Software Design Document

for

TalkBox

Group 15

York University
EECS2311 Software Development Project

24. March 2019

Contents

Purpose and Product Scope

This document describes the design of the TalkBox Software System (henceforth TalkBox). TalkBox is a graphical user interface delivered in three 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 third subsystem, the TBCLog, displays log files generated by the Configurer for study primarily by the developers and researchers of the first two subsystems.

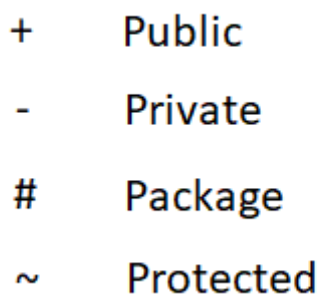
The TalkBox hardware device intends to provide a cost-effective way for speech-impaired individuals to communicate more effectively. TalkBox must provide an easy way for friends and family of a speech-impaired individual to customize the behavior of a TalkBox hardware device. The Configurer and Simulator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

Considering the product scope, this document details several class diagrams, sequence diagrams, as well as maintenance scenarios with the sequence diagrams. The class diagrams describe how the Java code of the TalkBox system is organized at a level higher than the code itself. The class diagrams include dependencies, aggregations and compositions along with their multiplicities. These features of the diagram show how different classes of the TalkBox system interact and interface with each other. The class diagrams are not a complete listing of the fields and methods contained in each class. Instead only the important fields and methods that relate to the core functionality of the classes are detailed. This makes the class diagrams easier to digest and provides a map of the design behind the TalkBox system. An automatically generated class diagram is used internally for keeping track of the full UML model of the TalkBox system.

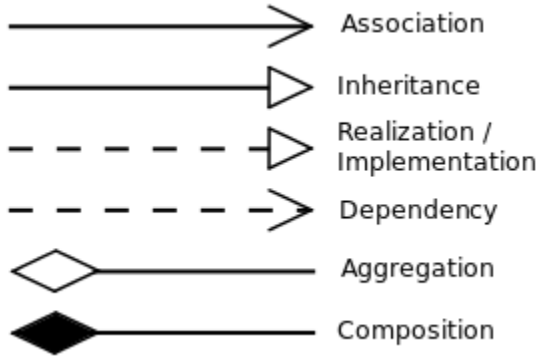
The sequence diagrams detail interactions between the users and the logical subsystems of the TalkBox software. Sequence diagrams show the logical flow necessary for use cases of the TalkBox system to succeed. Sequence diagrams also expose more detail about the order in which functions and methods are called and the communication required between segregated parts of the TalkBox system to complete required functionality. The sequence diagram section also talks about maintenance scenarios for each diagrams functionalities

UML Legend

The following describes the subset of UML used for the class diagrams:



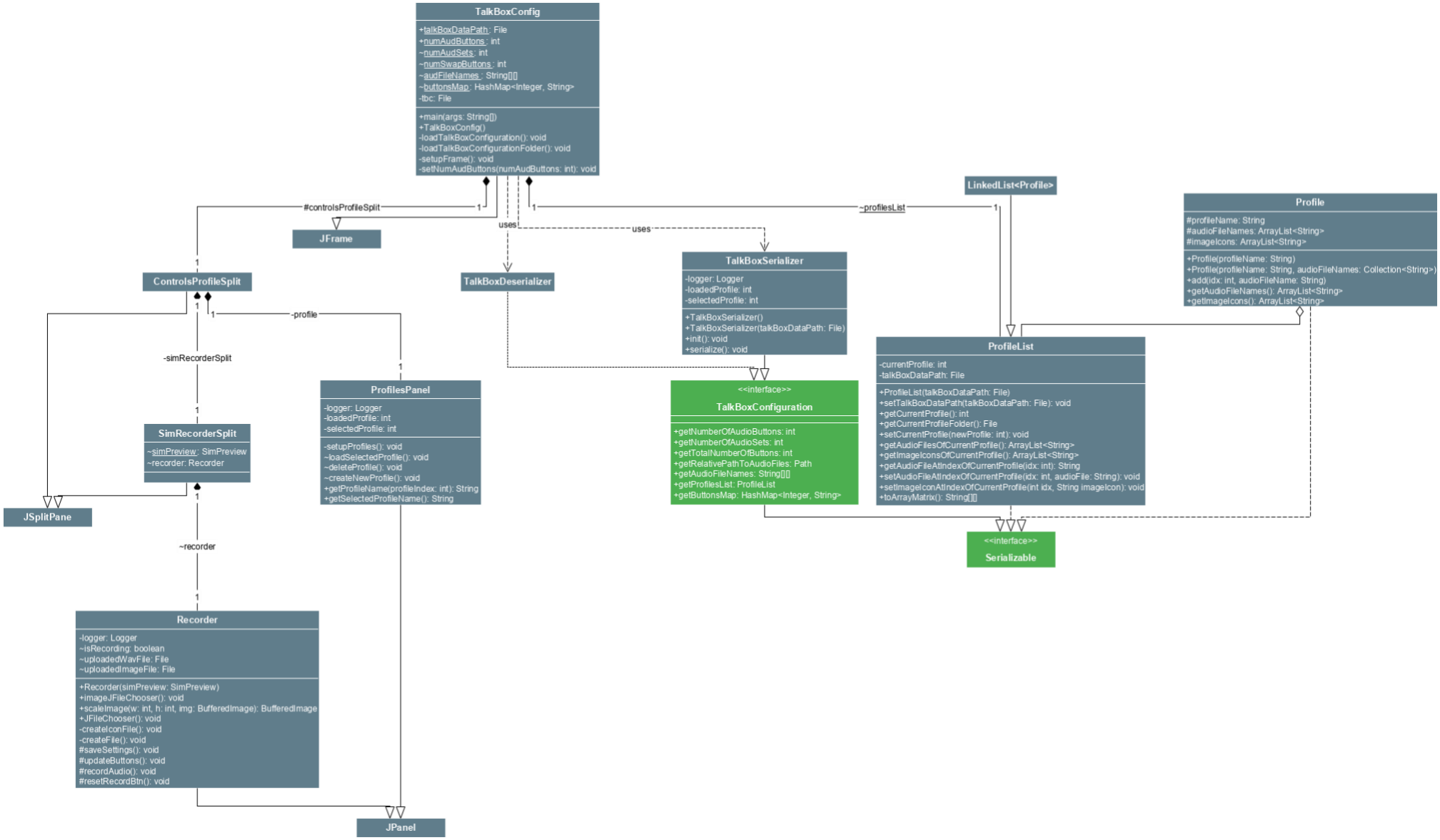
Field Visibility



UML Relations

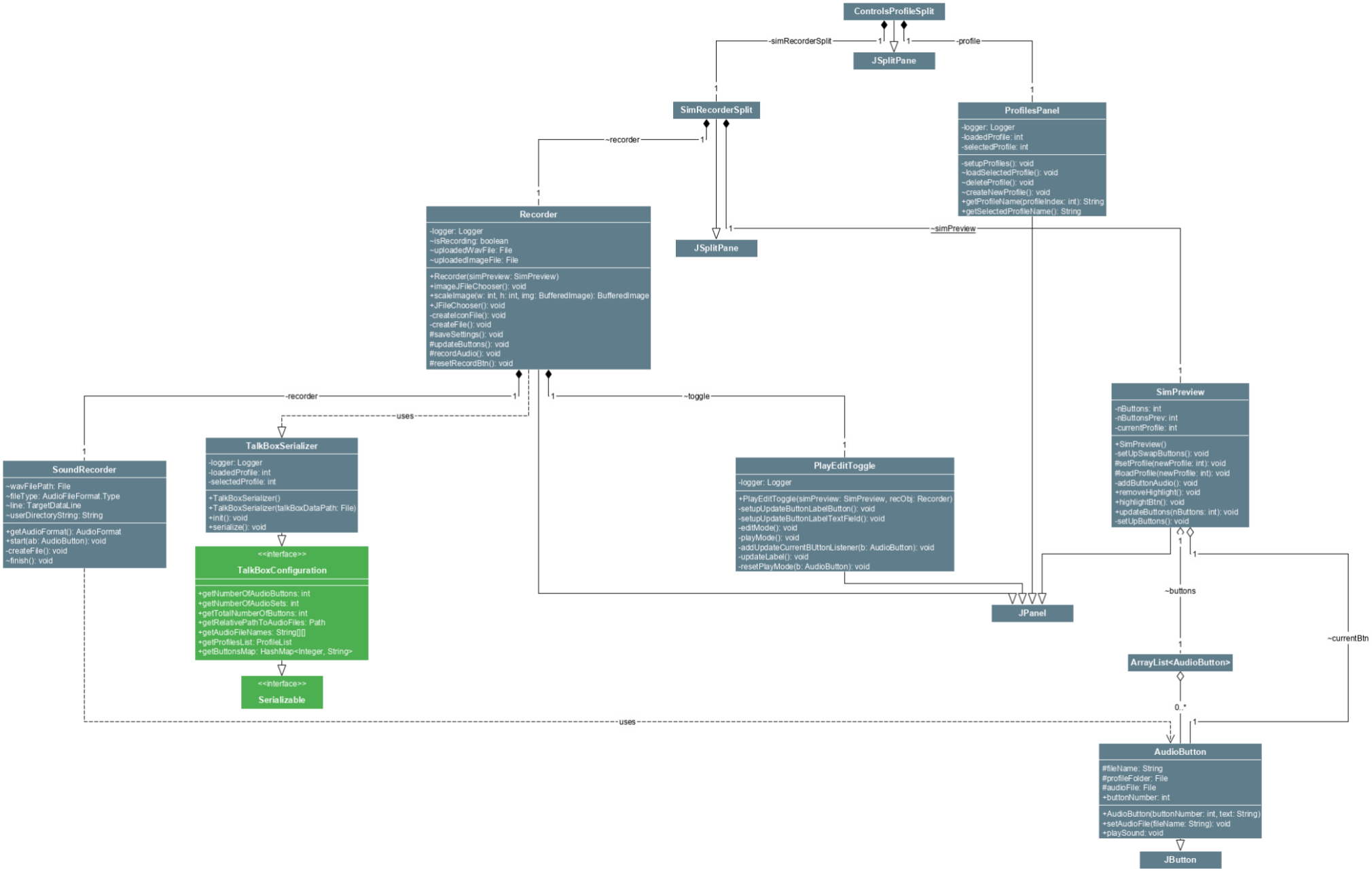
Class Diagrams

TalkBoxConfig High Level



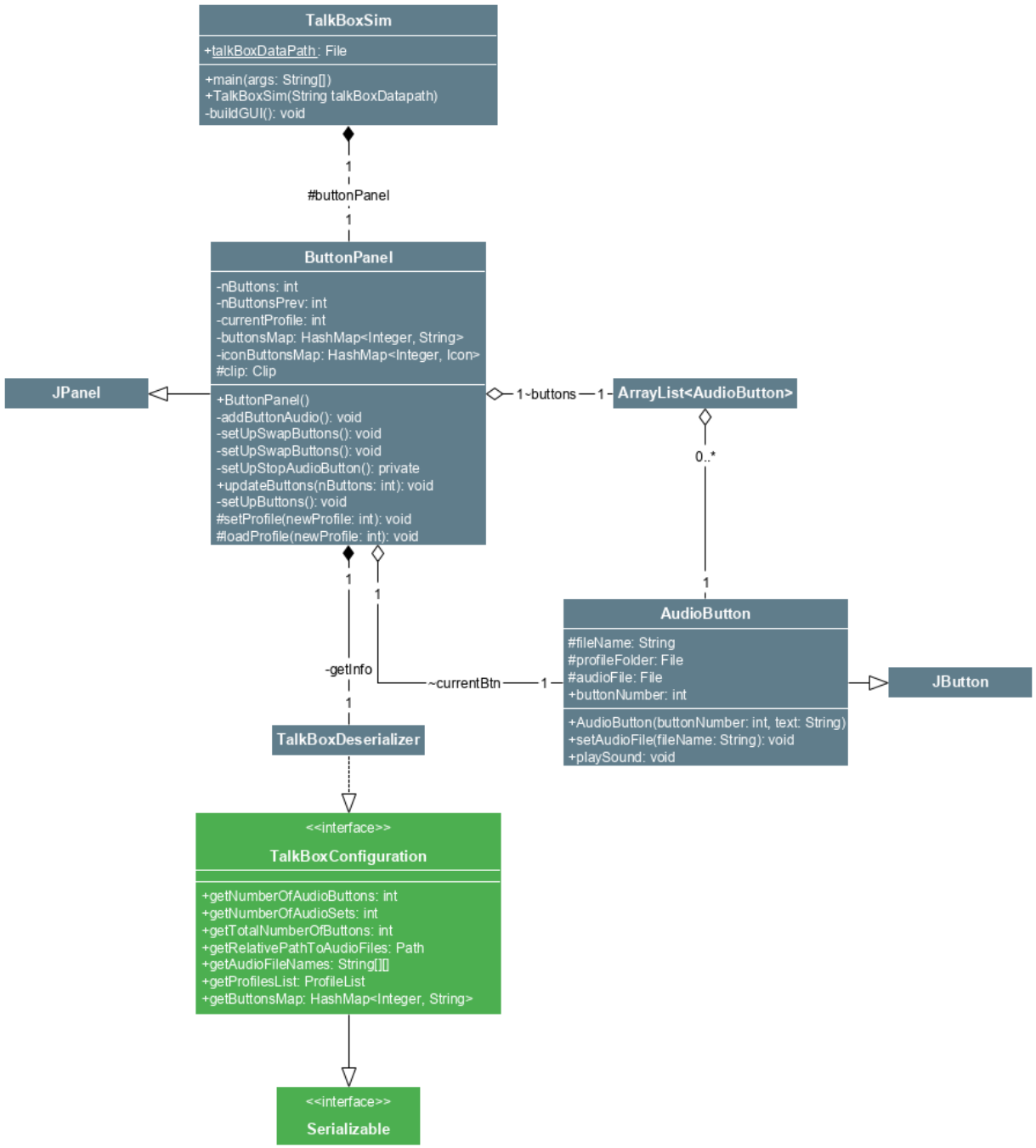
The TalkBoxConfig High Level class diagram provides the main classes involved in constructing the graphical user interface (GUI) of the Configurer. The diagram shows the TalkBoxConfig class at the top and how it is composed of exactly one ControlsProfileSplit class and one ProfileList class. It uses (depends) on the TalkBoxSerializer and Deserializer classes, which implement the TalkBoxConfiguration interface, for loading and saving TalkBox configuration data. The ControlsProfileSplit class itself is composed of exactly one SimRecorderSplit class and exactly one ProfilesPanel class. The ProfilesPanel class displays a list of profiles and provides buttons and methods for switching, loading, deleting, and creating profiles. The SimRecorderSplit class is composed of exactly one Recorder class and exactly one SimPreview class (shown here as just a field to reduce detail level). The Recorder class provides buttons and methods for recording audio and changing the layout, labels, and image icons. The SimPreview visually mimics the Simulator app and provides the same functionality but can be configured dynamically using the Recorder class.

TalkBoxConfig Low Level



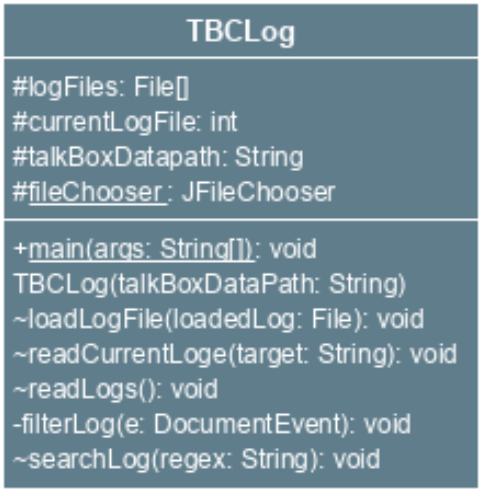
The TalkBoxConfig Low Level class diagram provides more details about the design of the Recorder and SimPreview classes. It shows that the Recorder is composed of exactly one SoundRecorder with methods for recording and saving audio files recorded by the user's microphone. The Recorder is also composed of a subpanel, the PlayEditToggle class that provides methods for switching between play mode and edit mode. This allows the user to easily playback audio in play mode, or to switch to edit mode and select buttons for configuring the button's icon, label, and associated audio file. The Recorder also uses a TalkBoxSerializer instance to save settings. The SimPreview also has an aggregation of exactly one ArrayList<AudioButton> which is itself is an aggregation of AudioButton classes. AudioButton classes provide the functionality of audio playback and compose the main portion of the SimPreview.

TalkBoxSim


TalkBoxSim
+talkBoxDataPath: File
+main(args: String[])
+TalkBoxSim(String talkBoxDatapath)
-buildGUI(): void
1
#buttonPanel
1
ButtonPanel
-nButtons: int
-nButtonsPrev: int
-currentProfile: int
-buttonsMap: HashMap<Integer, String>
-iconButtonsMap: HashMap<Integer, Icon>
#clip: Clip
+ButtonPanel()
-addButtonAudio(): void
-setUpSwapButtons(): void
-setUpSwapButtons(): void
-setUpStopAudioButton(): private
+updateButtons(nButtons: int): void
-setUpButtons(): void
#setProfile(newProfile: int): void
#loadProfile(newProfile: int): void
JPanel
1
~buttons
1
ArrayList<AudioButton>
0..*
1
1
1
-getInfo
1
~currentBtn
1
AudioButton
#fileName: String
#profileFolder: File
#audioFile: File
+buttonNumber: int
+AudioButton(buttonNumber: int, text: String)
+setAudioFile(fileName: String): void
+playSound: void
JButton
TalkBoxDeserializer
<<interface>>
TalkBoxConfiguration
+getNumberOfAudioButtons: int
+getNumberOfAudioSets: int
+getTotalNumberOfButtons: int
+getRelativePathToAudioFiles: Path
+getAudioFileNames: String[][]
+getProfilesList: ProfileList
+getButtonsMap: HashMap<Integer, String>
<<interface>>
Serializable

The TalkBoxSim class diagram is roughly a subset of the TalkBoxConfig Low Level class diagram. Here, there TalkBoxSim class is composed of exactly one ButtonPanel class. This ButtonPanel class is the main panel that the app displays. The ButtonPanel is composed of exactly one TalkBoxDeserializer, which it depends on to load the saved TalkBox configuration and to switch profiles (audio sets). The ButtonPanel also has an aggregation of exactly one ArrayList<AudioButton> which is itself is an aggregation of AudioButton classes. AudioButton classes provide the functionality of audio playback and compose the main portion of the ButtonPanel. As shown elsewhere, the TalkBoxDeserializer implements the TalkBoxConfiguration interface.

TBCLog

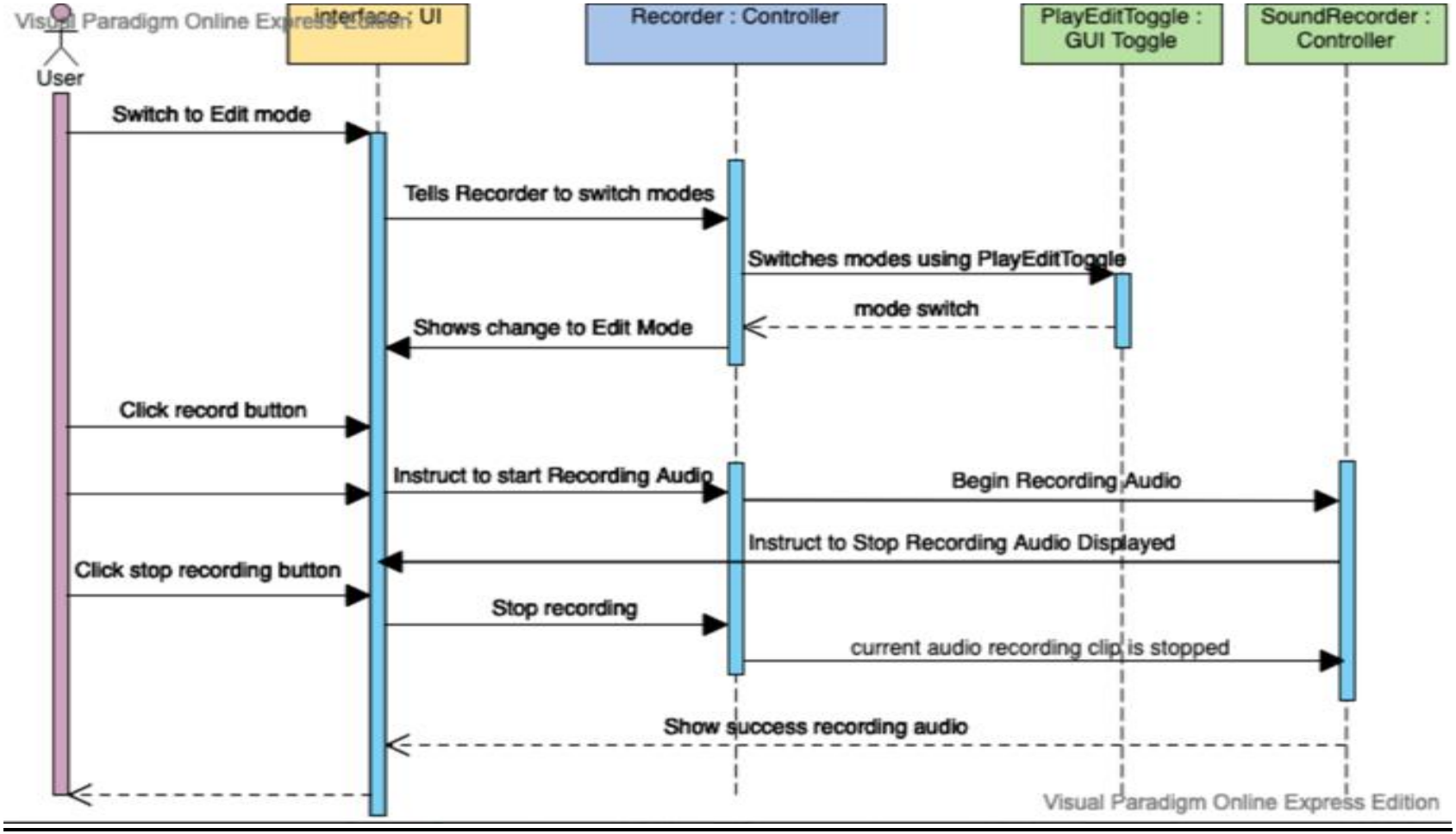


The TBCLog class diagram is very simple as it consists of a single class. The class does not interact directly with other classes or parts of TalkBox. The class diagram thus serves primarily to record the important fields and methods contained in the TBCLog app. The methods provided allow the TBCLog app to load log files, to jump to the previous or next chronological log, and to filter logs using a search field.

Sequence Diagrams and Maintenance Scenarios

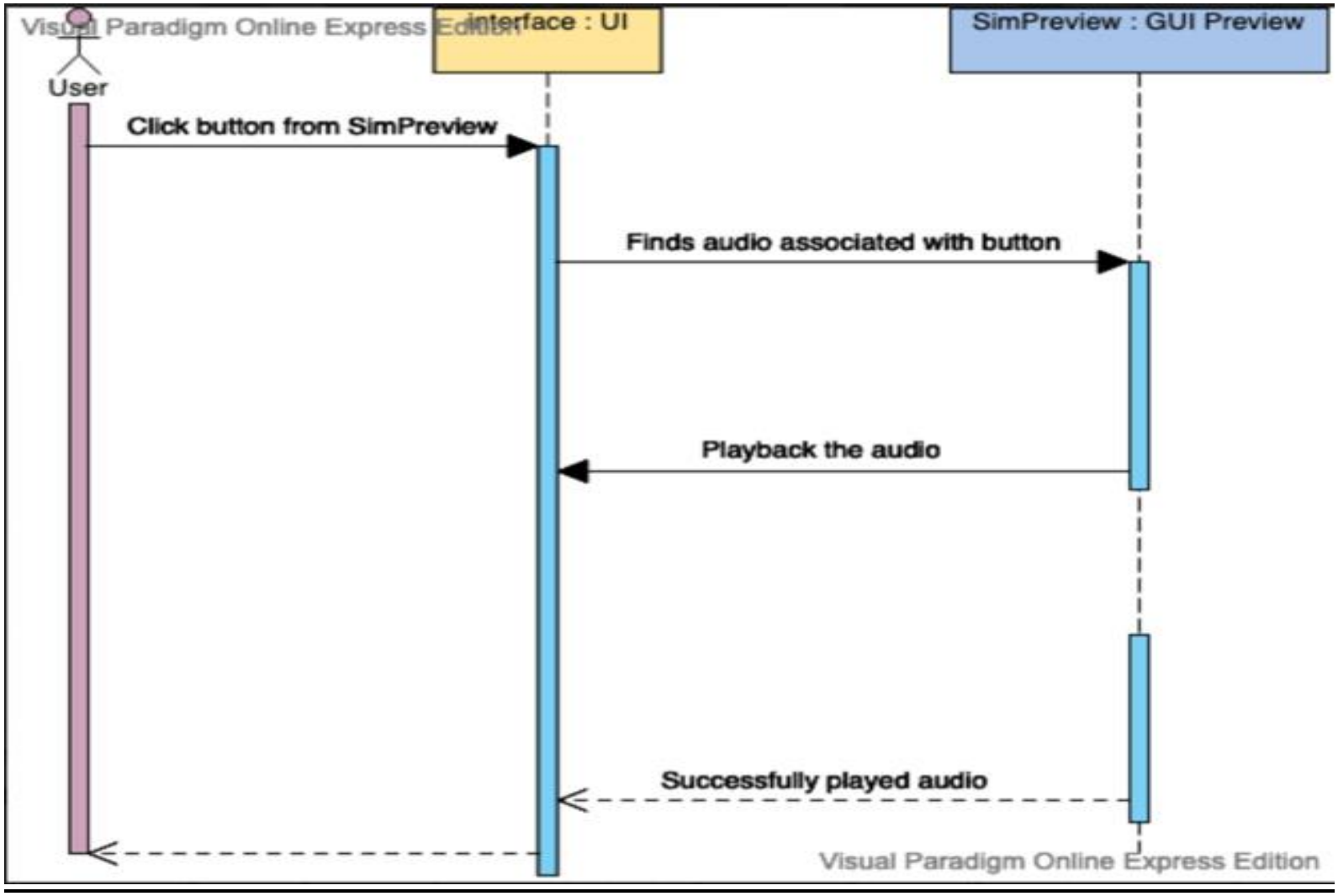
TalkBox Configuration

Record Audio:



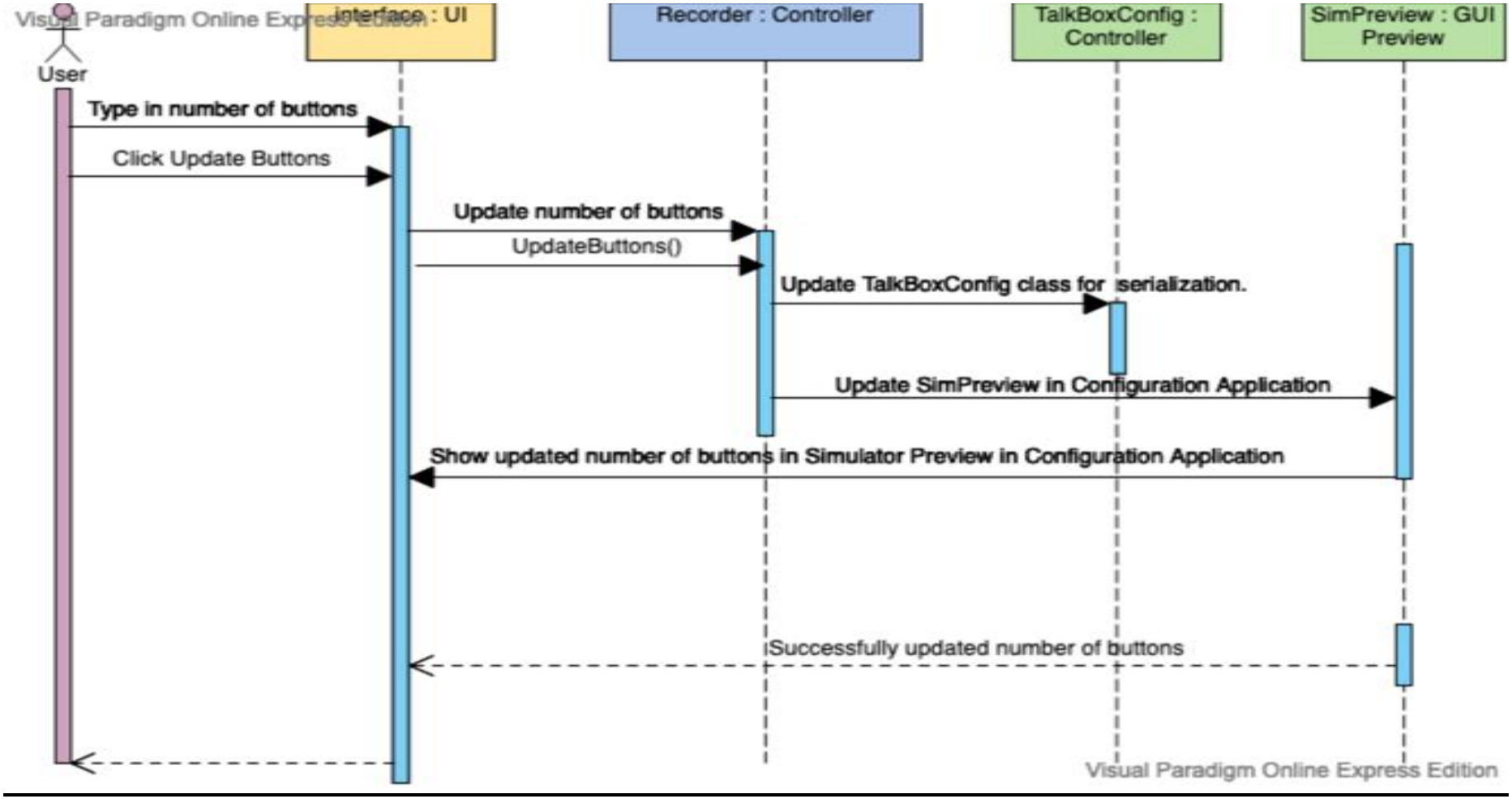
This is the sequence diagram for the record audio functionality of the TalkBox application. This functionality operates as follows: User must switch to edit mode, this tells the recorder class that the mode is being switched and hence a call to the PlayEditToggle class is made, and the mode is changed and shown to user. After this the user clicks the record button and a call to the recorder class is made, which then calls the SoundRecorder class which opens up an input audio stream to record. The user then stops recording and the SoundRecorder is instructed to close the current audio clip input, and recording is stopped. The user sees that the recording has stopped successfully, and that they have recorded audio successfully. Any maintenance regarding the recording audio feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

Playback Audio from a Button:



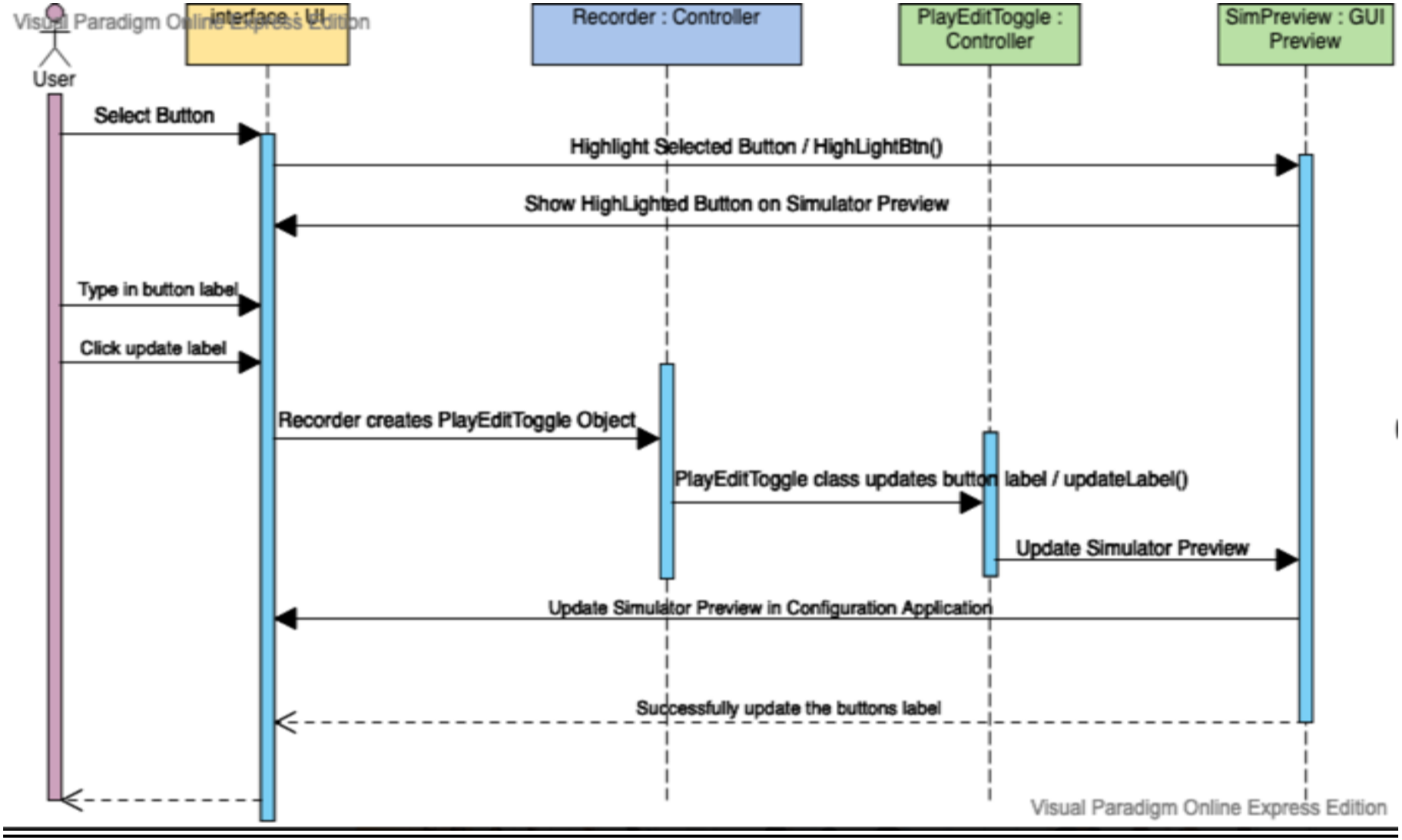
This is the sequence diagram for the user playing back audio from a button in the configuration application. The user first clicks a button that has an associated audio file, after which a call to the SimPreview class is made, which respectively contains the functionality to playback audio associated with a button. The simPreview class is essentially a preview of the simulator inside the configuration application. This completes the playback audio sequence. Any maintenance regarding the playback of audio feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

Update Number of Buttons:



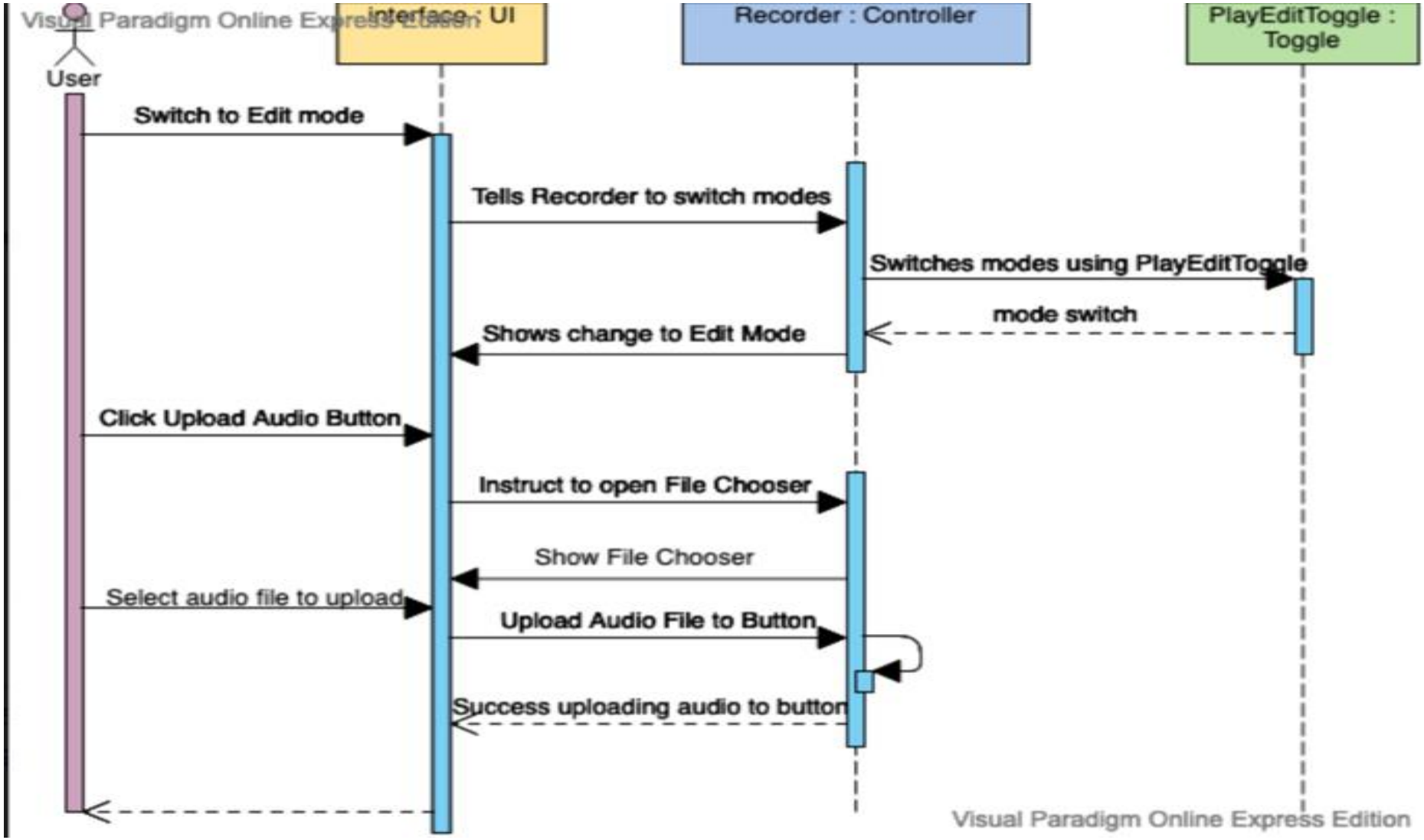
This is the sequence diagram for updating the number of buttons on the simulator through the configuration application. The user first types in a number of buttons and either hits enter on their keyboard, or clicks the update buttons button. This then calls the recorder controller class which updated a field inside the TalkBoxConfig main class that contains the number of buttons for the simulator. This field is used for serialization for the simulator application. Alongside this, the recorder also updated the simPreview class by updating the number of buttons of the simulator preview, and applying this change to the GUI in the configuration application. After the update to SimPreview, the user is shown the updated number of buttons successfully. Any maintenance regarding the updating number of buttons feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

Update Button Label:



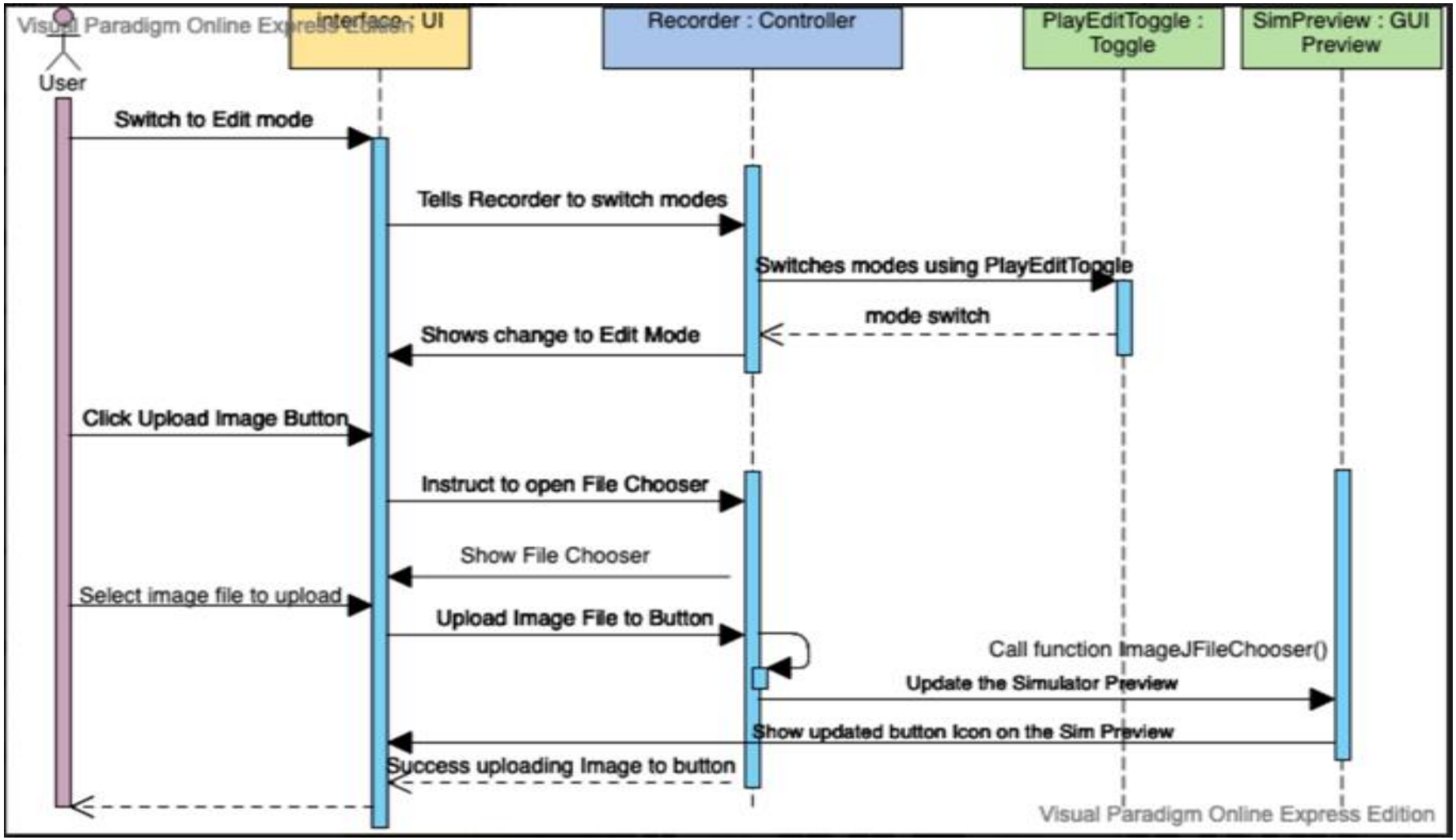
This is the sequence diagram to update the label on each button. The user first selects a button and the button is highlighted using a call to the SimPreview class. The user then types in the label they wish to have on the button, and this instructs the recorder to create a PlayEditToggle object which allows mode switches as well as updating of the button label. After the label is updated within the PlayEditToggle class, an update is done to the SimPreview class to allow the user to see the respective change of the label. The label is then updated and successfully shown to the user. Any maintenance regarding the updating button label feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

Upload Audio to Buttons:



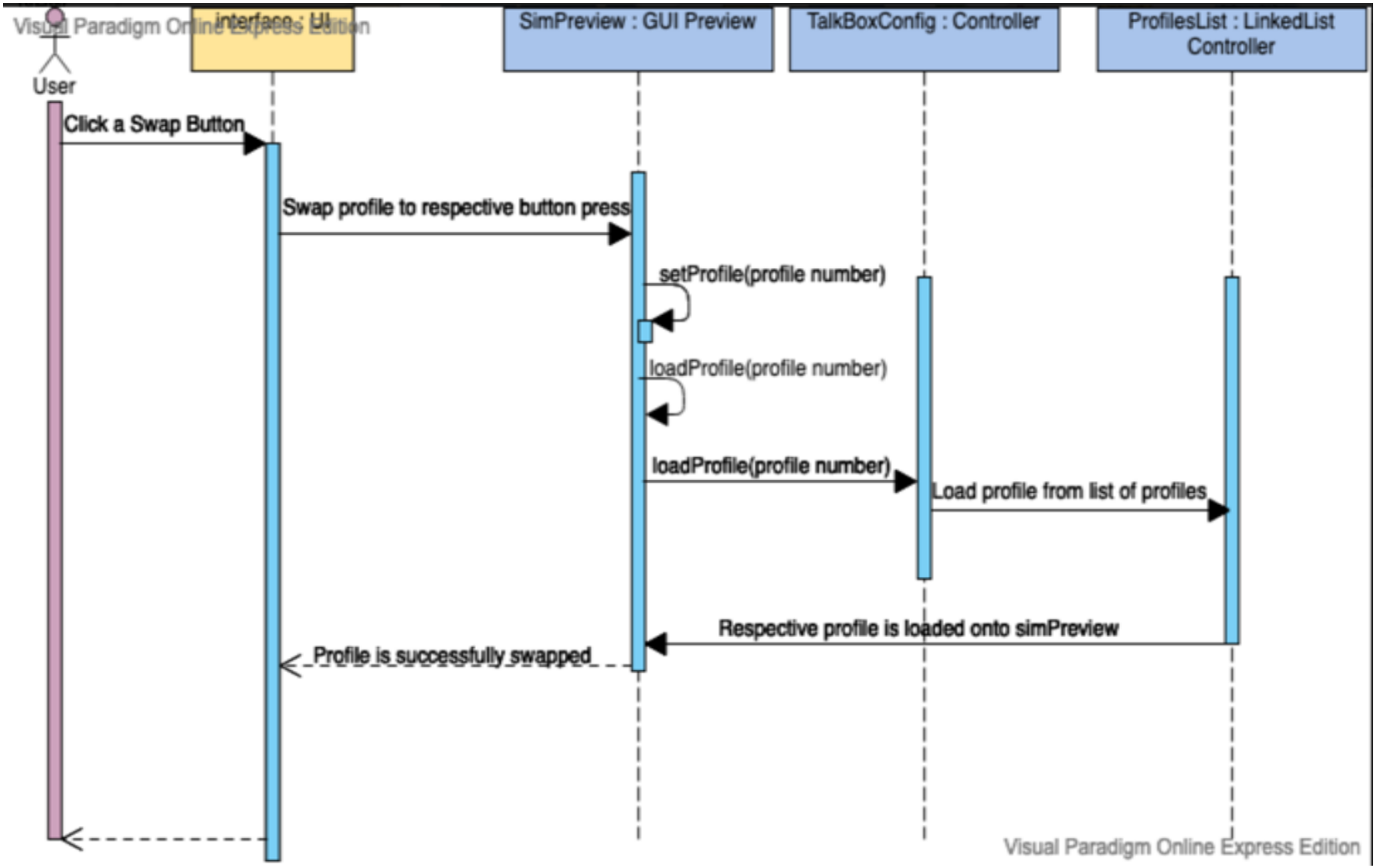
This is a sequence diagram showing the upload audio to a button procedure within the program. User switches to edit mode, this tells the recorder class that the mode is being switched and hence a call to the PlayEditToggle class is made and the mode is changed and shown to user. The user then clicks the update audio button, and a call is made to the recorder class to allow the sequence to progress. User is shown a file chooser and asked to select an audio file (.wav). After the user has selected an audio file, the recorder class within itself, calls method to add the audio to the associated button. Once this is complete the sequence is successful and the audio is uploaded to the button, and can now be played back by the user. Any maintenance regarding the upload audio feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

Upload Images to Buttons:



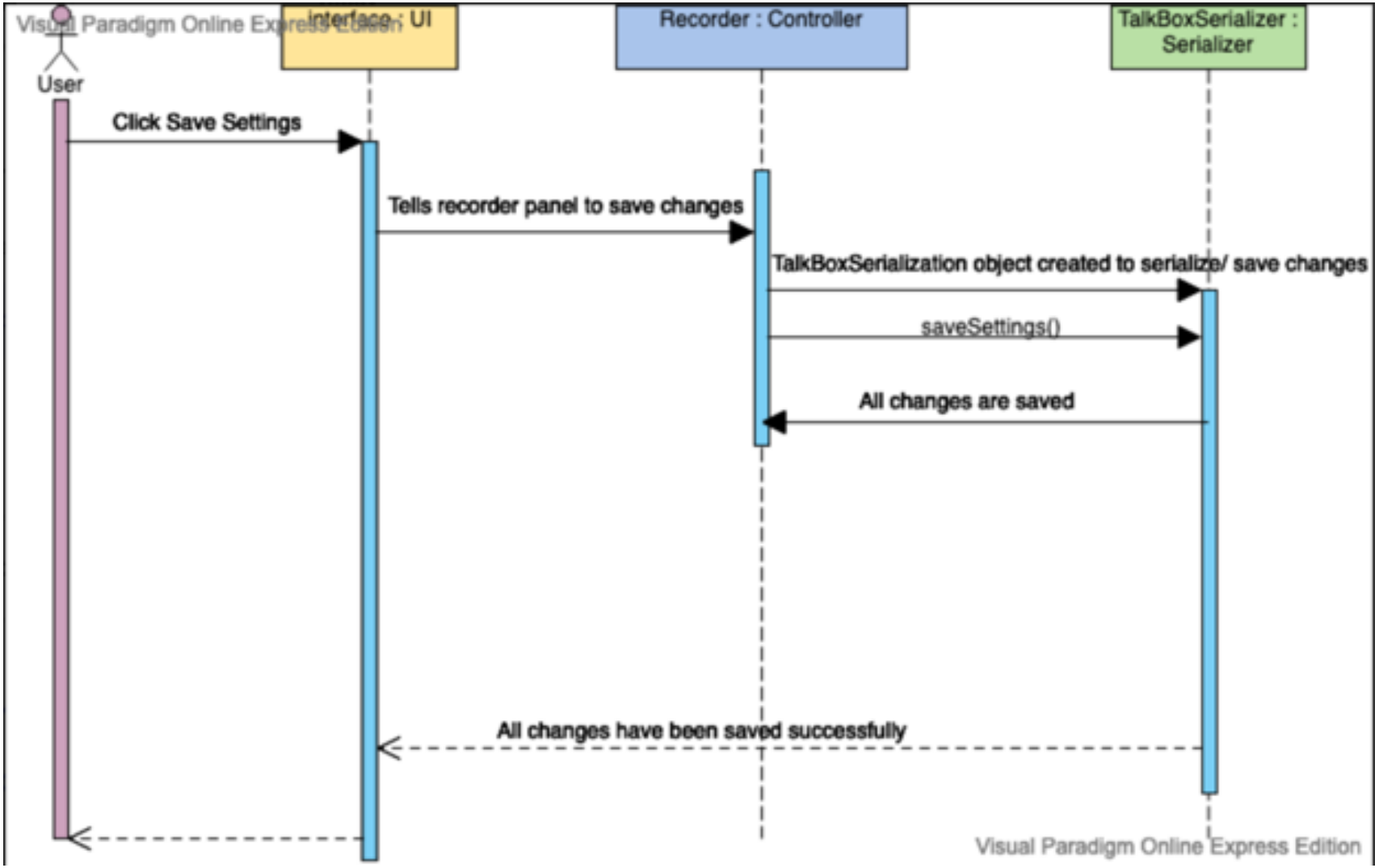
This is a sequence diagram showing the upload image to a button procedure within the program. User switches to edit mode, this tells the recorder class that the mode is being switched and hence a call to the PlayEditToggle class is made and the mode is changed and shown to user. The user then clicks the update image button, and a call is made to the recorder class to allow the sequence to progress. User is shown a file chooser and asked to select an image file. After the user selects a file a call is made within the recorder class to update the buttons associated image file, and then a call is made to SimPreview. This call to SimPreview allows the buttons icon to be updated on the GUI, and shown to the user. The sequence is then complete and an image is uploaded to a button successfully. Any maintenance regarding the upload images feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

Swap Profiles:



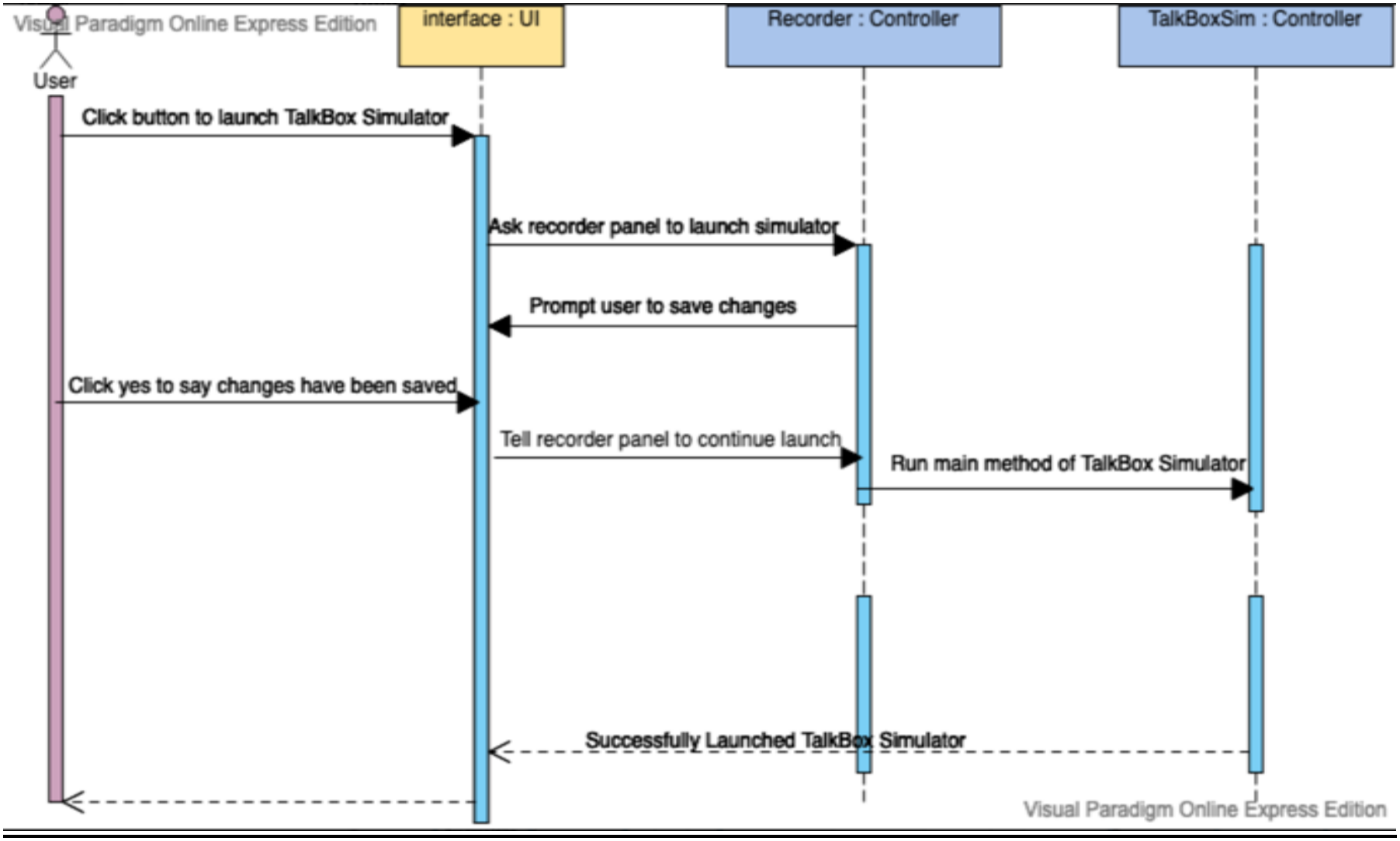
This is the sequence diagram for swapping profiles on the simulator preview inside the configuration application. The user first clicks a swap button and a call is made to SimPreview which respectively updates the simulator preview to have the correctly swapped profile. This is done by calling the TalkBoxConfig class which holds a profile list field. This class finally calls the ProfileList class (which is a list of created profiles). From this class (list), the respective profile is returned back to the simPreview class and the simulator preview in the configuration is updated with the newly swapped profile. Any maintenance regarding the recording swap profiles feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

Save Settings / Serialize Changes:



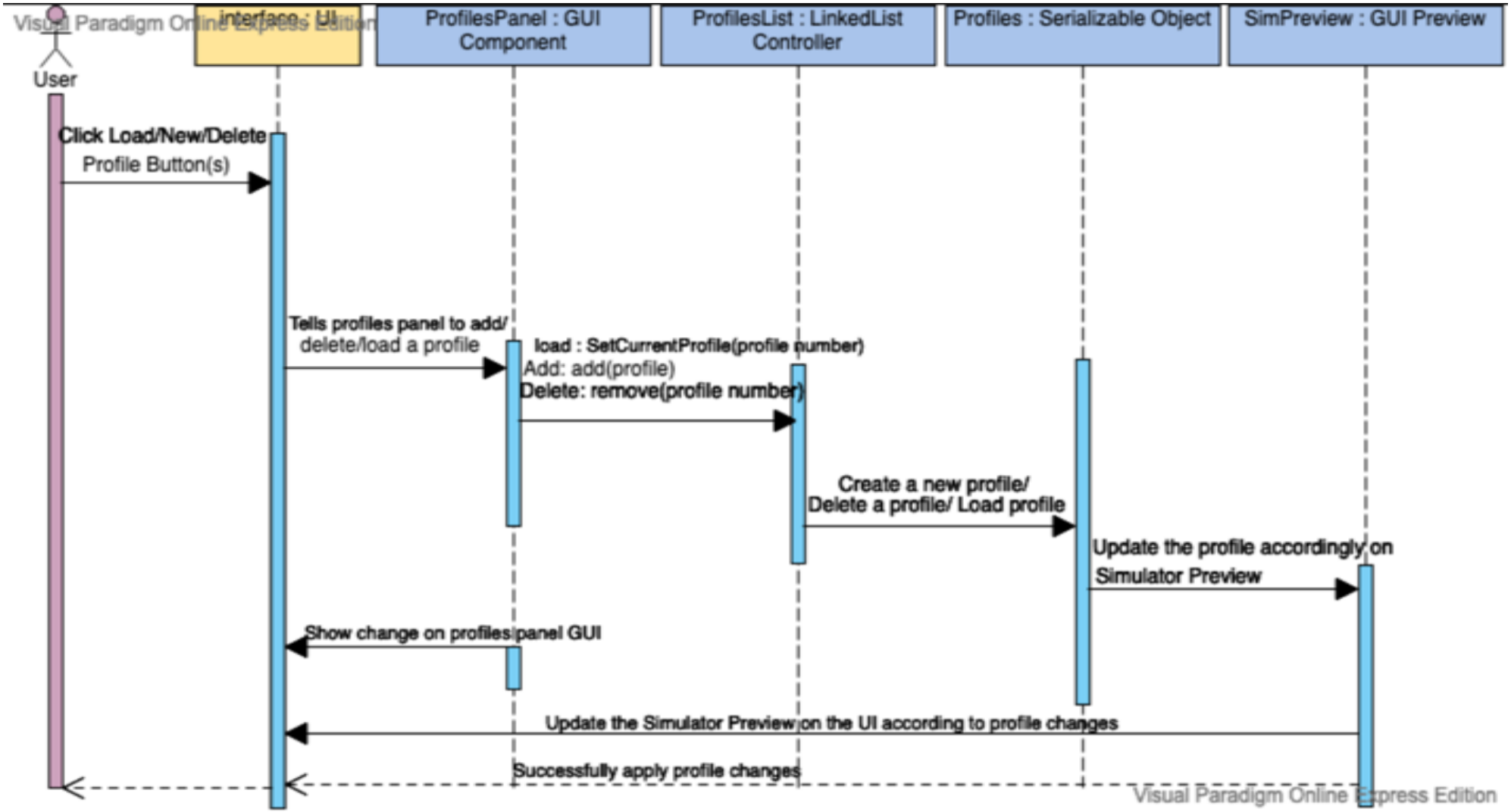
This is a sequence diagram for saving the settings / serializing the changes from the configuration application for other applications such as the simulator application and TalkBox Configuration Log application. The user first clicks save settings button, this button is contained within the recorder panel and thus the Recorder class. The Recorder class then creates a TalkBoxSerialization object to serialize the changes. The saveSettings() method is called within the TalkBoxSerialization class, this method puts all the respective changes into a .tbc file. All changes are then saved successfully and the sequence is complete. Any maintenance regarding the save settings/serialize changes feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

Launch TalkBox Simulator Application:



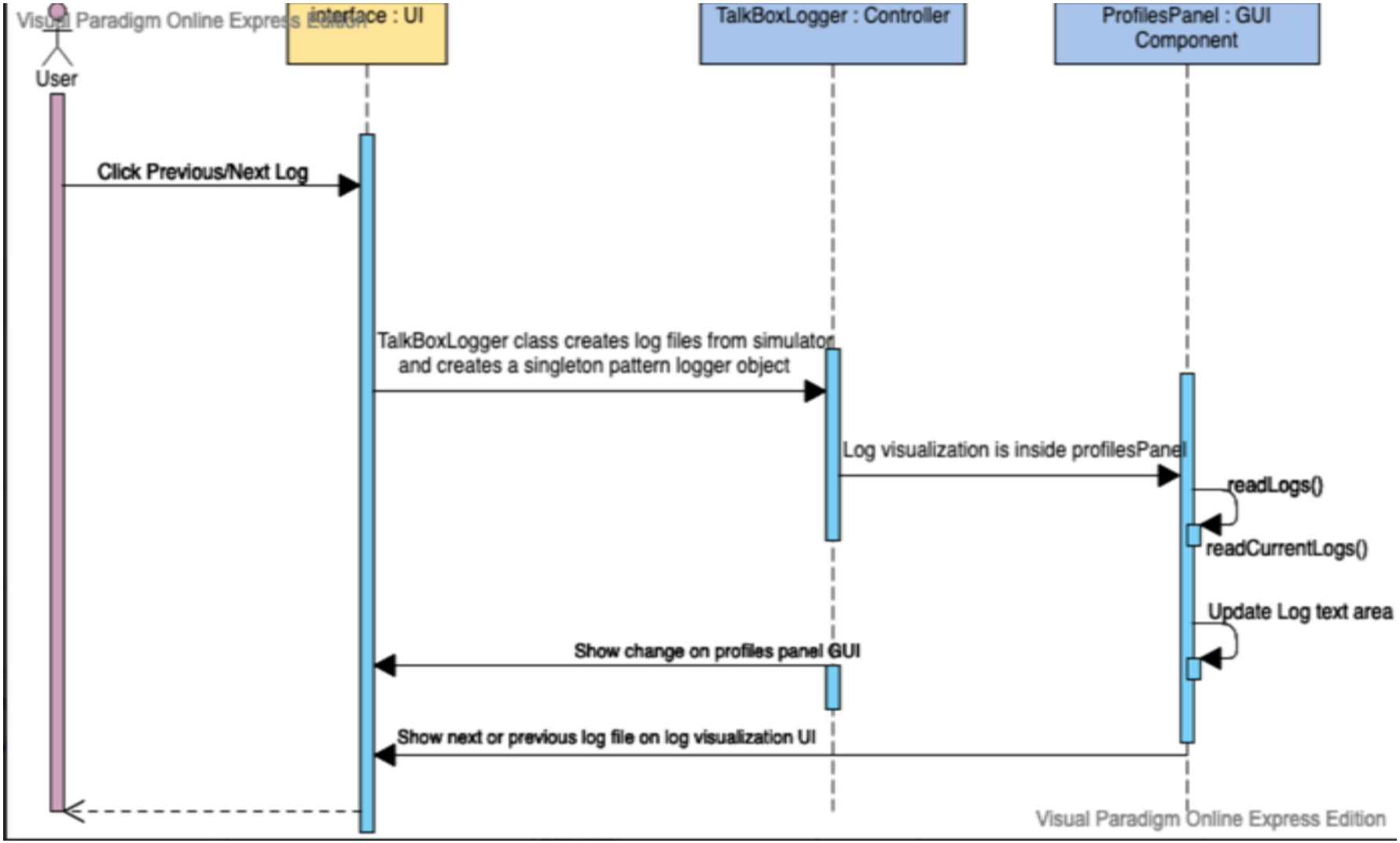
This is the sequence diagram for launching the simulator application from the configuration application. The user first clicks the Launch Simulator button, so a call is made to the Recorder class to tell the class the user has clicked the button. The Recorder class then opens up a window that prompts a user to save changes before launching the simulator application. The user hits yes, they have saved all changes. The Recorder class then makes a call directly to the TalkBoxSim class main method, which is a standalone application. The application is then successfully launched and displayed to the user. The sequence for launching the TalkBox Simulator is hence complete. Any maintenance regarding the launch simulator application feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

Load/Delete/New Profile:



This is the sequence diagram for loading/deleting/creating a new profile. The user first hits the respective button (load/new/delete profile). A call is then made to the ProfilesPanel GUI class that passes the respective instruction. This calls the Linked List controller class called ProfilesList, which contains the methods setCurrentProfile(profile number) (for loading), add(profile) (adds a profile), remove(profile number) (deletes a profile). This class calls a class known as profiles, which is a serializable class that represent an individual profile. This class adds, removes, or loads a profile respectively. After it has done this, it will pass the respective parameters and update values to the SimPreview class which will update the profile on the simulator preview. After this is done the UI is updated with the respective profile changes, whether it be loading, creating, or deleting a profile. The sequence for loading, creating, or deleting a profile is hence complete and successful. Any maintenance regarding the load/delete/new profile feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

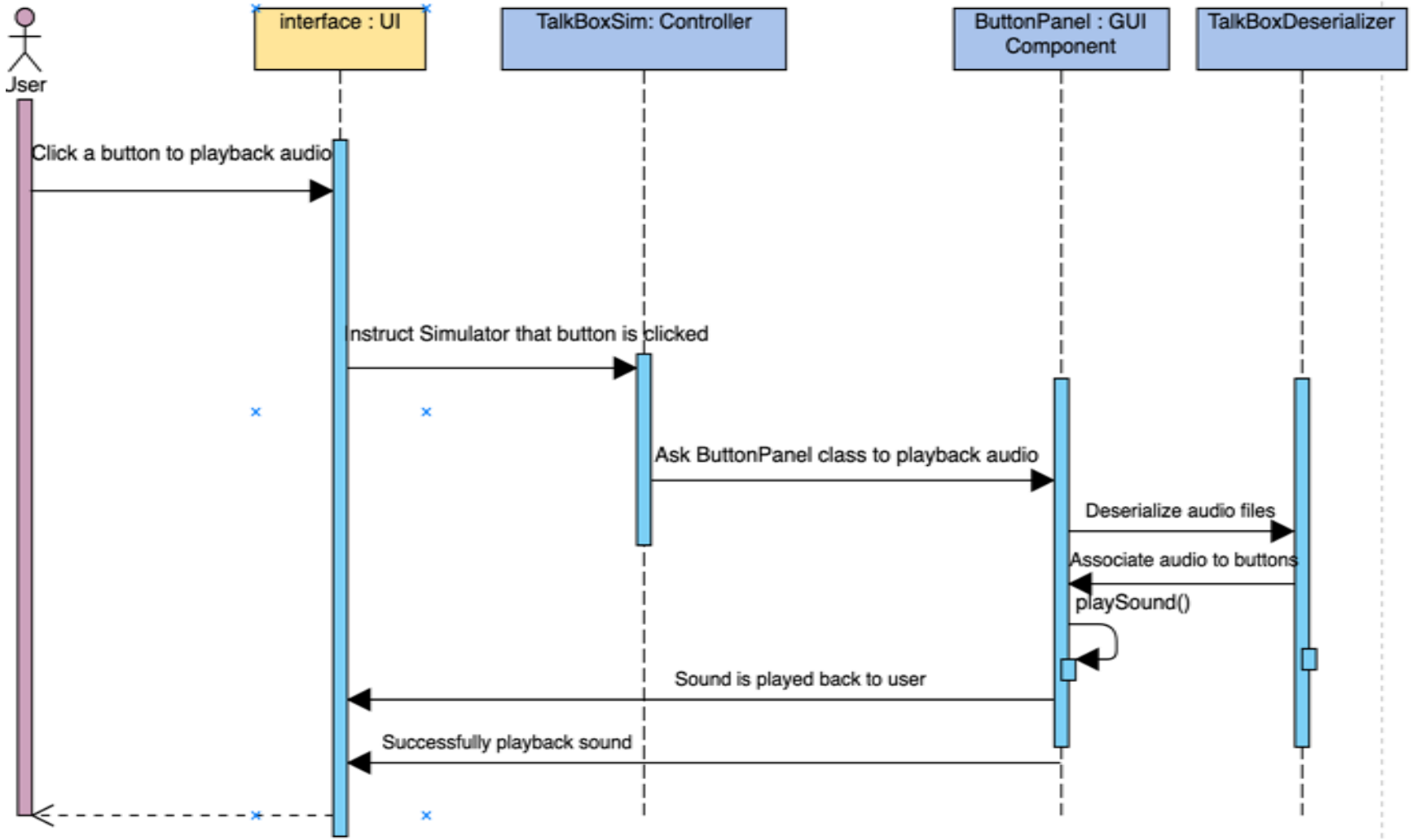
Next/Previous Log:



This is the sequence diagram for loading the next or previous simulator log onto the log preview inside the configuration application. The user first clicks Previous/Next Log after which a call is made to the TalkBoxLogger controller class. This class creates the log files from the simulator using a singleton design pattern global logger object. The log visualizations are then read onto the ProfilesPanel class GUI component, which contains the logging visualization. This visualization is updated from the files being read inside the TalkBoxLogger controller class. Within the ProfilesPanel class, readLogs(), readCurrentLogs(), and updateLogs() methods update the log to match the users action of next or previous log. After this the GUI is fully updated and the sequence for next or previous log is completed successfully. Any maintenance regarding the next/previous log feature in the TalkBox Configuration can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

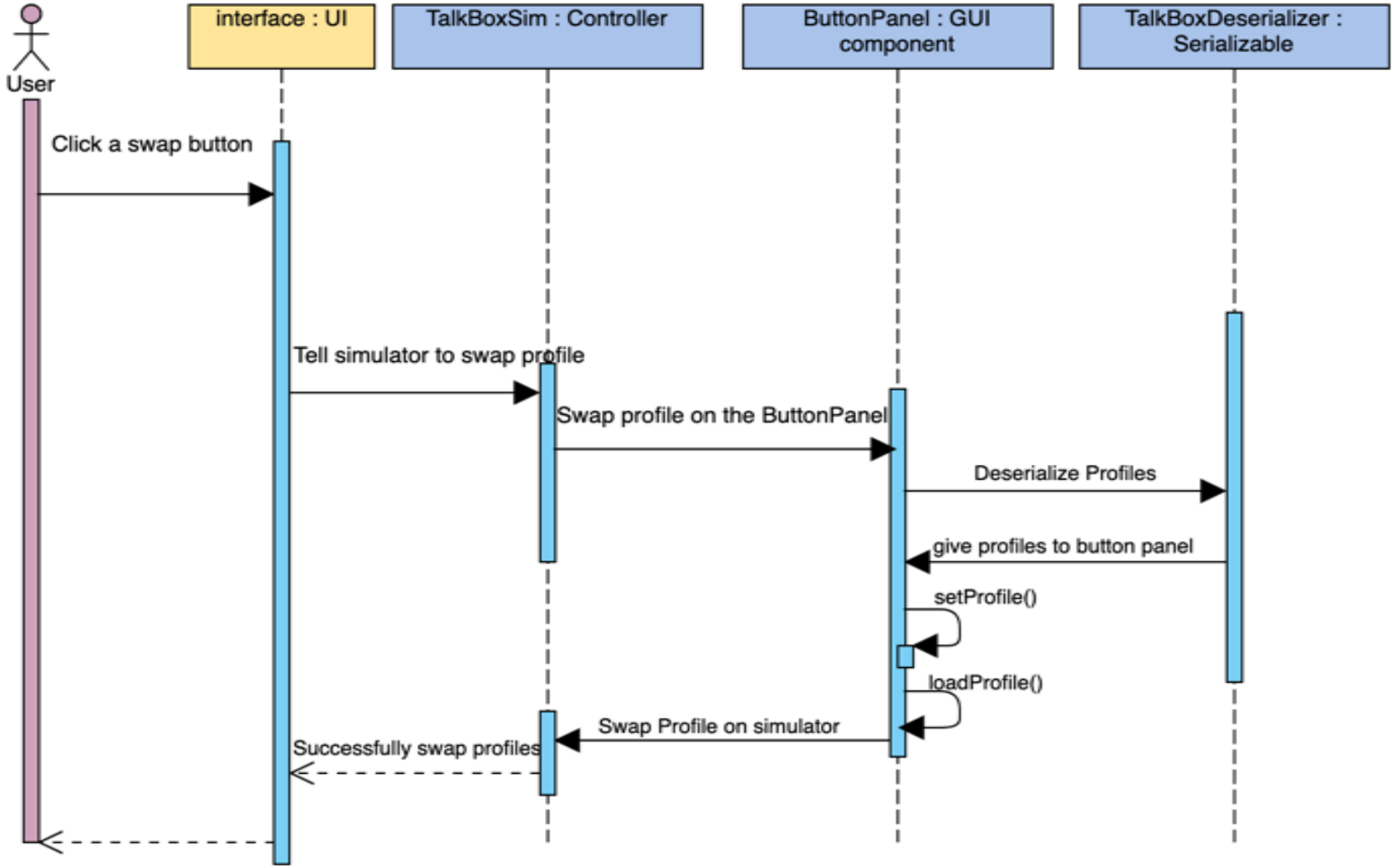
TalkBox Simulator

Playback Audio from a Button:



This is the sequence diagram for playing audio from a button inside the TalkBox Simulator application. The user first clicks a button to playback audio from, this calls the TalkBoxSim controller class and lets it know the button was clicked, this class then invokes the ButtonPanel class to playback audio from the button. The ButtonPanel class first calls the TalkBoxDeserializer class to deserialize all the changes from the configuration application. After this deserialization, the audio is associated with each button respectively. Then the users initial command continues its sequence, and the playSound() method is called inside the ButtonPanel on the button that has been clicked. This then successfully plays back the audio to the user, and the sequence is thus complete. Any maintenance regarding the playback audio feature in the TalkBox Simulator can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

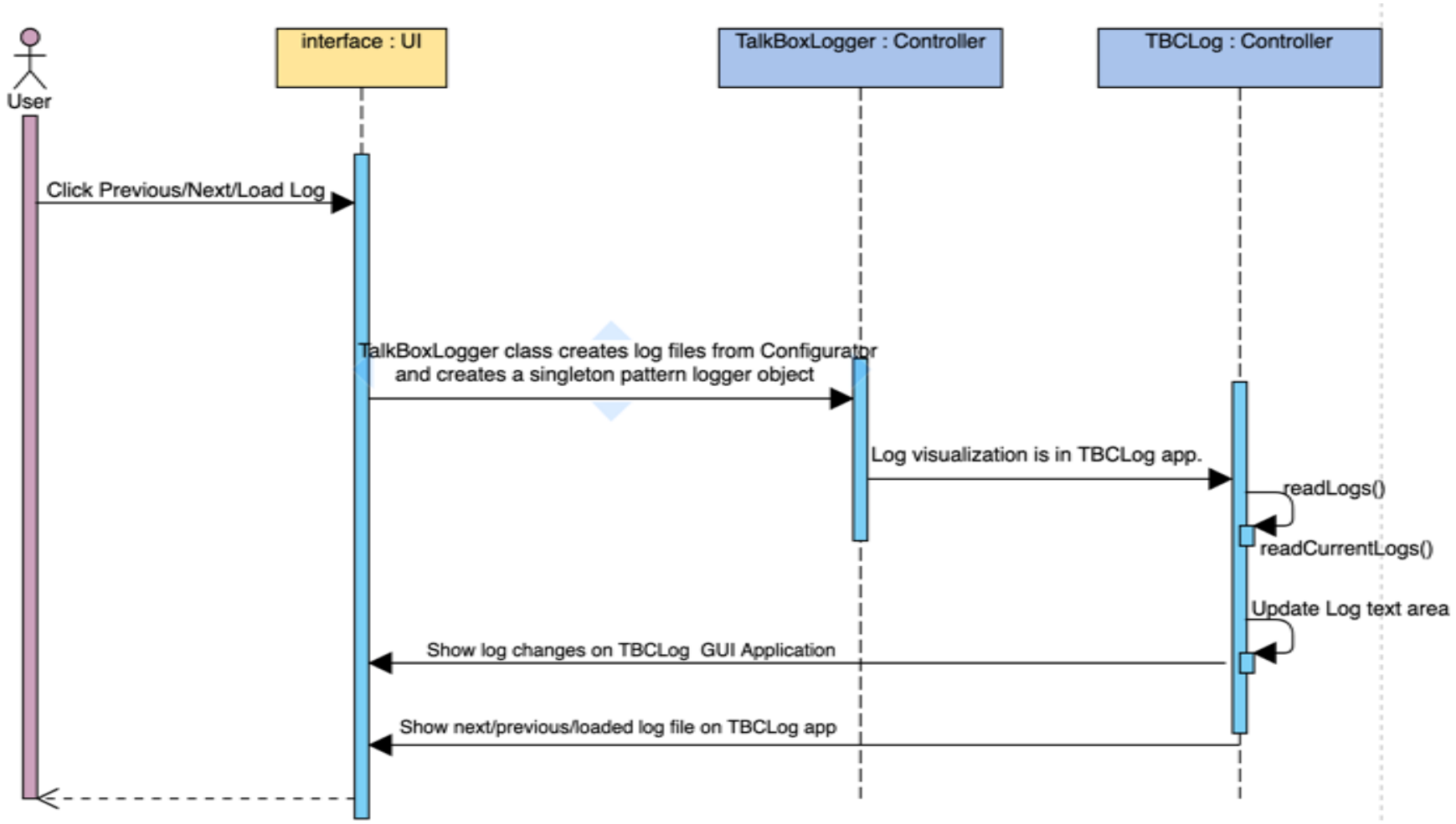
Swap Profiles:



This is a sequence diagram to swap profiles within the TalkBox Simulator application. The user first clicks a swap button (any of the 4 swap buttons), and a call is made to the TalkBoxSim controller class. This class then calls the ButtonPanel class, which calls the TalkBoxDeserializer class to deserialize the changes made in the configuration application. The deserialization is done, and the ButtonPanel within itself calls the methods setProfile(), and loadProfile(), which set and load a profile onto the simulator. This update is then done to the TalkBoxSim class as well as the GUI, and the user has successfully swapped a profile(s). Any maintenance regarding the swap profiles feature in the TalkBox Simulator can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.

TalkBox Configuration Log (TBCLog)

Next/Previous/Load Log:



This is a sequence diagram for Next/Previous/Loading a log in the TalkBox Configuration Log application. The user first clicks Previous/Next Log after which a call is made to the TalkBoxLogger controller class. This class creates the log files from the configuration using a singleton design pattern global logger object. This class then invokes the TBCLog controller class which reads the logs that were created by the TalkBoxLogger from the TalkBox Configuration application. The TBCLog class then calls the methods: readLogs(), readCurrentLogs(), and updates the logging text area with the respective logs, whether it be the previous log, the current log, or the next log. It updates the GUI by reading from the respective created log files, and the users command of Previous/Next/Load Log is successfully completed and shown on the GUI component. This ends the logging sequence. Any maintenance regarding the next/previous/load log feature in the TalkBox Configuration Log can be handled using the above diagram to make respective maintenance updates and changes, as each classes relation to this functionality is shown in the diagram.