

TalkBox Testing Document

Implemented Test Cases:

1) TalkBoxMain Test:

This test case ensures a version of the TalkBox home screen can be instantiated and opened. This test case is important since it ensures future tests can proceed. The use case actor associated to this test is the “programmer” of the TalkBox app since it is a test that occurs before specific cases can be tested.

2) TalkBoxListener Test:

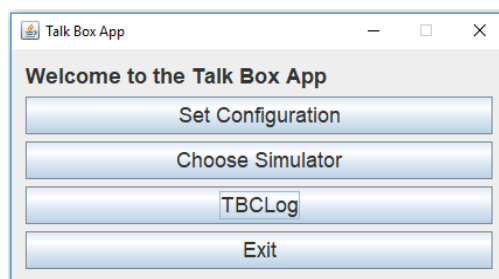
This test case ensures the TalkBox listener can be instantiated. This test case is important since it allows for all logging to be tested later. The use case actor associated to this test is again the “programmer” since it too is a test that occurs before specific cases can be tested.

3) 1 Button GUI:

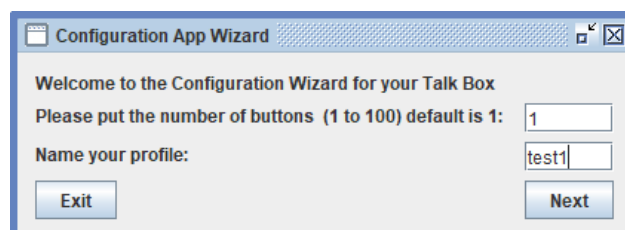
This test case is designed to determine how the simulator will behave if only one button exists in the configuration. The main emphasis of the test case was to see how the image and button were affected and proportioned across the frame, and if the sound functionality was maintained. The use case actor associated to this test is the “administrator” of the TalkBox since they are the client that will be configuring the app in its practical use environment.

Testing Process:

1. The administrator runs TalkBoxMain. It opens as so:



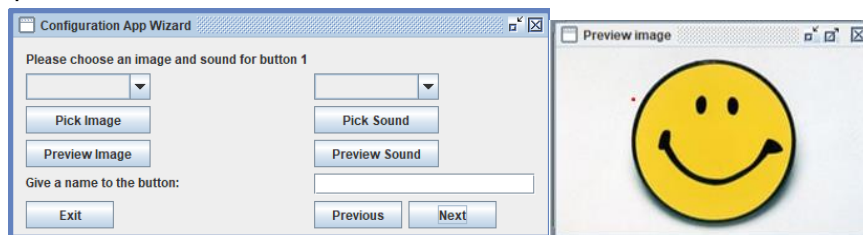
2. The user selects 'Set Configuration' and is taken to the Configuration setup home page:



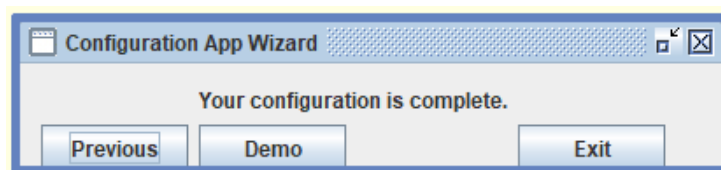
3. The following window appears after clicking “Next”. This is where the user can specify the images and sounds for the specific button. This window will appear as many times as the user specified in the first step. Notice that the button number is displayed for the user to reference.



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:



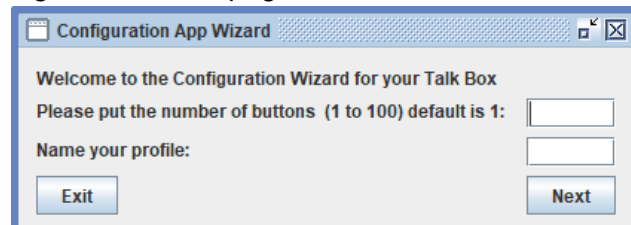
2) 4 Buttons GUI:

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

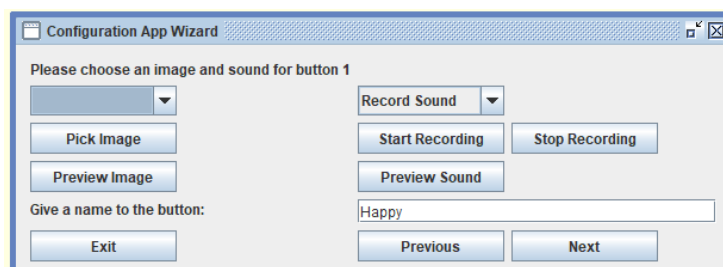
confirm that the frame supported the default case we had designed for our TalkBox. Note that the default case of 4 buttons is formed from the TalkBox originally shown during the project introduction. The use case actor associated to this test is the “administrator” of the TalkBox since they are the client that will be configuring the app in its practical use environment.

Testing Process:

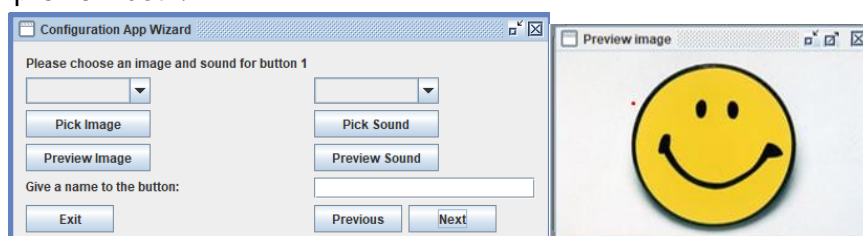
1. Repeat the same step 1 as above with the TalkBoxMain frame. They end up once on the configuration home page



2. Second, the user specifies in the text field the number of buttons desired. For this test case it is 4. They also enter an appropriate name for the configuration.
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.

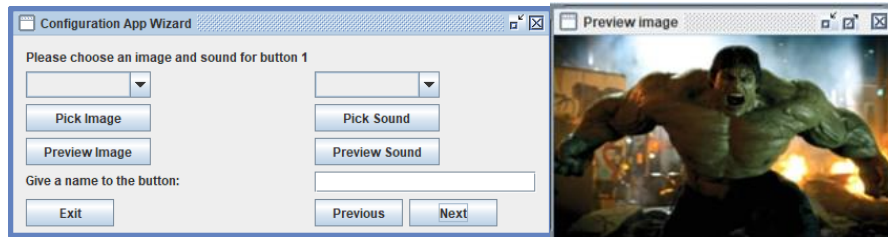


4. Here the user can upload their image or pick any sound. The user can also record the sound. To make sure that the image and sound were successfully uploaded we preview both:

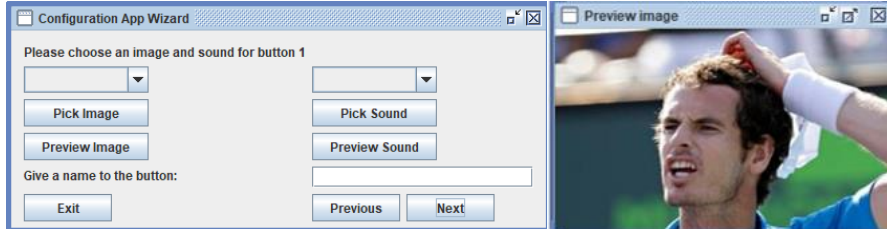


5. After clicking next, the user will be taken to the exact same window to configure the next button. This will occur four times until all buttons are configured. These are the results:

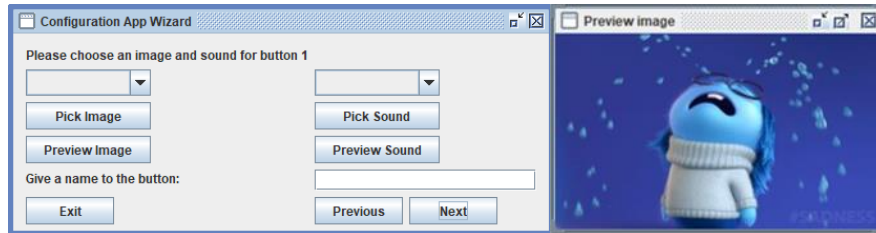
a. Second Button:



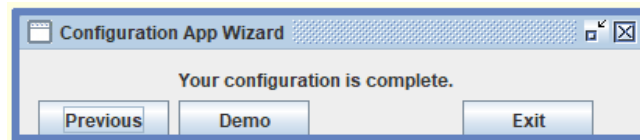
b. Third Button:



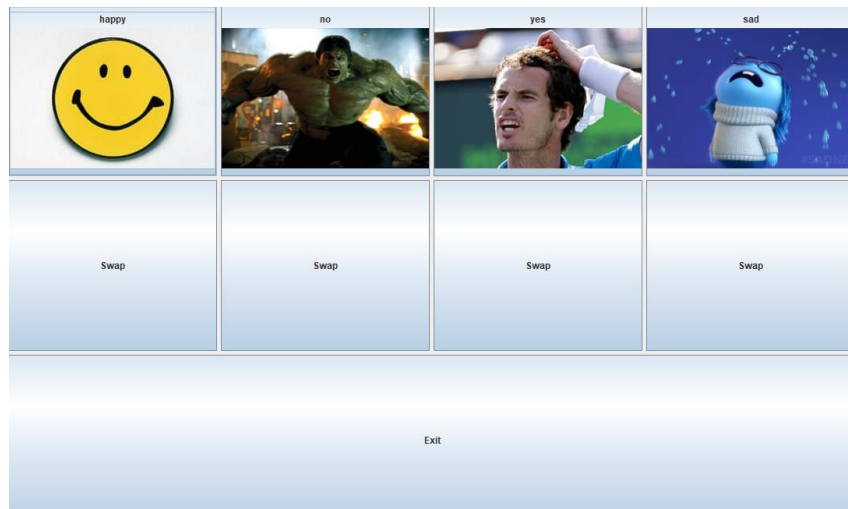
c. Fourth Button:



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:

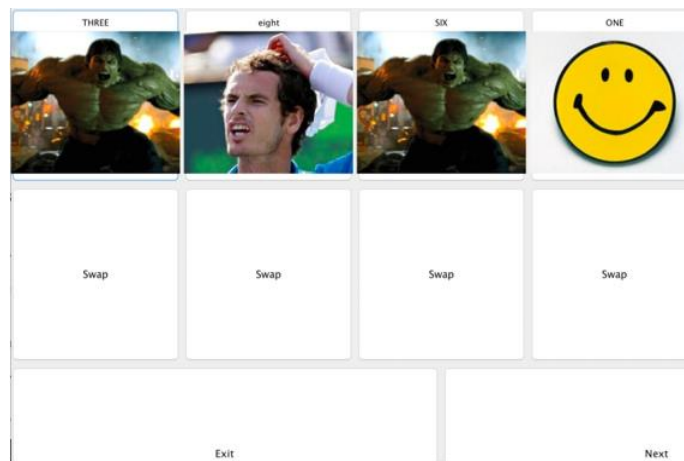


3) 20 Buttons GUI:

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. The use case actor associated to this test is the “programmer” since they would want to ensure that special cases are covered in their app and it will react appropriately to them.

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 number of buttons and the available default test files:



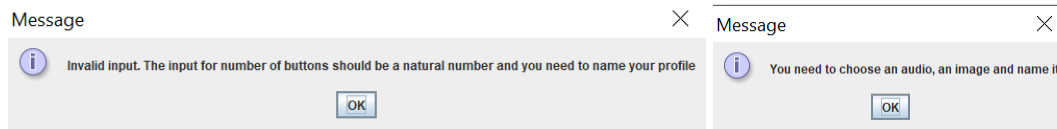
The key difference in this simulator frame is the population of the next button. Pagination allows for numerous buttons to be split by page, allowing for a cleaner design.

4) Invalid Buttons/Input 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. The use case actor associated to this test is the “programmer” again since they would want to ensure that special cases are covered in their app, especially ones substantial enough to cause lost in use.

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.



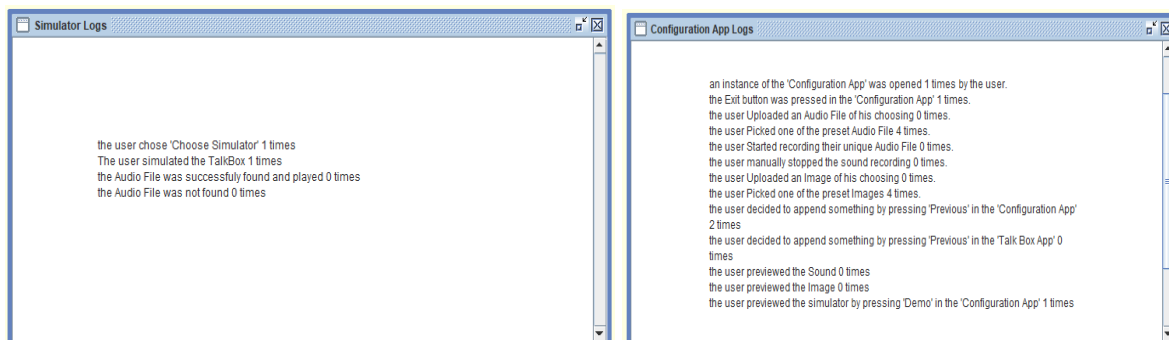
3. As an additional safety implementation, an error message is presented to the user. This allows for them to know their action is illegal and must be changed. Two versions exist and explain the exact error occurring. However, the user can still go to previous buttons and alter specifications if desired.

5) Simulator/Configuration Logger Test

This test case is designed to test the logger functionality works as desired for both simulator and configuration logs.

Testing Process:

Follow the testing process described in test case 2 (4 Button GUI), but instead go back to the home TalkBox frame and select 'TBC Log'. Clicking this will open two frames for each logger, confirming they work as desired. The screenshots below visualize this:



Sufficiency of Test Cases:





The tests are sufficient because they cover the various situations that occur when using the app. 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. Case 5) tests the last requirement feature, confirming it does log the actions as desired.

Implementation Process:

The test cases were implemented using JUnit testing (version 5). A version of the app was instantiated and ran. Assertions were made on various functionalities throughout its run process to confirm that they indeed functioned as desired. These assertions included checking action listeners, specific boolean values (simClicked for example) to see if changes had occurred throughout the completion of the test. Examples of assertions used were assert equals, assert true, and assert false. Each step of a use (from configuration to logging) was checked to ensure the validity of the TalkBox implementation.

Coverage:

Element	Covera...	Covered Ins...	Missed Instr...	Total Instruc...
▼  TalkBox	 55.5 %	4,161	3,333	7,494
>  src	 55.5 %	4,161	3,333	7,494

Coverage was found to be 55.5%. This was due to the JUnit testing covering functionality of the TalkBox app but not the background functionality (i.e. fileLoader class) that is only called on for other uses. The final submission saw a 4.5% increase from the midterm submission, with more features being added and covered by test cases.