 2nd step we specify that we want 20 buttons and for the 5th step we will configure 20 buttons. Some of the pictures might be used more than once due to the amount of buttons and the available default test files:



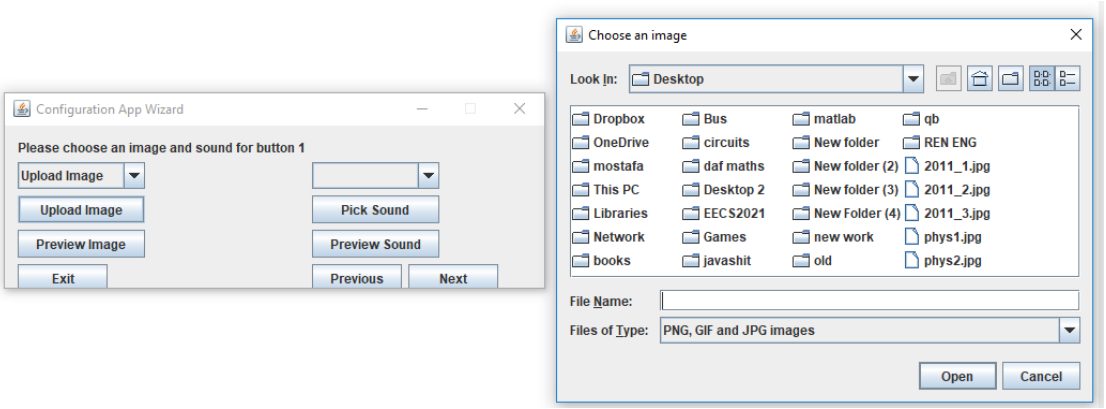
4) Invalid Buttons GUI:

This test case is designed to determine how the simulator will behave with invalid buttons, namely using invalid inputs for audio and/or images. The main emphasis of the test case was to see how the image and button duo were affected when one or both were invalid, and how the simulator frame would react to invalid buttons. We wanted to confirm or deny if it would crash and if anything “weird” would occur.

Testing Process:

1. We will test this using only 1 button configuration and see how the program behaves.

2. For the step where the user needs to choose an image and sound, the window that opens (depicted below) doesn’t show files that don’t match the intended format. Thus, it prevents the user from choosing an incompatible file.




3. As an additional safety implementation, the “Next” button doesn’t respond if the user does not set the image and sound files. This will prevent the user from mistakenly pressing next without having configured the button. However, the user can still go to previous buttons and alter specifications if desired.

Sufficiency of Test Cases:

The tests are sufficient because they cover the various situations that occur. Case 2) is the “base case”. It is what should happen at default and ensures what to expect from practical usage of the TalkBox. Cases 1) and 3) are both extreme cases on opposite ends of the usage spectrum. They shed light on how the simulator will react in specific unique situations, which validate the significance of these tests. Case 4) is important since it is the “fail” case and will show the effects of invalid input. This case allows for us to decide how we should handle these invalidities internally.

Coverage:

| Element | Covera... | Covered Ins... | Missed Instr... | Total Instruc... |
|---|---|---|---|---|
| TalkBox | 51.0 % | 2,222 | 2,131 | 4,353 |
| src | 51.0 % | 2,222 | 2,131 | 4,353 |

Coverage was found to be 51.0%. This was due to the JUnit testing only covering specific functionality of the configuration app and simulator app. The missed scenarios and features will be tested and allow the coverage to be greater than 75% (preferably upwards of 90%) for the final submission.