**TalkBox Testing Documentation**

EECS 2311 – Midterm

Bryan Barcos Caro – 215876139

Param Balaganeshan – 215686694

Dhruv Patel – 215910011

**Table of Contents**

**Test Method Page Number**

Play Button……………………………………………………………………………...………..2

Stop Button…………………………………………………………………………………..…..2

Pause Button………………………………………………………………………………..……2

Resume Button……………………………………………………………………………..……3

Reset Button……………………………………………………………………………………...3

Swap Button……………………………………………………………………………………...4

Save Changes Button…………………………………………………………………………...4

Add Initial Button………………………………………………………………………………...5

Remove Initial Button……………………………………………………………………………5

Add Final Button…………………………………………………………………………………6

Remove Final Button…………………………………………………………………………….6

Launch Simulator………………………………………………………………………………..6

Find Files…………………………………………………………………………………………7

1. Test Click Play Button

The purpose of this button is to begin playing an audio file that has been selected by the user.

To ensure an audio file is correctly played, the setSelectedIndex() method is used to select an arbitrary audio file. Once selected, the Play button is clicked using the doClick() method. When running the Junit test, one can observe the sound being played if the test is passed.

To pass this test, the audio selected must begin playing after the Play button is clicked.

This method has a coverage of 29.6% in the entire project, and a coverage of 56.4% within the class ConfigurationAppGUI.java.

1. Test Click Stop Button

The functionality of the Stop JButton is tested in this test case. The purpose of the Stop button is to stop any audio that is currently playing.

To test if the stop button is fully functional, a file is selected and played. Once played, before the audio ends, the Stop button is clicked using the doClick() method. One can observe the audio file stop suddenly during testing.

To pass this test, the Stop button must successfully stop any file currently playing. If the Play button is clicked after the stop button, the audio file must be played from the start rather than resumed.

This method covered 33.3% of the entire project code and 58.7% of the code from the ConfigurationAppGUI.java class.

1. Test Click Pause Button

The purpose of the Pause button is to stop the audio file from playing, but contains the ability of resuming the audio file instead of starting from the beginning afterwards.

To test the functionality of the Pause button, a file is selected and played as done in the Play button test. Once an audio file is playing, the Pause button is clicked before the end of the file has been reached. After the audio output has stopped playing, the play button is clicked once again to check if the output is played from the correct place.

To pass this test, the Pause button must stop audio output when clicked without resetting the audio file to the beginning, providing the user with the ability to resume the audio output.

This method covered 33.2% of the entire project code and 58.7% of the ConfigurationAppGUI.java class.

1. Test Click Resume Button

The purpose of the Resume button is to allow the user to continue playing an audio file once it has been paused.

Testing the functionality of the Resume button requires for an audio file to be played using the Play button, the audio file to be paused using the Pause button before reaching the end of the file, and then finally click the Resume button using the doClick() method.

To pass this test, the audio file must continue playing from the same place where it was paused rather than playing from the beginning.

The coverage for this method is 33.8% of the entire project and 59.1% of the ConfigurationAppGUI.java class.

1. Test Click Reset Button

The purpose of the Reset button is to allow the user to return to the default setting of their audio list. The default setting is having only 3 buttons available and an empty final list.

To test the Reset button, the application must return to default settings after being clicked. The methods getNumberOfAudioSets() and getNumberOfAudioButtons() are used, asserting that the values are 4 and 3, respectively. To further test the Reset button, 2 audio files are added to the final list and 2 buttons are also added. Once added, the Reset button is clicked and we once again assert the number of set and buttons return to their default. Also after clicking the Reset button for a second time, the audio array list is compared to the default list, and check that the final list is empty.

To pass this test, the number of audio sets after clicking reset must be 4, the number of audio buttons must be 3, and the final list must be empty. The initial list must also return to its default setting even after adding buttons.

This method has a coverage of 36.3% for the project and 66.4% of the class ConfigurationAppGUI.java.

1. Test Click Swap Button

The Swap button allows the user to cycle to the next audio files, where is equal to the number of buttons currently available.

To test the Swap button, the reset button is clicked to bring the application to its default setting of 3 audio files in the initial list. The first three audio files are added to a list prior to clicking the Swap button then compared to the files currently in the final list to ensure equality. Next, the following 3 audio files are placed in a list. This list should be equal to the final list after the Swap button is pushed. This same process is done for a second time to ensure accuracy and consistency.

To pass the test, the Swap button must correctly cycle through to the next available audio files when clicked.

Coverage for this method is 36.9% of project and 62.7% of ConfigurationAppGUI.java.

1. Test Save Changes Button

The Save Changes button allows the user to save their selected final list in the order they choose. Once saved, the TalkBox application should store this setting, making their saved final list their new initial list both after saving. Opening a new TalkBox application will also contain the saved final list as the new initial list.

To test the Save Changes button, the TalkBox application is set to its default setting using the Reset button. Once reset, a selected audio file is added to the empty final list and an attempt to save is made. We must ensure this save is not successful since there are currently 3 buttons available but only one is set in the final list. Two more audio files are placed in the final list and the Save Changes button is clicked using the doClick() method. Using an assertion method, initial list after saving is compared to the final list prior to saving. To further test this method, a new TalkBox is constructed and the initial list of the new TalkBox is also compared to the prior final list.

To pass this test, the Save Changes button must successfully store the final list prior to saving into the initial list field. The Save Changes button must not alter the initial list if the user attempts to save with fewer items in the final list than buttons made available.

Coverage of entire project is 40.2%, 73.8% coverage for ConfigurationAppGUI.java.

1. Test Add Initial Button

The Add Initial Button allows to user to add more buttons to the TalkBox simulator if needed. The user can add as many buttons up-to the number the audio files are available.

Testing the Add Initial button requires knowing the amount of audio files stored inside the initial list. To know this, we use the Reset button to start the testing with 3 files inside the initial list. Then, the Add Initial button is clicked, increasing the size of the initial list by one. This same pattern is done twice more for consistency. Next we test the method at the limits. When attempting to add more buttons than there are audio files, the initial list must not increase and must prompt the user of this failure.

To pass this test, the TalkBox must successfully add a new button for each time the Add Initial button is clicked. The TalkBox must also not allow the user to add more buttons than audio files available, in which it must pass an assertion test indicating the number of buttons has not been altered.

This test covered 33.4% of the project code and 60.2% of the ConfigurationAppGUI.java class code.

1. Test Remove Initial Button

Remove Initial Button allows the user to remove buttons if needed. The user can remove as many buttons as needed until there are 3 buttons left available.

First, the test method uses the Reset button to set the initial list size to 3. The doClick() is used to add 3 buttons, then the Remove button is tested. Knowing there are 6 buttons, we test for the correct decrease in the number of buttons after each click. The method is also tested for the removal of too many buttons, attempting to remove a button when there are only 3 buttons left.

To successfully pass this test, the number of buttons must decrease by one for each click of the Remove Initial button. If there are only 3 buttons available, the number of buttons must not decrease when the button is clicked and should maintain its current value.

The coverage for the entire project is 33.5% and 62.0% of the ConfigurationAppGUI.java class.

1. Test Add Final Button

The Add Final button is meant to allow the user to add a selected audio file to their final list.

Testing this method requires checking the size of the final list for each time an audio file is added. First, we use the Reset button to ensure the final list is empty. Two selected audio files are added to the list, and the increase in size is asserted. The Add Final button must not allow the user to add the same audio file twice to the final list, so there should not be any increase in size if attempted. The list should also not increase in size if the user attempts to add more audio files than there are buttons available.

To pass this test, the size of the final list must not change when attempting to add more files than there are buttons and when the user attempts to add the same file more than once.

32.9% of the TalkBox project is covered with this method and 60.1% of its main class is covered.

1. Test Remove Final Button

Remove Final button gives the user the ability to remove audio files from the current final list.

Each time the Remove Final button is pressed, the size of the final list must decrease by one. After clicking Reset, making the final list empty, 3 audio files are arbitrarily added to the final list. After clicking the Remove Final button, the size is checked for each to ensure correct behaviour is displayed. When the list is empty, clicking the Remove Final button must not decrease the size of the list.

Passing the test requires that only one audio file is removed each time the Remove Final button is clicked, and that the size of the final list never reaches less than zero.

This test method covers 33.0% of the project and 61.1% of the project’s main class.

1. Test Launch Simulator

The simulator is a function constructed to allow the programmer to test his/her configuration application. The simulator will contain the same behaviour as the hardware used by the user.

Clicking the Launch Simulator button will open a new frame with the options of either playing and audio file or swapping to a different set. To ensure the frame is active, the isActive() method is used to ensure a new window opens. When the Reset button is clicked prior to launching the simulator, the number of buttons in the initial list should be 3. Since there are currently 12 audio files in total, the number of sets available should be 4. Using methods inside the SimulatorApp class, the consistency of the simulator is tested by checking the number of buttons available, total audio files, and number of sets. The swap button is clicked to check functionality and an audio button is also clicked.

To pass this test, the Launch Simulator button must successfully construct a new frame containing a simulator which has the same behaviour as the hardware made for the user.

This method has a coverage of 40.9% of the entire project and covers 58.4% of the ConfigurationAppGUI.java class.

1. Test Find Files

The findFiles(DirectoryName, Suffix) method from the ConfigurationAppGUI class is meant to find audio files from a directory, with the option of returning those files that end with a desired suffix.

To test this method, the listFiles() method from the Java File class is used to compare the results of the findFiles method. The files that end with ‘wav’ are selected first in order to test when a suffix argument is passed. At the same time, all files within the known directory are passed to a different array to compare when a null suffix argument is passed. If a directory which either does not exist or is not a directory is passed, then the method should return null.

To pass this test, the findFiles() method must correctly return the audio files that are in any valid directory. If a suffix is passed, the findFiles() method must only return those audio files that end with the suffix, such as ‘wav’ or ‘mp3’.

This test method has a total coverage of 31.8% and a coverage of 58.3% for the project’s main class.