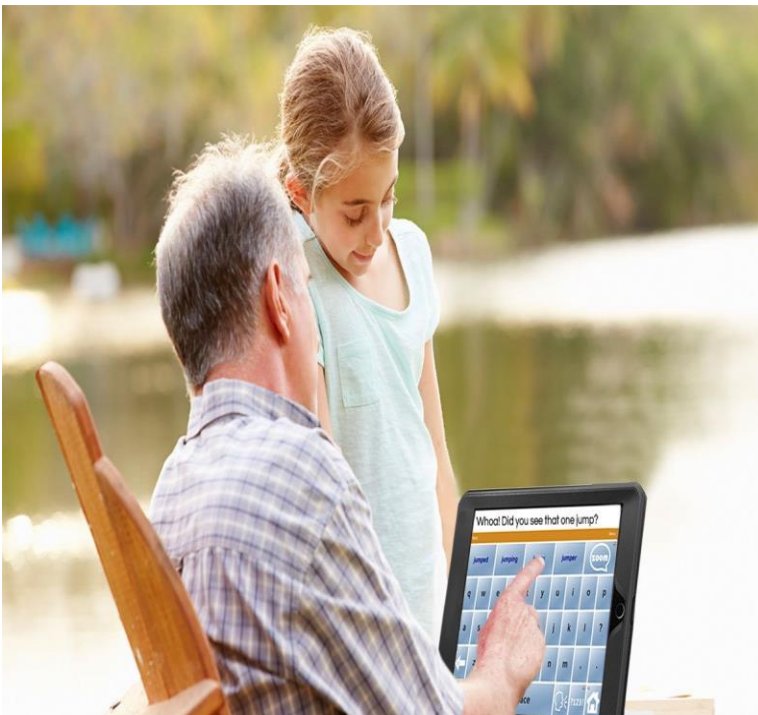

TALKBOX TESTING



APRIL 3, 2019

VERSION 2.0

GROUP 9

Authored by: Neharika Puri, Eric Pham, Yonis Abokar

Table of Contents

Introduction.....	3
Unit Testing.....	3-4
GUI Testing.....	4-9
Coverage.....	9-10

1. Introduction

1.1 Purpose

The purpose of the testing document will keep track of any bugs through testing and debugging. The document will track through the use of test cases and test coverages. The documentation will discuss each test and explain its significance in its finding. The documentation will also discuss the significance of the test coverage.

2. Unit Testing

2.1 Test Method

The unit testing will be done using the latest JUnit framework, JUnit 5. The functionality of each class was tested as individual methods.

2.2 Test Cases

Test Class/ Tester	Test Case Description
AudioClip/ AudioClipTest	AudioClipTest tests if the audio successfully plays. It tests this by using the “boring” wav file as a test file.
FileInputOutput/ FileInputOutputTest	FileInputOutput tests path finding and file retrieving
Serializer & TalkBoxConfiguration/ TestSerialization	Description- TestSerializtion will test whether the configurator successfully serializes and deserializes the data in TalkBoxData.tbc using a specific configuration.
Sound/ TestSound	Description – Sound will test the recording feature
AudioSampleList/AudioSampleListTest	Tests if the AudioSampleList class is able to add the audio files to different profiles

2.2.1 Test Case Derivation & Significance

i) AudioClipTest

Derivation: AudioClipTest comprises of two tests; testAudioClipStartsSuccessfully & testAudioClipThrowsException. These tests were derived due to errors received when the path to the audio files were incorrect.

Significance: Successfully passing the AudioClipTest signifies that the TalkBox device will be able to successfully play audio through the correct path

ii) FileInputOutput

Derivation: In our TalkBox configuration app, it displays all wav files found in a certain directory. Using this, the user can import all the audio files they want from a specified directory. These tests were derived due to errors when starting the configurator and the path was incorrect.

Significance: Successfully passing all FileInputOutput tests signifies that the TalkBox device will be able to start properly and find all audio files

iii) AudioSampleList

Derivation: The TalkBox configurator uses profiles to store the users desired audio files. Using the AudioSampleList class, the user can visually see all their audio files in the specified directory.

Significance: Successfully passing all the AudioSampleList test signifies the TalkBox is able to display all wav files and users should be able to create buttons and profiles.

iv) Serializer/TalkBoxConfig

Derivation: The simulator works by taking in a TalkBoxData.tbc, which is created by the TalkBox configuration app, and deserializes it. It then takes all the information and forms the simulator app. These tests were derived due to the dependency of the simulator app. The correct information needs to be serialized in order to create the simulator.

Significance: Successfully passing the given Serialization tests signifies the correct information was serialized and deserialized, allowing the simulator app to be made.

v) Sound

Derivation: The TalkBox configurator needs allow users to successfully record their own audio. Using a recording class, it is able to record audio and save it as any filename the user wants. These tests were derived in order to test if a wav file is successfully created and stored.

Significance: Successfully passing all the Sound tests signifies the TalkBox can successfully record audio.

GUI Testing

3.1 Test Method

For testing the GUI, manual testing was used to test the configurator and simulator.

3.2 Test Cases Configurator

Test Class/ Item Tested	Test Case Description
Profilelist/ Profile TextField & TreeView	When entering a profile, the profile is added to the TreeView
ButtonPanel, Profilelist & AudioSampleList/ ListView, TreeView, GridPane	After creating a profile, the GUI adds the clicked audio file to the desired profile as well as creates the button on the preview grid pane
ImportFiles AudioSampleList/ ListView, MenuItem	When clicking the import audio button, the user's home directory appears and allows the user to navigate between their personal directories and pick which wav file they would like to add. The audio file should then appear in the list view of available audio files.
ProfileList/ TreeView	The TalkBox preview displays the profile which the user clicks
AudioClip/ Audio Buttons, GridPane	Buttons successfully plays desired audio
ContextMenuClass, ButtonPanel & TalkBoxButtons/ Right Click Menu, MenuItem, TextField	A context menu with the options "Load Image" and "Rename" appears when the user right clicks. The user is able to use the renaming TextField to rename the desired button.

ContextMenuClass, ButtonPanel/ Right Click Menu, MenuItem	A context menu with the options "Load Image" and "Rename" appears when the user right clicks. The user is able to navigate through their personal directories and import a image onto the button. The image resizes to fit the button.
GuiConfig/GUI Launch	Simulator successfully launches from configurator.

3.2.1 Test Case Derivation and Significance

i) Profile TextField & TreeView

Derivation: TalkBox must allow users to create profiles to hold their desired audio files

Significance: Successfully creating the profiles signifies the TreeView and the Profile TextField is functional and the user is able add audio files.

ii) ListView, MenuItem

Derivation: A feature that was implanted was to allow users to import their own wav files from their personal computers and import them into the TalkBox. The audio will then appear at the bottom of the ListView.

Significance: Successfully importing audio signifies users will not only be allowed to record but also import any wav file they may already have.

iii) ListView, TreeView, GridPane

Derivation: Users should be able to add audio files to their desired profiles and the TalkBoxPreview just by clicking the audio file from the ListView.

Significance: Successfully adding an audio files signifies the TreeView, ListView and GridPane are all able to work with each other as intended.

iv) TreeView

Derivation: The user should be able to switch between profiles and audio sets

Significance: Successfully displaying the clicked profiles signifies the TalkBox is able to switch between both profiles and audio sets.

v) Audio Buttons, Grid Pane

Derivation: TalkBox must be able to playback audio

Significance: Successfully playing sounds signifies the TalkBox is able to play audio.

vi) Right Click Menu, MenuItem, TextField

Derivation: The user should be able to rename the audio button to any desired name.

Significance: Successfully renaming the button signifies that the user will be able to change the name of the button while still being able to play the desired audio.

vii) Right Click Menu, MenuItem

Derivation: The user should be able to add images to the audio buttons

Significance: Successfully adding an image signifies the user will be able to import any image and have it resize and fit to the desired button while still being able to play the desired audio.

viii) Launch Button

Derivation: The user can launch the simulator from the configurator with their desired settings

Significance: Successfully launching the simulator signifies the functionality of the configurator and the simulator works as intended.

3.3 Test Cases Configurator

Test Class/ Item Tested	Test Case Description
TalkBoxSim/ All GUI components	Simulator starts successfully
TalkBoxSimMenu/ Menu, MenuItem	Simulator displays the correct buttons, naming and imaging when clicking on a profile
TalkBoxSimProfiles/ Audio Buttons	The Simulator plays back the correct audio button associated with the profile and button.

3.3.1 Test Case Derivation and Significance

i) All GUI components

Derivation: TalkBoxSim has to start properly with the latest TalkBox

Significance: Successfully playing sounds signifies the TalkBox is able to play audio.

ii) Right Click Menu, MenuItem, TextField

Derivation: The Simulator should allow users to switch between profiles and audio sets

Significance: Successfully switching between profiles using the MenuItem signifies that the simulator is able to swap audio sets.

iii) Audio Buttons

Derivation: The Simulator should simulate a real TalkBox and play back audio when a button is pressed.

Significance: Successfully playing back audio signifies a working simulator.

3.4 Test Cases Logger

Test Class/ Item Tested	Test Case Description
TalkBoxLogger/ TextField, TreeView, ListView, Button, MenuItem	When clicking a GUI component in the configurator or simulator, the Logger should record the action save to a txt file
TalkBoxLog/ All GUI Components	The TalkBoxLog GUI should display the desired text file through the TextField

3.4.1 Derivation and Significance

i) TextField, TreeView, ListView, Button, MenuItem (TalkBoxConfig & Sim)

Derivation: The logger should be able to record all actions performed while the Configurator or Simulator is in use

Significance: Successfully recording each action signifies the logger is able to log all actions and save to a text file where an individual can look through at any time.

ii) All GUI Components (TBCLog)

Derivation: The GUI should be able to read the text file created by the logger and display it

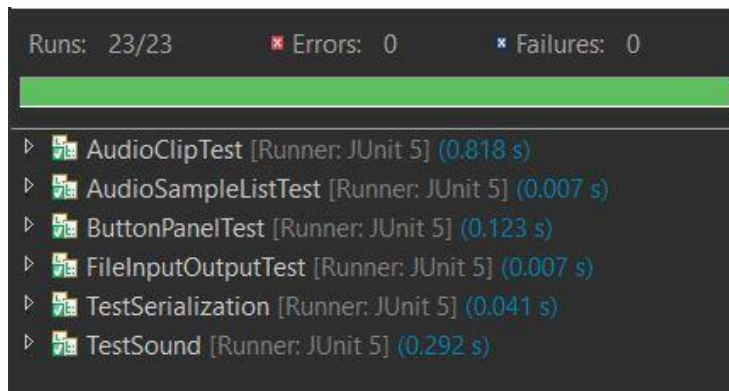
Significance: Successfully being able to display the text signifies a working TBCLog GUI and allows users to look through their past actions.

4. Test Coverage

4.1 Coverage Method

To calculate test coverage, EcEmma was used.

4.2 JUnit Test Results



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
TalkBox2.0	21.8 %	750	2,687	3,437
TalkBox/src	21.8 %	750	2,687	3,437
_main	4.6 %	83	1,713	1,796
TalkBoxSim	0.0 %	0	478	478
io	31.6 %	114	247	361
talkBoxLog	0.0 %	0	179	179
audio_players	77.6 %	177	51	228
Tester	95.2 %	376	19	395
FileInputOutputTest.java	86.7 %	65	10	75
TestSound.java	93.5 %	87	6	93
AudioClipTest.java	88.5 %	23	3	26
AudioSampleListTest.java	100.0 %	45	0	45
ButtonPanelTest.java	100.0 %	30	0	30
TestSerialization.java	100.0 %	126	0	126

4.2.1 Discussion

The JUnit tests results show that all tests pass, meaning the TalkBox functionalities should all work as intended. The coverage calculated by EcEmma, is seen to be at only 21.8%. This is because the GUI code is an executed through the test cases. The test cases do not test the GUI and the event fired, rather, it tests the functionality each the given classes. All the functionality classes are stored in TalkBoxConfig, which explains why the coverage for the TalkBoxSim is at 0%. The testing coverage

was recorded to be 95.2% with the last 4.8% not being used during testing due to the use of several try and catch.

4.3 Manual Testing Results

Fig. Configurator and Simulator Coverage

















Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▾ TalkBox2.0	 73.9 %	2,536	895	3,431
▾ TalkBox/src	 73.9 %	2,536	895	3,431
▸ Tester	 0.0 %	0	395	395
▸ talkBoxLog	 0.0 %	0	179	179
▸ _main	 90.3 %	1,622	174	1,796
▸ io	 82.5 %	298	63	361
▸ audio_players	 76.1 %	169	53	222
▸ TalkBoxSim	 93.5 %	447	31	478

Fig. Logger GUI Coverage

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▾ TalkBox2.0	 5.6 %	192	3,239	3,431
▾ TalkBox/src	 5.6 %	192	3,239	3,431
▸ _main	 0.0 %	0	1,796	1,796
▸ TalkBoxSim	 0.0 %	0	478	478
▸ Tester	 0.0 %	0	395	395
▸ io	 4.4 %	16	345	361
▸ audio_players	 0.0 %	0	222	222
▸ talkBoxLog	 98.3 %	176	3	179

4.3.1 Discussion

The manual testing done resulted in desired outcome which is that the both the TalkBox configuration app and the TalkBox simulator works as intended. The coverage data shows a percentage of 73.9%, much higher than the JUnit testing. The testing package shows 0% as it the JUnit tests are never used in manual testing. The logger GUI is also never used in the configurator and simulator testing thus why is shows 0 in the first figure and only 5.6% overall in the second figure. When adding the JUnit coverage, the logger coverage and the manual coverage, the total coverage comes up to 100%. The true coverage is less since both the JUnit coverage and the manual coverage both use some of the same classes. Nevertheless, when both the coverages are added, it shows that almost all the code is used when running the apps.