Software Design Document

by

Aaron Ennis
C00190504

Supervisor: Paul Barry

Music Application Project

Date: 18/04/2018

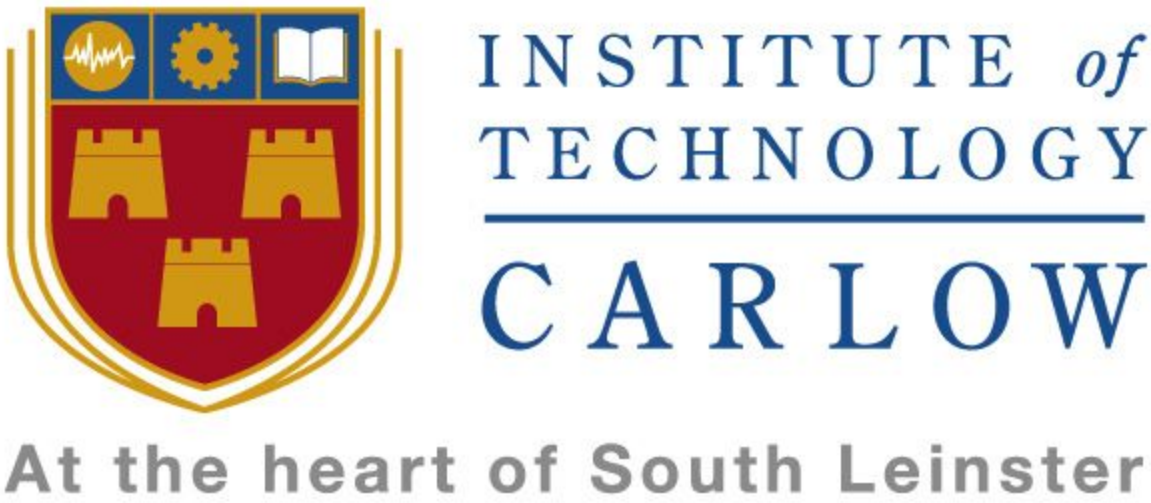


Table of contents

Introduction

Purpose

This software design documents purpose is to clearly show how the application was designed for both the internal (code) and external (file management + UI) components. It should be readable for a person with a software background and also for a user without a software background.

Scope

This project is a music application aimed towards musicians that write and record their own music. The user will be able to run the application on their computer or laptop. When the application is run, the user will be able to record a piece of music and save the audio. When they save the clip of music, the application will then transcribe the piece of music to tablature form. The projects aim is for guitarists to easily write, record, read, and playback music they have written.

Reference Material

Software Design Document (SDD) Template

Link to PDF

This PDF document contains the template I used to base this software design document off of. It was used to create the headings for this document.

Ukulele Tuner

Link to ukulele tuner document

This document is for a ukulele tuner that was created by Mzucker on GitHub. It is a ukulele tuner written in Python using the Fast Fourier Transformation. This document not only gave me access to important information that I would need to know to complete this project, it also gave me a good foundation to begin constructing the internal components of the application.

System Overview

Description

This application was created to make life easier for musicians that write and record their own music. The user will be able to run the application on their computer or laptop. All a user needs is a working microphone for the laptop or computer, and a guitar to use for the recording.

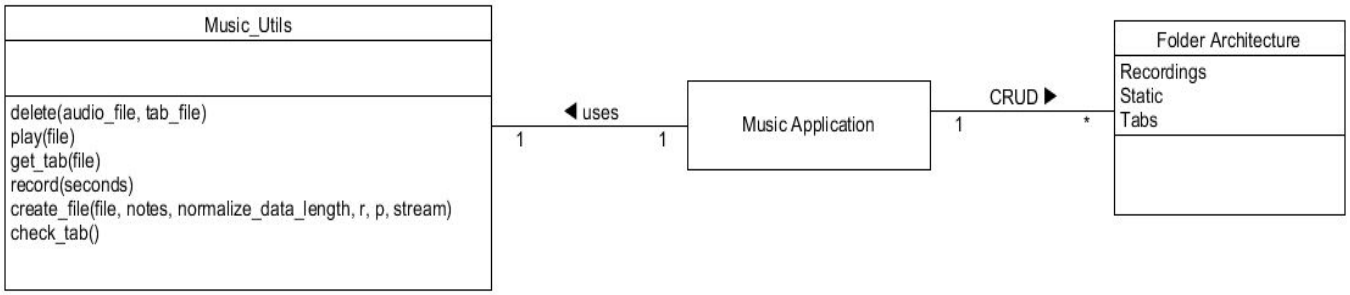
When the application is run, the user will have the option to record a piece of music. The user has the option to either record a 5, 10, 15, 20, 25, or 30 second clip. The audio saved by the user is saved locally and can be accessed at any time as it is stored in a “Recordings” folder in the applications folder. The audio file is saved in .WAV form so it can be played outside of the application. When the user records and saves a clip of music, the application will then transcribe the piece of music to tablature form. This is how the application makes life easier for musicians as it visualises their music.

The design of the User Interface (UI) should be as simple as possible. This is because the application was designed to make life easier for a user. The less button clicks a user will have to do to achieve the task they want to do the better. The UI should be clearly readable and easy to navigate even for first time users.

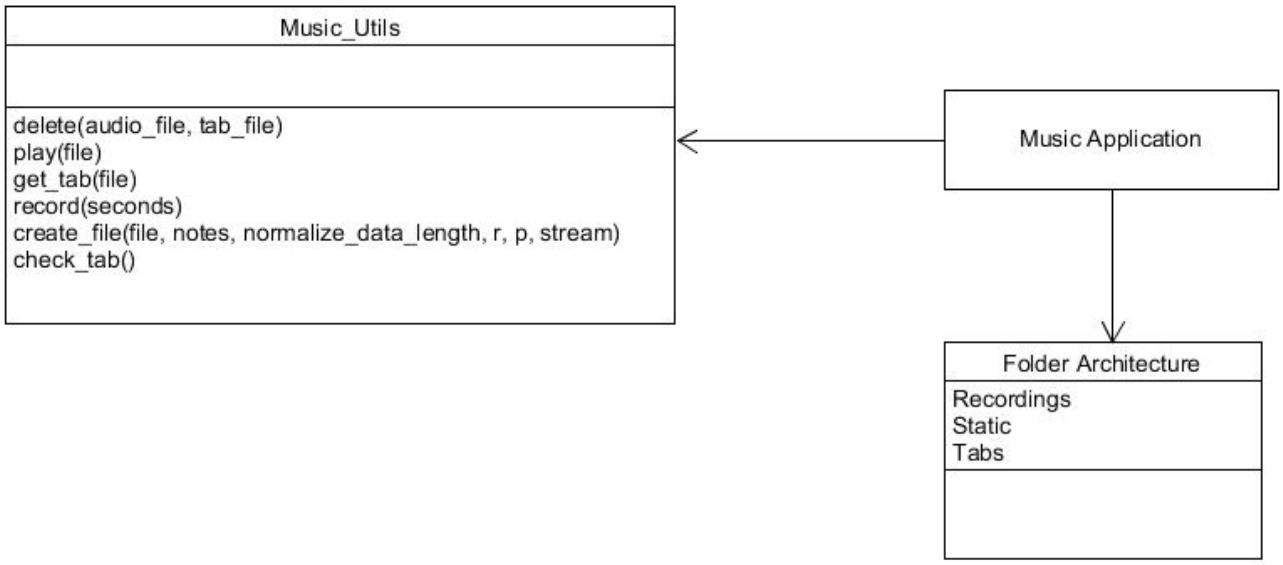
The idea for this project was created to stop a common problem musicians have when saving recordings of their music. The problem is when a recording is saved on a machine and forgotten about, it can be hard to remember how to play that piece of music. The projects aim is for guitarists to easily write, record, read, and playback music they have written.

System Architecture

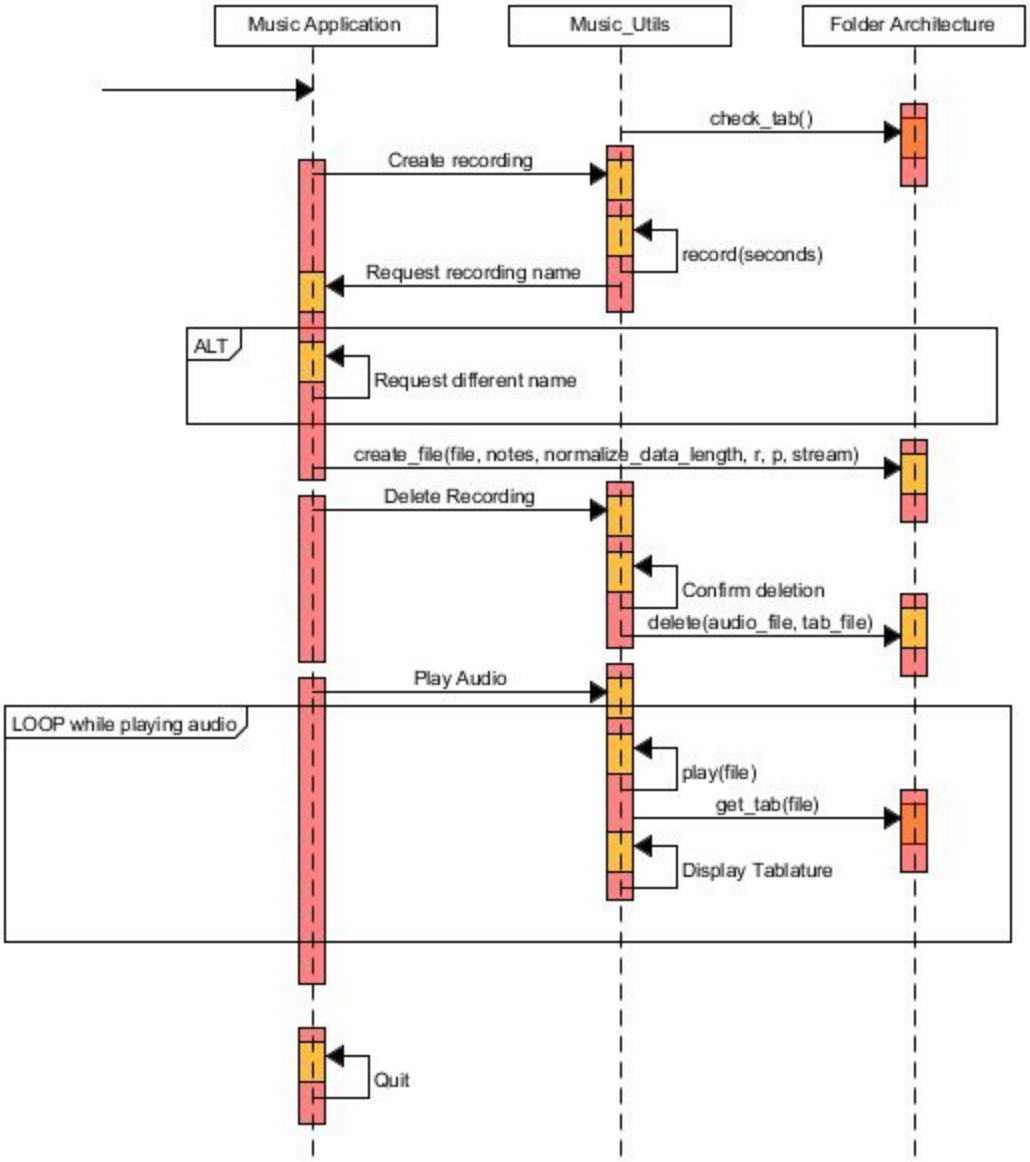
Domain Model



Class Diagram



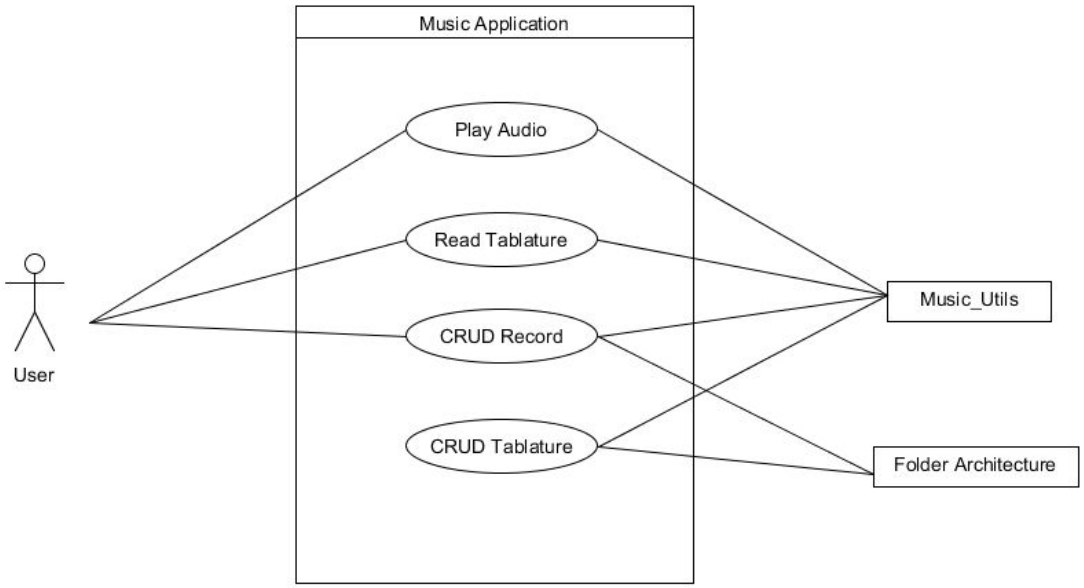
System Sequence Diagram



The ALT event happens when the user enters a name of a recording that already exists or it does not meet the naming requirements. The naming requirements would be if the text field is left empty the user will trigger this event, or if the name entered is 20 characters or more this event will trigger.

The LOOP event will continue until the full recording has been completed. After the user selects the audio they wish to play, the tablature is displayed of that audio. This will continue until the recording has completed playing. The tablature will remain visible until another recording is played.

Use Case Diagrams



Play Audio Use Cases

Actors

User

Description

This Use Case begins when the user wishes to listen to a recording they have created. The user selects the recording in the “Play” screen and the application plays the audio.

Main success scenario

1. The user goes to the “Play” screen.
2. The user selects a recording from the drop down menu.
3. The application begins to play the selected recording.
4. While the recording is playing, the user can not record, delete, or play another recording until the current one playing has completed.
5. After the recording has completed playing, the user can record, delete, or play another recording.

Read Tablature Use Cases

Actors

User

Description

This Use Case begins when the user wishes to read a tablature for a recording. The user selects the recording in the “Play”. When the recording begins to play the tablature for the selected recording is displayed.

Main success scenario

1. The user goes to the “Play” screen.
2. The user selects a recording from the drop down menu.
3. The application begins to play the selected recording.
4. The tablature for that recording is displayed at the top of the application.
5. While the recording is playing, the user can not record, delete, or play another recording until the current one playing has completed.
6. After the recording has completed playing, the user can record, delete, or play another recording.
7. The tablature remains displayed until another recording is selected.

Alternative

4.1 There is no existing tab for that recording.
4.2 The application notifies the user that there is no existing tablature for the selected recording.

CRUD Record Use Cases

Actors

User

Description

This Use Case begins when the user wishes to create a new recording. The user selects how long they want the recording and then clicks the “Record!” button on the “Home” or “Empty Home” screen. The application begins recording. The user is asked to enter a name for the recording.

Main success scenario

1. The user selects how long they want the recording from the drop down menu on the “Home” or “Empty Home” screen.
2. The user clicks on the “Record!” button.
3. The application begins to record the user for the selected amount of time.
4. While the application is recording, the user can not interact with the User Interface.
5. After the recording has completed, the user is asked to enter a name for the recording.

Alternative

5.1. The name entered by the user either already exists or does not follow the naming requirements.
5.2 . The application notifies and asks for another name to be entered.

CRUD Tablature Use Cases

Actors

None

Description

This Use Case begins when the user creates a new recording. After the user enters a name, the application creates a JSON file containing the tablature information.

Main success scenario

1. The user selects how long they want the recording from the drop down menu on the “Home” or “Empty Home” screen.
2. The user clicks on the “Record!” button.
3. The application begins to record the user for the selected amount of time.
4. While the application is recording, the user can not interact with the User Interface.
5. After the recording has completed, the user is asked to enter a name for the recording.
6. The application creates a new JSON file with the same name as the new recording.
7. The tablature information is saved in this new JSON file.

Internal Components

The internal components of this application can be found in the file Music_Utils.py which is in the application folder. It contains multiple functions involved for the transcription of audio and also to allow a user to record, save, delete and play their audio files. The functions used are:

delete(audio_file, tab_file)

This function deletes a selected audio file. The user selects which audio file they would like to delete (audio_file). After an audio file is selected for deletion, the JSON file containing the tablature data for the deleted audio file is also deleted (tab_file).

play(file)

This function allows a user to listen to an audio file they have saved. It plays back audio from a .WAV file. The user selects the file in their “Recordings” folder and the play(file) function plays it.

get_tab(file)

The get_tab(file) function finds the JSON file containing the tablature data of a desired audio file. It then pulls out all of the files data. This data is then used to create the tablature displayed for the user. This function returns the entire tab for each string on a guitar.

record(seconds)

This function allows a user to record a piece of music. Before recording begins, the user selects how long they would like the recording to be (in seconds). The application begins to record for the selected amount of time. After it is complete, the user is asked to name the file. After the naming of the file is complete, the recorded data is sent to the “create_file()” function. While this function is running (recording), the function is also transcribing the notes being played in real time. It uses the Fast Fourier Transformation (FFT) to transcribe the audio. The function returns “notes”, “normalize_data_length”, “r”, “p”, and “stream”.

create_file(file, notes, normalize_data_length, r, p, stream)

This function is used directly after a user creates a new recording with the “record()” function. It takes in all of the information about the new recording and saves the audio in the “Recordings” folder, and also saves the tablature JSON file in the “Tabs” folder. The function takes 6 parameters:

1. file = name of the new recording.
2. notes = tablature data (the notes that were playing in the recording).
3. normalize_data_lenght = The length of the “normalize” audio file in the static folder. The length of this file is needed to remove it from the finished audio file.

4. r = The audio data to be saved of the new recording.
5. p = This is the information required to create the .WAV file format. It stands for PyAudio, which is the module used in this application to create .WAV files.
6. stream = This is a requirement for PyAudio to work. It closes the recording stream.

check_tab()

This function is run every time the application is opened. Its purpose is to check if there is any JSON files in the “Tabs” folder that does not have a corresponding recording in the “Recordings” folder. If a JSON file is found with no corresponding recording, the JSON file is then deleted. This is to reduce the memory used in the application if a user deletes the audio files manually and never deletes the JSON files.

Data Design

Data Description

This application stores all of it’s data locally on the users machine. The data created by the application would be the audio files and the document files used to store the information of the tablature notation for each audio file.

Each of the recordings are saved in a folder named “Recordings” which is located in the folder the application is saved in. The audio files are saved as .WAV format which means they can be used outside of the application.

The tablature information is saved in JSON files in a folder named “Tabs” which is also located in the folder the application is in. A JSON file is created each time a new recording is saved. The files are named the same as the recording that they hold the tablature data for. When a recording is deleted in the application, the corresponding JSON is deleted too. If a user deletes the audio file manually from the “Recordings” folder, the next time the application is started up the JSON file for the deleted audio file will be deleted.

Tablature JSON Files

The JSON files contain the data for transcribing the audio in the audio files. The data saved is a list of all of the notes played in the audio (including silences) and the time stamp the note was played. The JSON files can be read outside of the application if desired. The application uses the information saved in the JSON file to construct the tablature notation in real time at the user's request.

Required Files

As all of the data created is stored locally on the users machine, the user must have space free on their hard drive to create new recordings, otherwise the files will not be able to save. The application comes with files that are required for it to run. In the “Static” folder, a .WAV file named “normalize” is saved. The file is used to normalize the audio recorded by the user in the application. It is a requirement for the application to run. The applications folder also contains two python files, “main.py”, “music_utils.py”. The main.py file is the main application file that contains all of the UI and is also the file that runs the application. The music_utils.py file contains all of the functions and algorithms used in the application. All of these files are required for the application to work correctly.

Component Design

Different Components

User Interface

The design of the User Interface (UI) should be as simple as possible. This is because the application was designed to make life easier for a user. The less button clicks a user will have to do to achieve the task they want to do the better. The UI should be clearly readable and easy to navigate even for first time users.

Music_Utils

The Music_Utils file contains all of the functions and algorithms used in recording and transcribing the audio files.

Folder Architecture

As the application stores all of the files created locally on the users machine, each element of the application has its own designated folder. The different folders are:

Application Folder

This is the main folder the application is kept. It contains everything the application requires to run including all of the folders used for storing the files created. This is also where the UI code, Music_Utils code, the license of the application and a README. The README describes the application.

Recordings

In this folder, all of the audio recordings created by a user using the application is stored. The audio files are saved as .WAV format.

Static

In this folder, an audio file named “normalize.wav” is stored. This file is required for the application as it is used to normalize the sound when recording.

Tabs

In this folder, all of the audio recordings tablature data is stored. They are stored in JSON format.

User Interface Design

Description

The design of the User Interface (UI) should be as simple as possible. This is because the application was designed to make life easier for a user. To follow the goal of simplicity, the UI was designed with a mentality of the less button clicks a user will have to do to achieve the task they want to do the better. The UI should be clearly readable and easy to navigate even for first time users. The selling point of this UI is how the tablature is displayed. It should be easily readable for the user and represent their piece of music correctly in tablature notation. Each button is labeled appropriately with exactly what they do to reduce ambiguity. For example, to record a piece of music the user would click the button labeled “Record”. By labeling everything in the UI appropriately, it will be easy to navigate.

Screen Description

There are 3 screens in this application. There is the “Home” screen, the “Empty Home” screen, and the “Play” screen. The “Home” and “Empty Home” screens are where a user can make a new recording. These are the first screens the user is greeted with upon starting up the application. The difference between them is that the “Empty Home” screen is displayed when the user has no existing recordings on their machine. The difference is that it does not let the user go to the “Play” screen. When a user creates a recording, populating the “Recordings” folder, the “Home” screen is then displayed to them, allowing them to go to the “Play” screen to see their recordings.

Home / Empty Home Screen

The “Home” and “Empty Home” screens are the same with the exception of the button allowing the user go to the “Play” screen. They both have a button labeled “Record!”, which allows the user to create a new recording. They also have a button labeled “Quit!”, which closes the application. There is a drop down menu to the left of the “Record!” button. It contains the numbers 5, 10, 15, 20, 25, and 30. This represents the number of seconds the application records for when the “Record!” button is clicked.

Play Screen

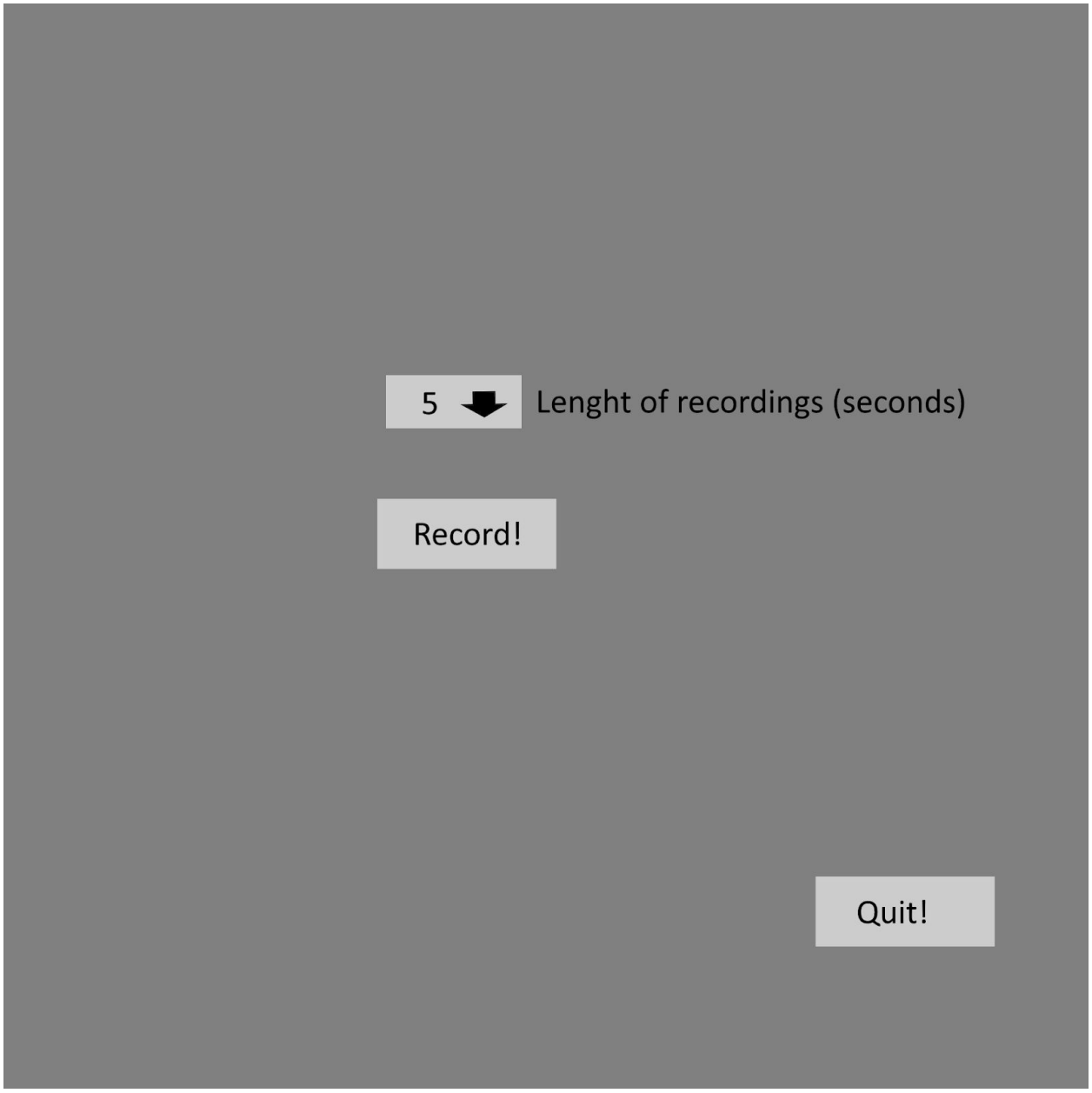
Each of the recordings saved on the users local machine will be displayed in a drop down menu on the “Play” screen. There is a drop down menu for playing the audio file and a separate drop down menu for deleting an audio file. The user clicks on the recording they wish to either play or delete from the drop down menus. When a user clicks on a recording to play, the tablature for that recording will be displayed to the user at the top of the “Play” screen. This is so the user can listen to the audio and see how it was played. Each time a new recording is added or deleted, the two drop down menus are updated. Each of the drop down menus are labeled “Play” and “Delete”.

Pop-Up Messages

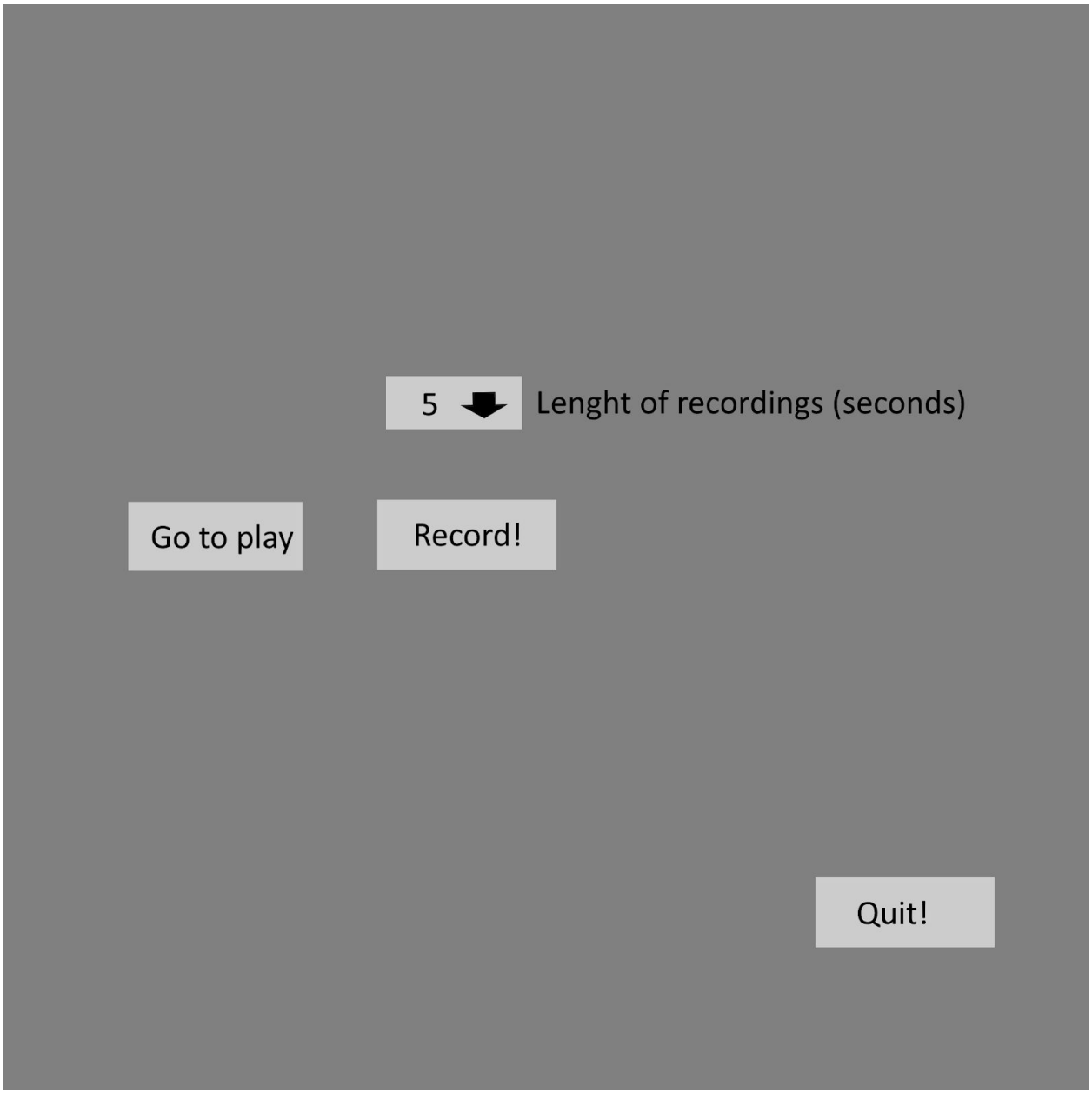
When a user wishes to delete a recording, or click on the “Quit” button to close the application, they are confronted with a pop up message asking them if they are sure they want to go through with this action. The user is also confronted with a pop up message when they record a new recording. After the recording has finished, the user is asked to enter in the name of the recording. If the recording already exists, or the naming requirements have not been met (e.g. left empty or too long), a new pop up message will appear and the user will be asked to pick a different name.

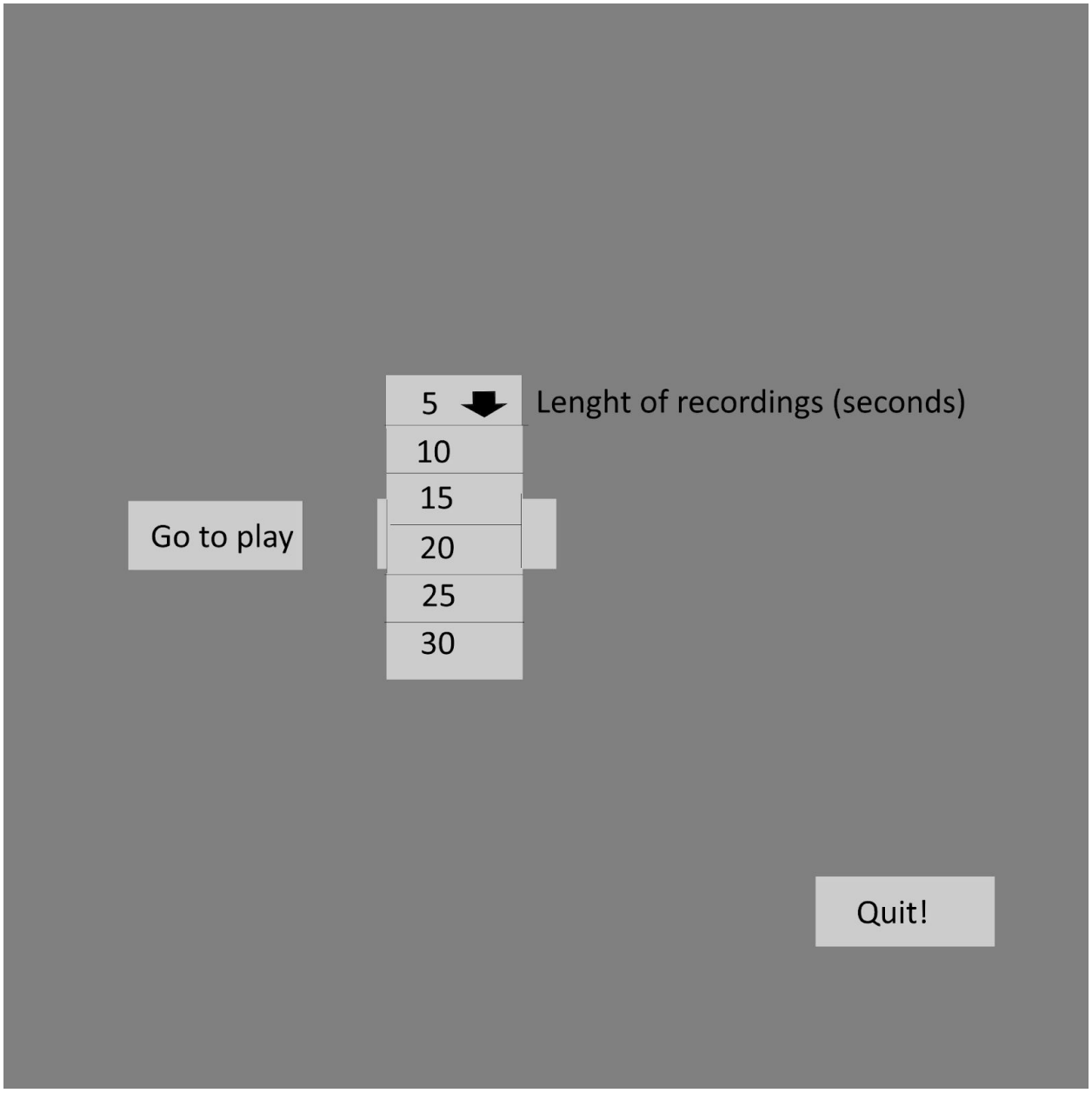
Screens

Empty Home Screen

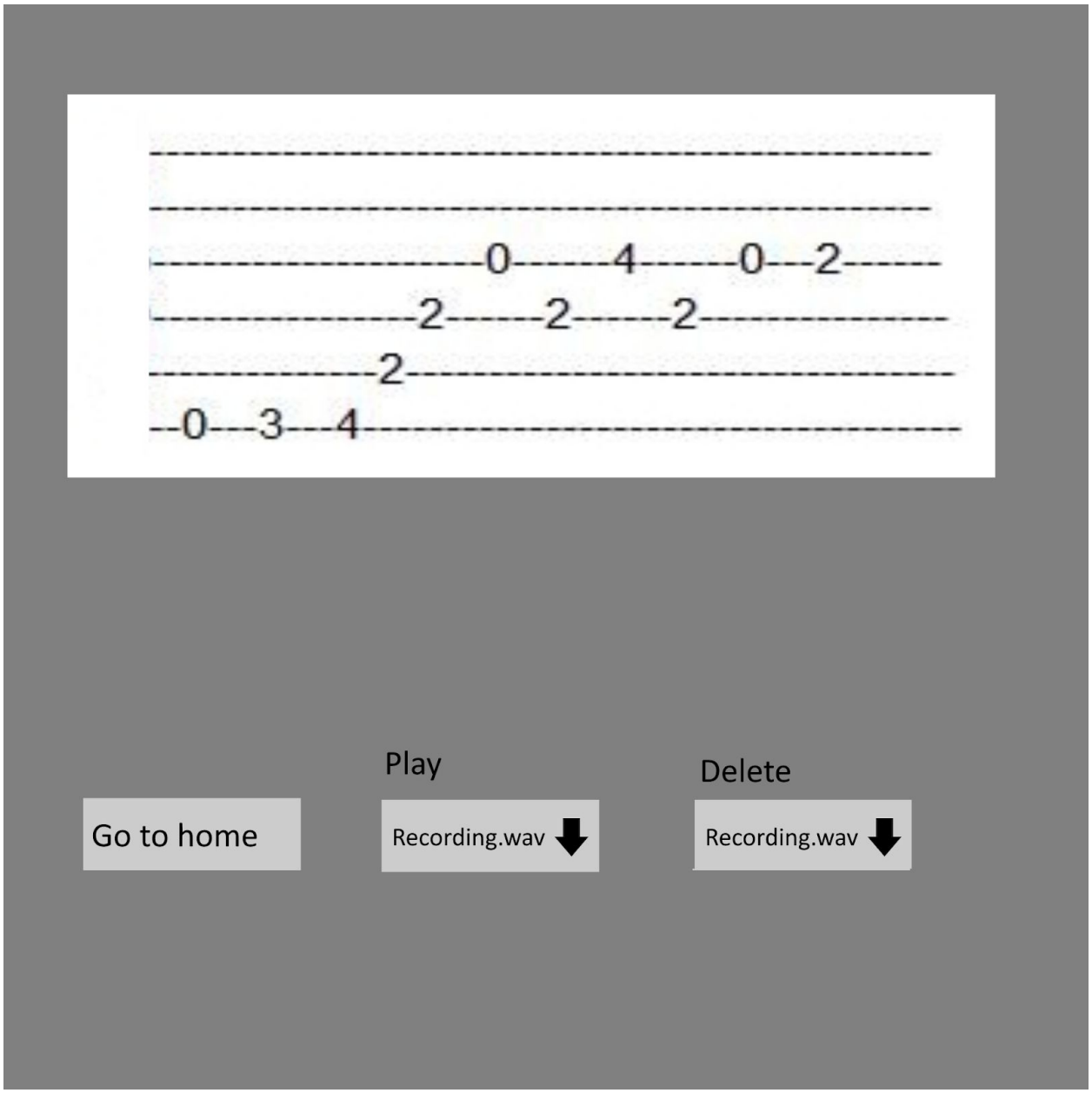


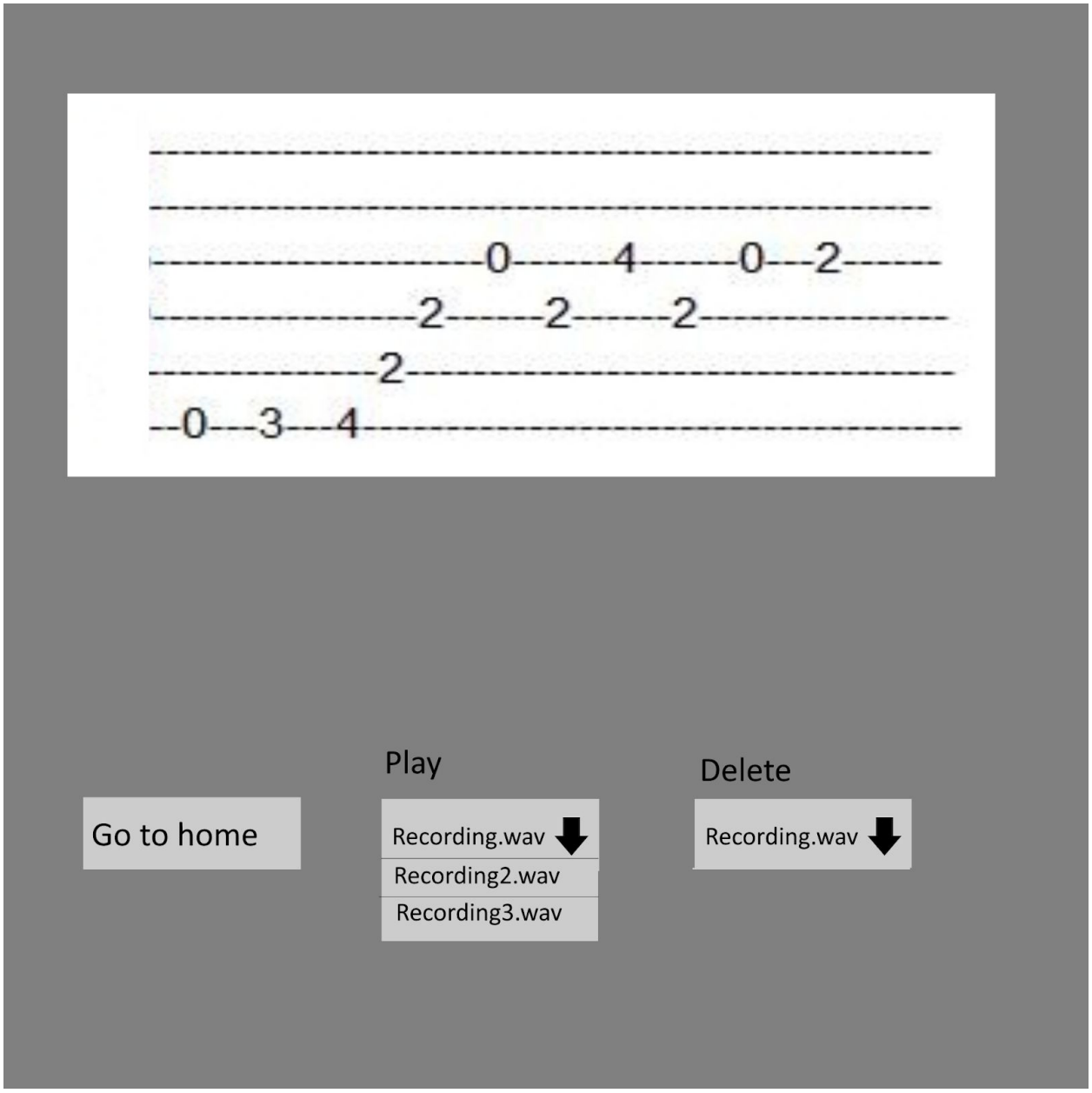
Home Screen




5
Lenght of recordings (seconds)
10
15
20
25
30
Go to play
Quit!

Play Screen


0 4 0 2
2 2 2
2
0 3 4
Play
Delete
Go to home
Recording.wav
Recording.wav



Screen design

As simplicity is one of the main goals for this application, the screens are all as basic as can be. This is to follow the theme of most music applications on the market. Since this application is similar to a guitar tuner, the design of the primary colour of grey was chosen. This is because a lot of existing tuners have a simple UI like this with plain colour like grey.

There is not a lot of buttons on each screen as the design follows the statement “The less buttons clicks a user will have to do to achieve the task they want to do the better.” Each button/drop down menu is clearly labeled and the Tablature in the “Play” screen is easily readable.