# TalkBox Software Requirements Specification

Contents

[TalkBox Software Requirements Specification 1](#_Toc5119732)

[Purpose and Product Scope 2](#_Toc5119733)

[User Classes and Characteristics 2](#_Toc5119734)

[Terminology 3](#_Toc5119735)

[Simulator Use Cases 3](#_Toc5119736)

[Use Case 1 – Load Configuration 3](#_Toc5119737)

[Use Case 2 – Play Button Audio 4](#_Toc5119738)

[Use Case 3 – Switch Profiles (Audio Sets) 5](#_Toc5119739)

[Configurer Use Cases 6](#_Toc5119740)

[Use Case 1 – Load Configuration 6](#_Toc5119741)

[Use Case 2 – Play Button Audio 7](#_Toc5119742)

[Use Case 3 – Switch Profiles (Audio Sets) 8](#_Toc5119743)

[Use Case 4 – Delete Profile (Audio Set) 9](#_Toc5119744)

[Use Case 5 – Create Profile (Audio Set) 10](#_Toc5119745)

[Use Case 6 – Record Button Audio 11](#_Toc5119746)

[Use Case 7 – Update Number of Audio Buttons 12](#_Toc5119747)

[Use Case 8 – Rename Audio Button 12](#_Toc5119748)

[Use Case 9 – Add Image Icon to Audio Button 13](#_Toc5119749)

[Use Case 10 – Add Audio File to Audio Button 14](#_Toc5119750)

[Use Case 11 – Save TalkBox Configuration Settings 15](#_Toc5119751)

[Use Case 12 – Launch Simulator from Configurer 16](#_Toc5119752)

[Use Case 13 – Load and View Simulator Log Files 17](#_Toc5119753)

[TBCLog Use Cases 18](#_Toc5119754)

[Use Case 1 – Load and View Log Files 18](#_Toc5119755)

[Use Case 2 – Load the Previous or Next Log 19](#_Toc5119756)

[Use Case 3 – Search Log Events in Log Files 19](#_Toc5119757)

[Acceptance Tests 20](#_Toc5119758)

[Simulator Acceptance Tests 20](#_Toc5119759)

[Configurer Acceptance Tests 23](#_Toc5119760)

[TBCLog Acceptance Tests 27](#_Toc5119761)

[Operating Environment 28](#_Toc5119762)

[Design and Implementation Constraints 28](#_Toc5119763)

## Purpose and Product Scope

This document specifies the requirements of the TalkBox Software System (henceforth TalkBox). TalkBox is a graphical user interface delivered in two subsystems. The first subsystem, the Simulator, provides users with an easy way to test the configuration and practice the usage of a TalkBox hardware device. The second subsystem, the Configurer, provides users with a simple interface for recording audio and organizing, loading, and saving settings for use with the Simulator or a TalkBox hardware device.

The TalkBox hardware device intends to provide a cost-effective way for speech-impaired individuals to communicate more effectively. TalkBox must provide an easy way for friends and family of a speech-impaired individual to customize the behaviour of a TalkBox hardware device. The Configurer and Simulator must accommodate TalkBox hardware devices of various shapes and sizes and a variable number of buttons.

## User Classes and Characteristics

The primary users of TalkBox are family and caretakers of speech-impaired individuals. These primary users are assumed minimally familiar with other software systems. They will use the system frequently and will use all product functions to ensure they provide their family or their charge with the best care.

Secondary users may include friends or guests of the speech-impaired individual. These users will use the software occasionally and will need to quickly become familiar with its usage. Secondary users will use a core subset of the functions of the system.

## Terminology

Preconditions are assertions about the system that must be true before the use case begins while postconditions are assertions that must be true about the system after the use case has completed.

Basic flow refers to the normal course of events that leads to the success of the use case. Alternate flows refer to variations from the basic flow that still lead to the success of the use case. Exception flows are exceptional cases that usually indicate an error has occurred or a necessary condition for success has not been met. Exception flows typically do not lead to the success of the use case but should be handled gracefully such that the application is able to revert to a good state and continue functioning.

The terms play mode and edit mode refer to modes of the Configurer app. These modes offer different but partially overlapping feature sets. The intention is that play mode is when the user wants to use the preview of the Simulator built into the Configurer to play back audio. Edit mode on the other hand allows the user to select audio buttons for editing and to record audio to the button.

## Simulator Use Cases

### Use Case 1 – Load Configuration

|  |  |
| --- | --- |
| **Name** | **Load Configuration** |
| Description | As a primary or secondary user, I want to load configuration settings for the TalkBox Simulator or hardware device so that I may test and confirm my preferred settings and saved audio files are saved and working as I want. This use case begins when a user launches the Simulator by itself and ends when the Simulator finishes displaying the loaded configuration. |
| Actors | * Simulator * Primary Users * Secondary Users |
| Preconditions | The Simulator app is not open. |
| Basic Flow | 1. The user launches the Simulator using the provided TalkBoxSim.jar file 2. The Simulator opens a file chooser dialog box 3. The user selects a valid TalkBox configuration file and clicks open in the file chooser dialog 4. The Simulator loads the selected configuration and its first profile, even if it contains no audio files |
| Exception Flow 1 | 1. The user selects an invalid TalkBox configuration file and clicks open in the file chooser dialog 2. The Simulator informs the user that the profile failed to load 3. The Simulator exits |
| Postconditions | If the TalkBox configuration file selected was valid, the Simulator app is open and idling in a good state. The first profile is loaded, and any available audio files are mapped to the correct button. If the TalkBox configuration file selected was invalid, the Simulator does not launch. |

### Use Case 2 – Play Button Audio

|  |  |
| --- | --- |
| **Name** | **Play Button Audio** |
| Description | As a primary or secondary user, I want to playback audio to test my saved audio files and their audio button associations. This use case begins when a user clicks an audio button in the Simulator and ends when the Simulator finishes playing back the correct audio file. The audio file should be the correct file associated with the clicked audio button for the currently loaded profile. |
| Actors | * Simulator * Primary Users * Secondary Users |
| Preconditions | The Simulator app is open and idling in a good state. An audio set profile is currently loaded. |
| Basic Flow | 1. The user clicks an audio button in the Simulator 2. The Simulator begins playing back the correct audio file 3. The Simulator finishes playing back the correct audio file |
| Alternate Flows | 1. The user clicks another audio button before the Simulator finishes playing back the current audio file 2. The Simulator stops playback of the current audio file and reinitiates this use case from basic flow step 1 |
| Exception Flow 1 | 1. The Simulator cannot play back the audio file as it is invalid or missing 2. The Simulator displays a helpful error message describing the button, it's associated audio file path, and why the audio file could not be played |
| Postconditions | The Simulator app is open and idling in a good state. No audio is playing. The profile loaded before this use case is still loaded. |

### Use Case 3 – Switch Profiles (Audio Sets)

|  |  |
| --- | --- |
| **Name** | **Switch Profiles** |
| Description | As a primary or secondary user, I want to switch profiles to test my saved audio files and their audio button associations across all of my saved profiles. This use case begins when a user clicks one of the fixed profile buttons labeled profile 1, profile 2, or profile 3 or when a user clicks the swap button. This use case ends when the Simulator finishes loading the correct audio set. The swap button cycles through all available profiles linearly starting from the currently loaded profile. |
| Actors | * Simulator * Primary Users * Secondary Users |
| Preconditions | The Simulator app is open and idling in a good state. An audio set profile is currently loaded. |
| Basic Flow | 1. The user clicks one of the following buttons: profile 1, profile 2, profile 3, or swap 2. The Simulator loads the correct profile as indicated by the button or loads the next available profile, cycling back to the first if the last profile is currently loaded |
| Alternate Flows | None |
| Exception Flow 1 | 1. If the profile to be loaded does not exist, then the Simulator does nothing else and the use case ends |
| Postconditions | The Simulator app is open and idling in a good state. Either a new profile is loaded according to the button clicked or the profile loaded before this use case is still loaded. |

## Configurer Use Cases

### Use Case 1 – Load Configuration

|  |  |
| --- | --- |
| **Name** | **Load Configuration** |
| Description | As a primary or secondary user, I want to load configuration settings for the TalkBox so that I may modify settings or record new button audio. This use case begins when a user launches the Configurer by itself and ends when the Configurer finishes displaying the loaded configuration. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is not open. |
| Basic Flow | 1. The user launches the Configurer using the provided TalkBoxConfig.jar file 2. The Configurer opens a file chooser dialog box 3. The user selects a valid TalkBox configuration file and clicks open in the file chooser dialog 4. The Configurer loads the selected configuration and its first profile, even if it contains no audio files |
| Alternate Flow 1 | 1. The user selects a directory which does not contain a configuration file named precisely TalkBoxData.tbc 2. The Configurer creates a TalkBoxData directory in the chosen directory 3. The Configurer creates a TalkBoxData.tbc configuration file in the TalkBoxData directory created in step 4 4. The Configurer loads the generated TalkBoxData.tbc configuration file and its first profile |
| Exception Flow 1 | 1. The user selects an invalid TalkBox configuration file and clicks open in the file chooser dialog 2. The Configurer informs the user that the profile failed to load 3. The Configurer exits |
| Postconditions | If the TalkBox configuration file selected was valid, the Configurer app is open and idling in a good state. The first profile is loaded, and any available audio files are mapped to the correct button. If the TalkBox configuration file selected was invalid, the Configurer did not launch. |

### Use Case 2 – Play Button Audio

|  |  |
| --- | --- |
| **Name** | **Play Button Audio** |
| Description | As a primary or secondary user, I want to playback audio to test my saved audio files and their audio button associations. This use case begins when a user clicks an audio button in the Configurer and ends when the Configurer finishes playing back the correct audio file. The audio file should be the correct file associated with the clicked audio button for the currently loaded profile. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. A profile is currently loaded. The Configurer is in play mode. |
| Basic Flow | 1. The user clicks an audio button in the Configurer 2. The Configurer begins playing back the correct audio file 3. The Configurer finishes playing back the correct audio file |
| Alternate Flow 1 | 1. The user clicks another audio button before the Configurer finishes playing back the current audio file 2. The Configurer stops playback of the current audio file and reinitiates this use case from basic flow step 1 |
| Exception Flow 1 | 1. The Configurer cannot play back the audio file as it is invalid or missing 2. The Configurer displays a helpful error message describing the button, it's associated audio file path, and why the audio file could not be played |
| Postconditions | The Configurer app is open and idling in a good state. No audio is playing. The profile loaded before this use case is still loaded. The Configurer is in play mode. |

### Use Case 3 – Switch Profiles (Audio Sets)

|  |  |
| --- | --- |
| **Name** | **Play Button Audio** |
| Description | As a primary or secondary user, I want to switch profiles to test my saved audio files and their audio button associations across all my saved profiles. This use case begins when a user clicks one of the fixed profile buttons labeled profile 1, profile 2, or profile 3 or when a user clicks the swap button. This use case may also begin when the user selects a profile from the profiles list and clicks load profile. This use case ends when the Configurer finishes loading the correct audio set. The swap button cycles through all available profiles linearly starting from the currently loaded profile. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. A profile is currently loaded. The Configurer is in play mode. |
| Basic Flow | 1. The user clicks one of the following buttons: profile 1, profile 2, profile 3, swap, or load profile 2. The Configurer loads the correct profile as indicated by the button label or the currently highlighted profile in the profiles list. If the swap button is used the Configurer loads the next available profile, cycling back to the first if the last profile is currently loaded |
| Alternate Flows | None |
| Exception Flow 1 | 1. If the profile to be loaded does not exist, then the Configurer does nothing else and the use case ends |
| Postconditions | The Configurer app is open and idling in a good state. Either a new profile is loaded according to the button clicked or the profile loaded before this use case is still loaded. The Configurer is in play mode. |

### Use Case 4 – Delete Profile (Audio Set)

|  |  |
| --- | --- |
| **Name** | **Delete Profile (Audio Set)** |
| Description | As a primary or secondary user, I want to delete profiles that I no longer need so I can focus on the ones that do. This use case begins when a user clicks the delete profile button in the Configurer and ends when the Configurer removes the currently selected profile in the profiles list. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. A profile is currently loaded. |
| Basic Flow | 1. The user clicks the delete profile button 2. The Configurer warns the user that the delete operation is permanent and asks if they are sure they want to proceed 3. The user clicks ok to proceed 4. The Configurer deletes the currently selected profile from disk 5. The Configurer removes the currently selected profile from the interface 6. The Configurer selects the next profile above the deleted profile in the profiles list |
| Alternate Flow 1 | 1. The user clicks cancel to stop this use case from proceeding 2. The Configurer closes the warning dialog and returns to its state before the use case was initiated |
| Exception Flow 1 | 1. The Configurer cannot find the currently selected profile on disk 2. Resume from basic flow step 5 |
| Exception Flow 2 | 1. The Configurer cannot delete the profile from disk 2. The Configurer informs the user that the profile could not be deleted and that they should check if the profile is open in another program |
| Postconditions | The Configurer app is open and idling in a good state. The profile loaded before this use case is still loaded and displayed. |

### Use Case 5 – Create Profile (Audio Set)

|  |  |
| --- | --- |
| **Name** | **Create Profile (Audio Set)** |
| Description | As a primary or secondary user, I want to create profiles so I can organize sets of audio files and their button associations and so my charge can easily switch between sets of audio. This use case begins when a user clicks the create profile button in the Configurer and ends when the Configurer finishes displaying the new profile in the profile menu. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. A profile is currently loaded. |
| Basic Flow | 1. The user clicks the create profile button 2. The Configurer creates a new, automatically named, profile that is saved to disk as a folder of the same name 3. The Configurer adds the profile name to the end of the profiles list |
| Alternate Flow 1 | 1. The Configurer finds a folder with the same name already on disk uses it to store the newly created profile's data 2. Resume from basic flow step 3 |
| Exception Flow 1 | 1. The Configurer cannot create a new folder on disk to hold the profile because of a write access issue 2. The Configurer displays a user-friendly error message and does not create a new profile |
| Postconditions | The Configurer app is open and idling in a good state. A profile is currently loaded. If the basic flow was successfully completed, then one new profile has been added. Otherwise the profiles list is unchanged. |

### Use Case 6 – Record Button Audio

|  |  |
| --- | --- |
| **Name** | **Record Button Audio** |
| Description | As a primary or secondary user, I want to record new audio files and associate them with buttons so my charge can play them back using the TalkBox hardware device to communicate. This use case begins when a user clicks an audio button while in edit mode and ends when the user clicks the microphone button to end the recording. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. The Configurer is in edit mode. |
| Basic Flow | 1. The user clicks an audio button to select it for editing 2. The user clicks the microphone button to begin recording 3. The Configurer begins recoding audio from a connected microphone device and changes the icon and label of the microphone button to indicate it is recording 4. The user clicks the microphone button to end recording 5. The Configurer associates the recorded audio file to the button selected for editing |
| Alternate Flows | None |
| Exception Flow 1 | 1. The Configurer is unable to find or access a microphone device 2. The Configurer changes the icon and label of the microphone button to indicate that a recording device is unavailable 3. The Configurer displays a message below the microphone button asking the user to connect a recording device and try clicking the microphone button again |
| Postconditions | The Configurer app is open and idling in a good state. The Configurer is in edit mode. The audio button selected during this use case has the newly recorded audio associated with it. |

### Use Case 7 – Update Number of Audio Buttons

|  |  |
| --- | --- |
| **Name** | **Update Number of Audio Buttons** |
| Description | As a primary or secondary user, I want to set the number of audio buttons to match my TalkBox hardware device so I can configure my TalkBox settings in the right context. This use case begins when a user enters a number into the update number of buttons text field and ends when the user hits enter or clicks the update number of buttons button. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. |
| Basic Flow | 1. The user clicks the update number of buttons text field 2. The user types a positive value 3. The Configurer sets the number of buttons to that entered by the user |
| Alternate Flow 1 | 1. The user types non-positive value or non-numeric value 2. The Configurer does not alter the number of buttons and informs the user that a positive numeric value is required |
| Exception Flows | None |
| Postconditions | The Configurer app is open and idling in a good state. The number of buttons is the same as the number of buttons set during the use case. |

### Use Case 8 – Rename Audio Button

|  |  |
| --- | --- |
| **Name** | **Rename Audio Button** |
| Description | As a primary or secondary user, I want to set the button names so that I can remember what audio file is associated with a button at a glance and so I can label my TalkBox hardware device appropriately. This use case begins when a user enters a string into the update button label text field and ends when the user hits enter or clicks the update button label button. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. The Configurer is in edit mode and a button is selected for editing |
| Basic Flow | 1. The user clicks the update button label text field 2. The user types a textual string value 3. The Configurer sets the label of the currently selected button to the entered string value |
| Alternate Flows | None |
| Exception Flows | None |
| Postconditions | The Configurer app is open and idling in a good state. The button label of the selected audio button is updated. |

### Use Case 9 – Add Image Icon to Audio Button

|  |  |
| --- | --- |
| **Name** | **Add Image Icon to Audio Button** |
| Description | As a primary or secondary user, I want to set an image icon for each audio button so I can remember what audio file is associated with a button at a glance and so I can label my TalkBox hardware device appropriately. This use case begins when a user clicks the Upload Image button or drags and drops an image file onto an audio button. This use case ends when the Configurer updates the audio button with the chosen image icon. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. The Configurer is in edit mode and a button is selected for editing. |
| Basic Flow | 1. The user clicks the Upload Image button 2. The Configurer opens a file chooser dialog 3. The user selects a valid image file from disk and clicks the Open button 4. The Configurer sets the icon of the currently selected audio button to the uploaded image file |
| Alternate Flow 1 | 1. The user clicks Cancel to stop this use case from proceeding 2. The Configurer closes the file chooser dialog and returns to its state before the use case was initiated |
| Exception Flow 1 | 1. The Configurer is unable to read the image file on disk 2. The Configurer warns the user that the file could not be read and they should check whether the file exists and is not open in another program before trying again. |
| Exception Flow 2 | 1. The user selects an invalid image file and clicks the Open button 2. The Configurer warns the user that the file was not a valid image file that can be assigned as an icon to an audio button |
| Postconditions | The Configurer app is open and idling in a good state. The image icon of the selected audio button is the same as before the use case or is updated to the newly selected, valid image file. |

### Use Case 10 – Add Audio File to Audio Button

|  |  |
| --- | --- |
| **Name** | **Add Audio File to Audio Button** |
| Description | As a primary or secondary user, I want to set an audio file for an audio button so I can reuse audio files I have recorded or prepared outside of the TalkBox system. This use case begins when a user clicks the Upload Audio button or drags and drops an audio file onto an audio button. This use case ends when the Configurer updates the audio button with the chosen audio file. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. The Configurer is in edit mode and a button is selected for editing. |
| Basic Flow | 1. The user clicks the Upload Audio button 2. The Configurer opens a file chooser dialog 3. The user selects a valid audio file from disk and clicks the Open button 4. The Configurer sets the audio of the currently selected audio button to the uploaded audio file |
| Alternate Flow 1 | 1. The user clicks Cancel to stop this use case from proceeding 2. The Configurer closes the file chooser dialog and returns to its state before the use case was initiated |
| Exception Flow 1 | 1. The Configurer is unable to read the audio file on disk 2. The Configurer warns the user that the file could not be read, and they should check whether the file exists and is not open in another program before trying again. |
| Exception Flow 2 | 1. The user selects an invalid audio file and clicks the Open button 2. The Configurer warns the user that the file was not a valid audio file that can be assigned as an icon to an audio button |
| Postconditions | The Configurer app is open and idling in a good state. The audio of the selected audio button is the same as before the use case or is updated to the newly selected, valid audio file. |

### Use Case 11 – Save TalkBox Configuration Settings

|  |  |
| --- | --- |
| **Name** | **Save Settings** |
| Description | As a primary or secondary user, I want to save my configuration settings so I can load them into the TalkBox simulator or hardware device for testing or for use by my charge. As a primary or secondary user, I want to set the number of audio buttons to match my TalkBox hardware device so I can configure my TalkBox settings in the right context. This use case begins when a user clicks the save settings button and finishes when the Configurer finishes writing the current settings to disk. |
| Actors | * Configurer * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. |
| Basic Flow | 1. The user clicks the save settings button 2. The Configurer writes the current configuration settings to disk in the TalkBoxData setup at launch in Use Case 1, overwriting existing settings |
| Alternate Flows | None |
| Exception Flow 1 | 1. The Configurer is unable to write to disk 2. The Configurer warns the user that the current settings were not saved |
| Postconditions | The Configurer app is open and idling in a good state. The configuration settings saved on disk match the settings displayed by the Configurer. |

### Use Case 12 – Launch Simulator from Configurer

|  |  |
| --- | --- |
| **Name** | **Launch Simulator** |
| Description | As a primary or secondary user, I want to rapidly test my current configuration in the Simulator without having to launch it as a separate application. This use case begins when a user clicks the launch simulator button and ends when the Simulator app finishes launching. |
| Actors | * Configurer * Simulator * Primary Users * Secondary Users |
| Preconditions | The Configurer app is open and idling in a good state. |
| Basic Flow | 1. The user clicks the launch simulator button 2. The Configurer tells the user that settings will be saved and overwritten before launching the simulator and if they would like to proceed 3. The user clicks yes 4. The Configurer saves and overwrites settings 5. The Configurer launches the simulator providing it the path to the TalkBoxData directory 6. The Simulator loads the configuration settings from disk |
| Alternate Flow 1 | 1. The user clicks no 2. The Configurer does not save settings or launch the simulator 3. The Configurer returns to idling as before the use case |
| Exception Flow 1 | 1. The Configurer is unable to write to disk 2. The Configurer warns the user that the current settings could not be saved, and the Simulator could not be launched 3. The Configurer returns to idling as before the use case |
| Postconditions | The Configurer app is open and idling in a good state. The Simulator is open and idling in a good state. The configuration settings saved on disk match the settings displayed by the Configurer and by the Simulator. |

### Use Case 13 – Load and View Simulator Log Files

|  |  |
| --- | --- |
| **Name** | **Load and View Simulator Log Files** |
| Description | As a primary or secondary user, I want to load and view log files produced by the Simulator subsystem so I can analyze how it is used and can improve my configuration of the Simulator. This use case begins when a user clicks the Load Log button and ends when the Configurer app displays the log. |
| Actors | * Configurer * Primary Users * Tertiary Users |
| Preconditions | The Configurer app is open and idling in a good state with the log text area displayed. |
| Basic Flow | 1. The user clicks the Load Log button 2. The Configurer opens a file chooser dialog 3. The user selects a valid log file produced by the Simulator and clicks the Open button 4. The Configurer reads in the selected file and replaces its main text area contents with that of the file 5. The Configurer resets the Search text field |
| Alternate Flow 1 | 1. The user clicks the Cancel button 2. The Configurer closes the file chooser dialog and does not update its main text area |
| Exception Flow 1 | 1. The Configurer is unable to read the file on disk 2. The Configurer warns the user that the file could not be read, and they should check whether the file exists and is not open in another program before trying again. |
| Exception Flow 2 | 1. The log file chosen is not a valid text file that can be read into the main text area 2. The Configurer warns the user the file is not a valid log file and does not update its main text area |
| Postconditions | The Configurer is open and idling in a good state. The previously loaded log file or the newly selected log file is displayed in the main text area. |

## TBCLog Use Cases

### Use Case 1 – Load and View Log Files

|  |  |
| --- | --- |
| **Name** | **Load and View Log Files** |
| Description | As a tertiary user, I want to load and view log files produced by the Configurer or Simulator so I can analyze how the primary and secondary users are using them. This will help me improve the functionality of the app. This use case begins when a user clicks the Load Log button and ends when the TBCLog app displays the log. |
| Actors | * TBCLog app * Tertiary Users |
| Preconditions | The TBCLog app is open and idling in a good state. |
| Basic Flow | 1. The user clicks the Load Log button 2. The TBCLog app opens a file chooser dialog 3. The user selects a valid log file produced by the Configurer or the Simulator and clicks the Open button 4. The TBCLog app reads in the selected file and replaces its main text area contents with that of the file 5. The TBCLog app resets the Search text field |
| Alternate Flow 1 | 1. The user clicks the Cancel button 2. The TBCLog app closes the file chooser dialog and does not update its main text area |
| Exception Flow 1 | 1. The TBCLog app is unable to read the file on disk 2. The TBCLog app warns the user that the file could not be read and they should check whether the file exists and is not open in another program before trying again. |
| Exception Flow 2 | 1. The log file chosen is not a valid text file that can be read into the main text area 2. The TBCLog app warns the user the file is not a valid log file and does not update its main text area |
| Postconditions | The TBCLog app is open and idling in a good state. The previously loaded log file or the newly selected log file is displayed in the main text area. |

### Use Case 2 – Load the Previous or Next Log

|  |  |
| --- | --- |
| **Name** | **Load the Previous or Next Log** |
| Description | As a tertiary user, I want to quickly load the previous or next chronological log so I can analyze how user's use the Configurer or Simulator over time. This use case begins when a user clicks the Previous Log button or Next Log button and ends when the Log app displays the correct log. |
| Actors | * TBCLog * Tertiary Users |
| Preconditions | The Log app is open and idling in a good state. |
| Basic Flow | 1. The user clicks the Previous Log button or Next Log button 2. The TBCLog app reads in the previous or the next chronological log file available 3. The TBCLog app resets the Search text field |
| Alternate Flow 1 | 1. There is no previous or next log available 2. The TBCLog app informs the user there is no previous or next chronological log file available 3. The TBCLog app returns to idling as before the use case |
| Exception Flow 1 | 1. The TBCLog app is unable to read the file on disk 2. The TBCLog app warns the user that the file could not be read and that they should check whether the file exists and is not open in another program |
| Postconditions | The TBCLog app is open and idling in a good state. The previously loaded log file (before this use case) or the chronologically previous or next log file is displayed in the main text area. |

### Use Case 3 – Search Log Events in Log Files

|  |  |
| --- | --- |
| **Name** | **Search Log Events in Log Files** |
| Description | As a tertiary user, I want to filter log events so I can quickly get to the information I want and understand how a feature is being used. This use case begins when a user types in the Search text field and ends when the TBCLog app finishes displaying the filtered log file in the main text area and the number of matching events in the Matching Events label. |
| Actors | * TBCLog * Tertiary Users |
| Preconditions | The TBCLog app is open and idling in a good state. |
| Basic Flow | 1. The user types a character into the Search text field 2. The TBCLog app filters the contents of the main text area 3. The TBCLog app updates the Matching Events label to indicate the number of log events matching the search term |
| Alternate Flow 1 | None |
| Exception Flow 1 | None |
| Postconditions | The Configurer app is open and idling in a good state. |

## Acceptance Tests

Each use case will be tested separately. An acceptance test either passes or fails, there is no partial success. Acceptance tests will be derived from the basic, alternate, and exception flows from use cases and will be checked for success by establishing preconditions and then checking for the postconditions to be satisfied. The acceptance tests in this section provide additional checks on the behaviour of the program not fully captured by the use cases.

### Simulator Acceptance Tests

**RULE 1: Only launch the Simulator if a TalkBoxData folder containing a valid TalkBoxData.tbc file is selected**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | TalkBoxSim.jar was launched and the file chooser dialog is open | User selects a TakBoxData folder containing a valid TalkBoxData.tbc file | Simulator loads and displays an interface that matches the selected settings |
| 02 | TalkBoxSim.jar was launched and the file chooser dialog is open | User selects a TalkBoxData folder that does not contain a TalkBoxData.tbc file | Simulator informs the user that a TalkBoxData.tbc file was not found and exits |
| 03 | TalkBoxSim.jar was launched and the file chooser dialog is open | User cancels the file chooser dialog | Simulator exits |
| 04 | TalkBoxSim.jar was launched and the file chooser dialog is open | User selects a TakBoxData folder containing an invalid TalkBoxData.tbc file | Simulator informs the user that an invalid TalkBoxData.tbc file was selected and exits |
| 05 | TalkBoxSim.jar was launched and the file chooser dialog is open | User selects a folder not named TalkBoxData | Simulator informs the user that a folder named TalkBoxData must be selected and exits |

**RULE 2: Any audio playback is stopped when an audio button is clicked and then if the button has a valid associated audio file it is played**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Audio is NOT currently playing | User clicks an audio button with an associated audio file | Simulator plays the associated audio file |
| 02 | Audio is currently playing | User clicks an audio button with an associated audio file | Simulator stops playback of previous audio, then begins playing the associated audio file |
| 03 | Audio is NOT currently playing | User clicks an audio button without an associated audio file | Simulator displays a message telling the user the button clicked is not associated with any audio file |
| 04 | Audio is currently playing | User clicks an audio button without an associated audio file | Simulator stops audio playback AND displays a message telling the user the button clicked is not associated with any audio file |

**RULE 3: Valid and available profiles are loaded when a fixed profile swap button is clicked**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Profile 1 is currently loaded AND Profile 2 is valid and available | User clicks the profile 2 button | Simulator loads Profile 2 |
| 02 | Profile 1 is currently loaded AND Profile 2 is invalid or unavailable | User clicks the profile 2 button | Simulator warns the user that Profile 2 is invalid or unavailable. Profile 1 remains loaded. |

**RULE 4: Valid and available profiles are loaded in numerical order when the profile swap button is clicked, cycling back to the first profile when the last profile is loaded**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Profile 2 is currently loaded AND Profile 1 is valid and available. No other profiles are both valid and available. | User clicks the swap button | Simulator loads Profile 1 |
| 02 | Profile 2 is currently loaded. No other profiles are both valid and available. | User clicks the swap button | Profile 2 remains loaded |

**RULE 5: Audio Button Labels, Icons, and Associated Audio Files**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Profile 2 is currently loaded AND Profile 1 is valid and available. No other profiles are both valid and available. | User clicks the swap button | Simulator loads Profile 1 |
| 02 | Profile 2 is currently loaded. No other profiles are both valid and available. | User clicks the swap button | Profile 2 remains loaded |

### Configurer Acceptance Tests

**RULE 1: Only launch the Simulator if a TalkBoxData folder containing a valid TalkBoxData.tbc file is selected**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | TalkBoxConfig.jar was launched and the file chooser dialog is open | User selects a TakBoxData folder containing a valid TalkBoxData.tbc file | Configurer loads and displays an interface that matches the selected settings |
| 02 | TalkBoxConfig.jar was launched and the file chooser dialog is open | User selects a TalkBoxData folder that does not contain a TalkBoxData.tbc file | Configurer informs the user that a TalkBoxData.tbc file was not found and exits |
| 03 | TalkBoxConfig.jar was launched and the file chooser dialog is open | User cancels the file chooser dialog | Configurer exits |
| 04 | TalkBoxConfig.jar was launched and the file chooser dialog is open | User selects a TakBoxData folder containing an invalid TalkBoxData.tbc file | Configurer informs the user that an invalid TalkBoxData.tbc file was selected and exits |
| 05 | TalkBoxConfig.jar was launched and the file chooser dialog is open | User selects a folder not named TalkBoxData | Configurer informs the user that a folder named TalkBoxData must be selected and exits |

**RULE 2: Any audio playback is stopped when an audio button is clicked and then if the button has a valid associated audio file it is played**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Audio is NOT currently playing | User clicks an audio button with an associated audio file | Configurer plays the associated audio file |
| 02 | Audio is currently playing | User clicks an audio button with an associated audio file | Configurer stops playback of previous audio, then begins playing the associated audio file |
| 03 | Audio is NOT currently playing | User clicks an audio button without an associated audio file | Configurer displays a message telling the user the button clicked is not associated with any audio file |
| 04 | Audio is currently playing | User clicks an audio button without an associated audio file | Configurer stops audio playback AND displays a message telling the user the button clicked is not associated with any audio file |

**RULE 3: Valid and available profiles are loaded when a fixed profile swap button is clicked**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Profile 1 is currently loaded AND Profile 2 is valid and available | User clicks the profile 2 button | Configurer loads Profile 2 |
| 02 | Profile 1 is currently loaded AND Profile 2 is invalid or unavailable | User clicks the profile 2 button | Configurer warns the user that Profile 2 is invalid or unavailable. Profile 1 remains loaded. |

**RULE 4: Valid and available profiles are loaded in numerical order when the profile swap button is clicked, cycling back to the first profile when the last profile is loaded**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Profile 2 is currently loaded AND Profile 1 is valid and available. No other profiles are both valid and available. | User clicks the swap button | Configurer loads Profile 1 |
| 02 | Profile 2 is currently loaded. No other profiles are both valid and available. | User clicks the swap button | Profile 2 remains loaded |

**RULE 5: While in edit mode exactly one audio button is always selected for editing**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Configurer is in play mode | User clicks switch modes | Configurer switches to edit mode AND selects either button 1 OR the last button selected while in edit mode for editing |
| 02 | Configurer is in edit mode | User clicks an audio button | Configurer deselects the current audio button AND selects the clicked audio button for editing |
| 03 | Configurer is in edit mode | User clicks switch modes | Configurer deselects the current audio button AND switches to play mode |
| 04 | Configurer is in edit mode and the *nth* button is selected | User updates the button number to less than *n* | The Configurer selects button 1 after the number of buttons has been updated |

**RULE 6: Only allow recording audio when in edit mode**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Configurer is in edit mode | User clicks microphone button | Configurer begins recording |
| 02 | Configurer is NOT in edit mode | User clicks microphone button | Nothing happens |

**RULE 7: Only allow updating number of buttons to a positive number**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Configurer is in edit mode | User clicks microphone button | Configurer begins recording |
| 02 | Configurer is NOT in edit mode | User clicks microphone button | Nothing happens |

**RULE 8: Only one simulator is ever kept open by the Configurer**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Configurer has launched a Simulator | User clicks the launch simulator button | Configurer informs user that only one Simulator may be launched at a time |

**RULE 9: If there is a launched Simulator, closing the Configurer also closes the Simulator**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Configurer has launched a Simulator | User closes the Configurer | The launched Simulator and the Configurer are both closed |

**RULE 10: Simulator launch forces settings to be saved to keep Configurer and Simulator in sync**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | Configurer has no launched Simulator | User clicks launch simulator AND THEN yes in the dialog box confirming that settings will be overwritten | Settings are saved overwriting existing settings on disk |

### TBCLog Acceptance Tests

**RULE 1: The main log text area and matching events text label update automatically when the search text field is updated**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | The TBCLog app has loaded and is displaying a valid log file | User enters or deletes a character from the search text field and there are matching events | The TBCLog app filters the log text area and updates the matching events text label |
| 02 | The TBCLog app has loaded and is displaying a valid log file | User enters or deletes a character from the search text field and there are NO matching events | The TBCLog app displays an empty text area and shows that there are NO matching events |

**RULE 2: The main log text area shows the name of the currently displayed log file**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **GIVEN** | **WHEN** | **THEN** |
| 01 | The TBCLog app has loaded and is idling | User loads a new log file | The TBCLog app updates the log text area header to the name of the newly loaded log file |

## Operating Environment

The TalkBox Software System uses Java and the Java Virtual Machine. The TalkBox Software System will run on any operating system that runs version 1.8 or higher of the Java Virtual Machine.

## Design and Implementation Constraints

The TalkBox hardware device will run Java software on the Java Virtual Machine. The hardware device will deserialize a Java object byte stream. The object byte stream will be stored in a file by the Configurer and transferred to the file system of the Raspberry Pi connected to the hardware device.