CSY3010 19/20 Assignment (DRAFT updated on 20.01.2020)

Date for submission:	03/05/2020	Module Tutor: Signed:	Mu Mu
Student Name: Student ID:	Diwas Lamsal 18406547		

Assessment Feedback

Aspect (& weighting)	Excellent	Very Good	Satisfactory	Needs some more work	Needs much more work
Design and implementation of the user interface (20%)					
Design and coding of the functionalities (40%)					
Quality of code in the application to Media Technology (10%)					
Technical report (30%)					

By entering your **name**(s) and **student ID**(s) you are asserting that this submission is entirely your own individual.

--

Demo video link:

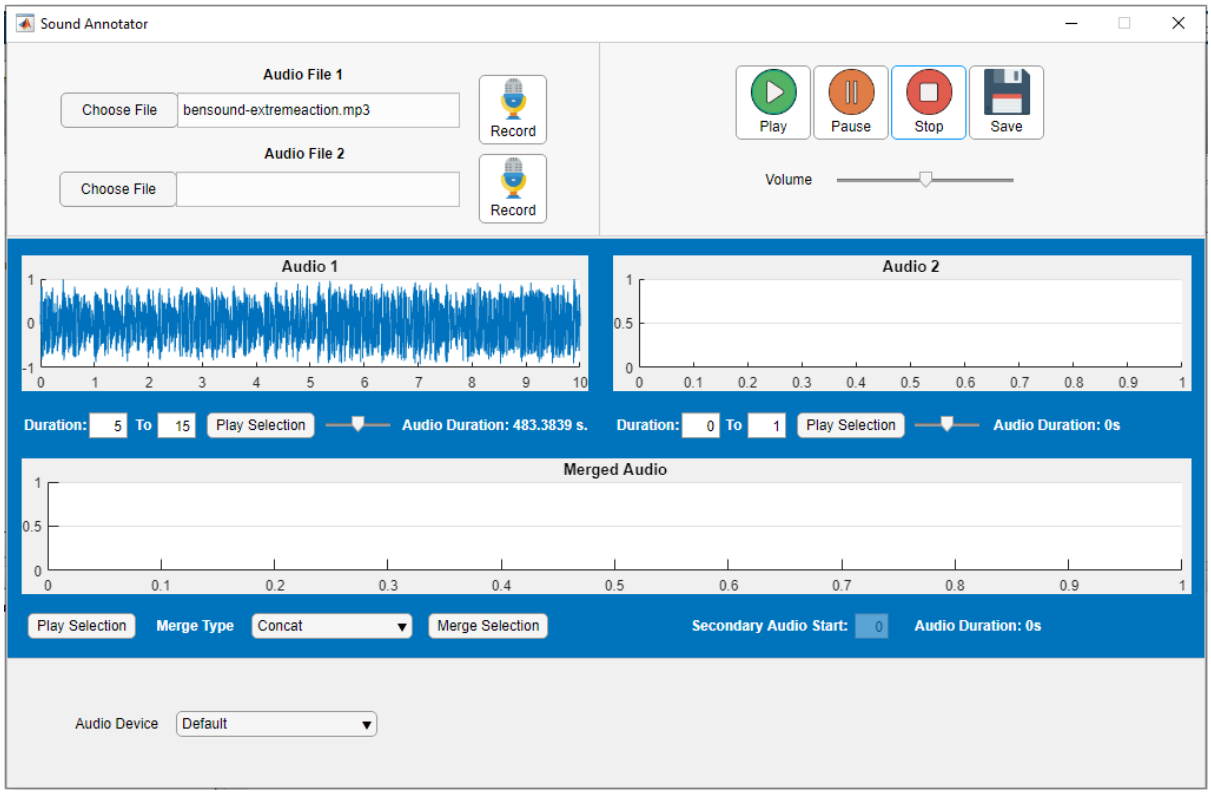
https://northampton.mediaspace.kaltura.com/media/18406547-diwas-lamsal-media-technology-assignment/1_7wa1jw2r

OR

https://youtu.be/vOADGdtlsLk

Content

LIST OF FIGURES


Sound Annotator
Audio File 1
Choose File
bensound-extremeaction.mp3
Record
Audio File 2
Choose File
Record
Play
Pause
Stop
Save
Volume
Audio 1
Audio 2
Duration: 5 To 15 Play Selection
Audio Duration: 483.3839 s.
Duration: 0 To 1 Play Selection
Audio Duration: 0s
Merged Audio
Play Selection
Merge Type
Concat
Merge Selection
Secondary Audio Start: 0
Audio Duration: 0s
Audio Device
Default

1. INTRODUCTION

Media technology involves representing, manipulating and processing media such as image, audio and video in digital form. Learn.org says media technology "applies interactive computer elements, such as graphics, text, video, sound, and animation, to deliver a message." (Learn.org, 2020). It is applied across many domains such as voice recognition, photo libraries, media streaming, etc. One of such domains is sound processing. Two types of sound are stored in computers – MIDI, which can be manipulated by bringing changes to characteristics such as pitch, duration, loudness and notes; and digital sound, which is the digitized form of real-world analog sound waves. This involves all the sound we record, store and transmit (Webopedia, 2020).

This paper discusses an approach to use sound processing techniques in MATLAB to manipulate audio. MATLAB allows mathematical manipulation of medias using a high-level programming language. It can be used very effectively for audio manipulation with relatively less code than other programming languages. As part of the project, a GUI application will be produced using MATLAB App Designer. The main intention of the application will be to allow merging of audios (example: adding voice commentary to existing football match audio). The project will be built completely from scratch and references from documentation and online forums will be taken wherever necessary. The upcoming sections will include review of relevant literature to the topic, comparable systems, system analysis and design documentation and any relevant test strategies as well as their results.

2. RESEARCH & LITERATURE REVIEW

2.1 Literature Review

2.1.1 Sound Processing With MATLAB

MATLAB has gained a lot of popularity because it is very easy to use and understand; more so if there is experience of other programming languages. Some books have discussed about the use of MATLAB in different sound manipulation methods such as The Fourier Transform, Discrete Wavelet Transform and Continuous Wavelet Transform among many others (Weeks, 2011). Some authors mention that it has gained this high level of popularity in the science and engineering computing fields due to a very sharp learning curve and is being used throughout

the world for courses like electronics (Andreatos et. al., 2009). An article discusses the use of MATLAB, which has included Arduino packages, together with Arduino platforms for sound capture, processing and reproduction (Silva et. Al., 2015). The authors discuss many examples in detail and about any limitations of the approach. We can see that MATLAB has many examples of use for sound processing.

2.1.2 MATLAB Applications

Similarly, a lot of applications have been built using MATLAB GUIDE and App Designer. While App Designer is fairly new and was kind of limited until a while back, it has started to gain some popularity. Authors argue it provides an "enhanced design environment" but was considered limited as it had "limited graphics support" in the earlier versions where they have articulated an approach to build a vehicle monitoring system using MATLAB and App Designer (Valle et. Al., 2017). The article mentioned earlier about popularity of MATLAB in Science and Engineering discusses about an application built using GUIDE that can be used for teaching Automated Control Systems (Andreatos et. al., 2009). A large number of GUI applications have been shared in the MATLAB File Exchange which fall under many different categories (File Exchange – Mathworks.com, 2020). It can be taken from the availability of such amount of resources that MATLAB has an active community and a lot of users have used it for designing broad range of applications.

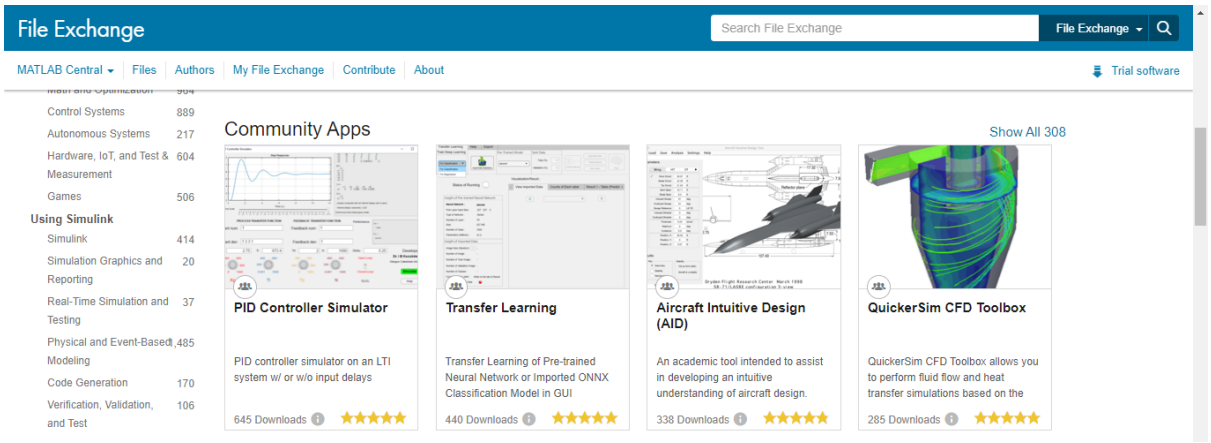


Figure 1 – MATLAB File Exchange

2.1.3 App Designer vs GUIDE

The MATLAB MathWorks website has provided a post about comparison of App Designer and GUIDE. This could be analyzed before implementing any one of these in order to build our system. They recommend App Designer over GUIDE and mention that they will stop supporting GUIDE which will be removed in an upcoming version. Some key differences between these two can be tabulated (Comparing GUIDE and App Designer - Mathworks.com, 2020).

App Designer	GUIDE
The app designer applications contain a single ".mlapp" file.	The GUIDE applications contain two files both of which need to be edited.
The app designer has advantages over GUIDE such as modern looks, embedded code editor and extra UI elements such as spinners, switches, etc.	GUIDE has some features that is limited to GUIDE only such as app templates and printing graphics support.
The object-oriented properties are followed very well in the App Designer and getters and setters are used. There is availability of private and public functions and use of objects like in any other Object-Oriented Programming Language.	Use of dot notation to change variable values.
Only code such as user-defined properties, functions and callbacks are editable whereas the code that manages the interface and other code is not editable by the developer.	All available code can be edited by the developer.
The use of OOP properties makes combining two or more applications simple.	Relatively harder to combine two or more GUIs.

2.2 Comparable Systems

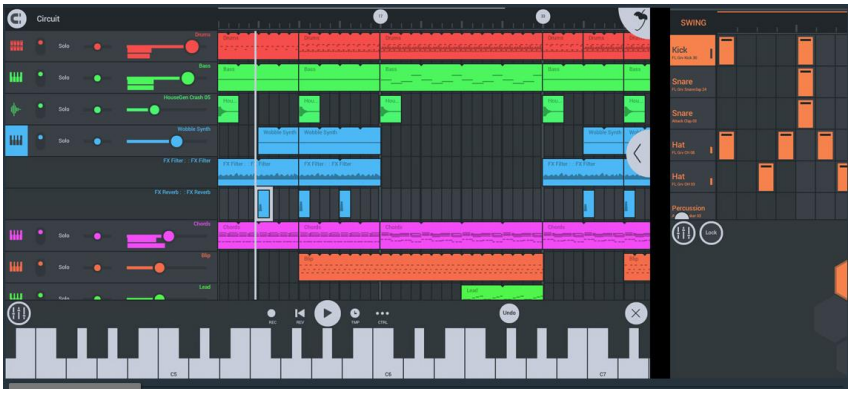
System	Description
MAGIX Music Maker www.magix.com	MAGIX is a company based in Germany and provides software for video, audio and photo creation/manipulation. The one closely related to the domain of our application is the MAGIX Music Maker. It allows wide range of features from making music, recording music to mixing and mastering music and is used by a lot of professionals in the music industry. (MAGIX, 2020)  The software contains a wide range of features and allows production of industry-ready music. The UI looks clean despite having a variety of features. There is also a free version which itself provides features such as inbuilt sounds ready for use. The task of mixing sounds is very simple in this application as it allows combination of a lot of instruments (drums, guitar, etc.). Sound waves can be manipulated for changing other aspects of the sound such as adding fading effect, changing volume of certain portions, etc. The task as part of this project could easily be solved by the use of this application.

FL Studio

www.image-line.com

FL Studio is another audio workstation from a company in Belgium, Image-Line. It is a complete music production environment which could be used similarly like the MAGIX Music Maker. The software claims to provide interface to “compose, arrange, record, edit, mix and master” music. Like MAGIX, many power users in the music industry use this application. (FL Studio, 2020)

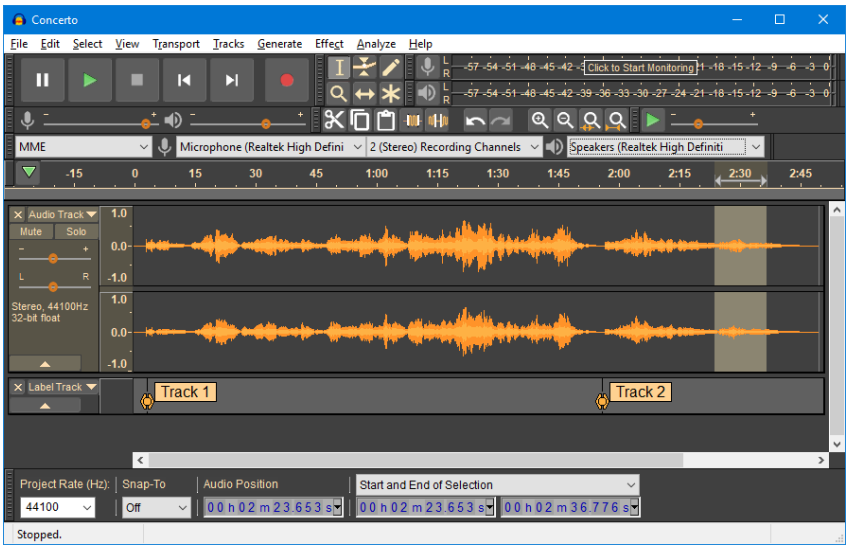
One key disadvantage over the MAGIX software could arguably be the lack of a free package. Otherwise, it contains similar or even more features than the MAGIX Music Maker, and this software too would solve the task for this project easily as unlimited amount of sounds can be merged together. Moreover, the software is also available for mobile devices with no compromise in usability.



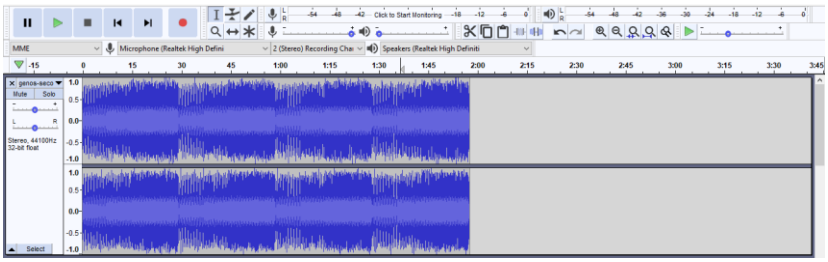
Audacity

www.audacityteam.org

Audacity is an open-source audio software for Windows, Mac OS, Linux and other operating systems which was initially released on May 28th, 2000. It can be used for recording and editing audio in computers. Unlike the previous applications, this one is fully free of cost. (Audacity, 2020)



It allows separate manipulation of both channels of audio. It could be argued that the UI is quite intended for advanced users as by default, the user has to manipulate the sound waves whereas in the other reviewed software, only technical users have to access the sound waves directly. This software is closest among the three to the application that will be produced as part of this project and can be referenced to for different sets of features (for example: instantaneous recording and manipulating audio).



3. DESIGN SPECIFICATIONS

This section will include logical as well as design aspects of the proposed application. The logical aspects will be presented in the form of flowcharts and use-cases whereas the design aspects will be presented in the form of wireframes and system event diagrams.

3.1 System Architecture

3.1.1 System Flowcharts

1. Loading and Playing Audio

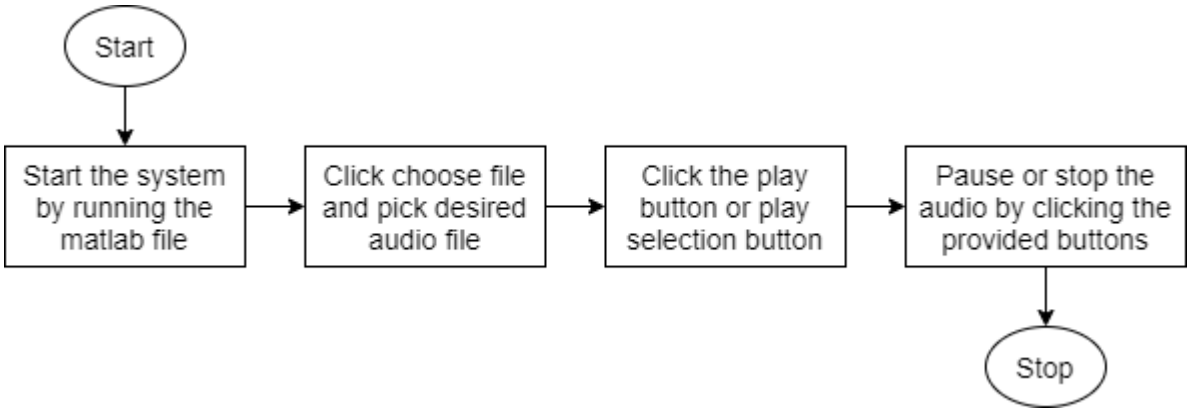


Figure 2 – Loading and Playing Audio Flowchart

2. Recording and Playing Audio

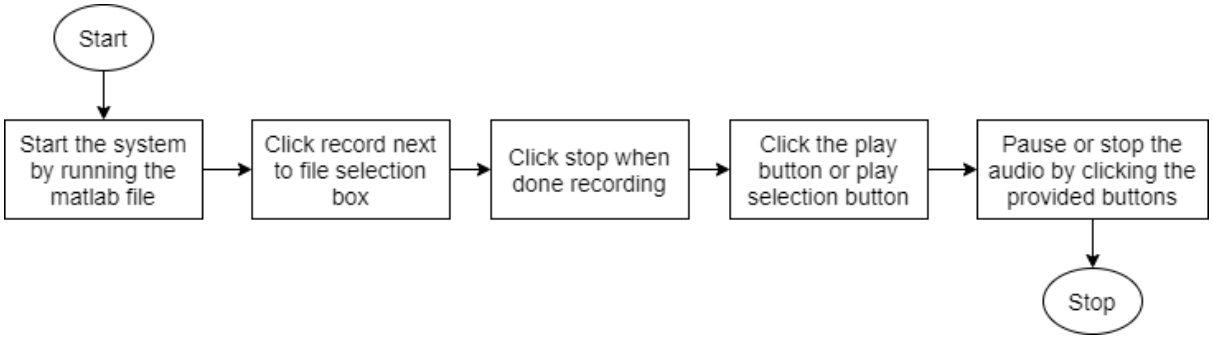


Figure 3 – Recording and Playing Audio Flowchart

3. Merging two audios (Concatenate/Truncate)

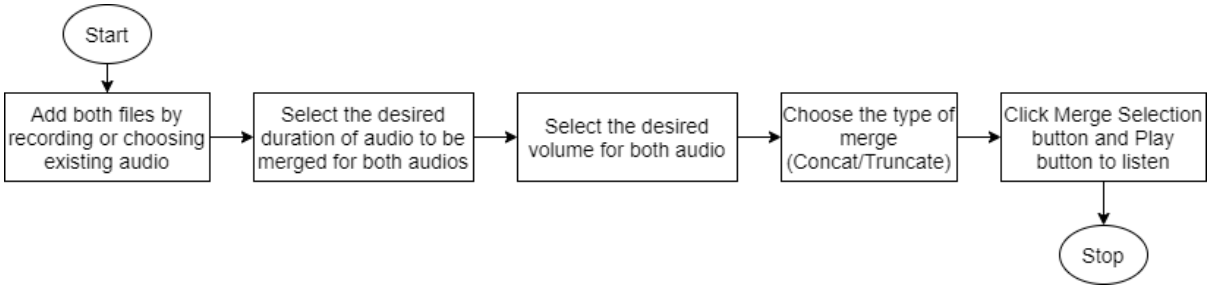


Figure 4 – Merging audio flowchart 1

4. Merging two audios (With Primary Audio)

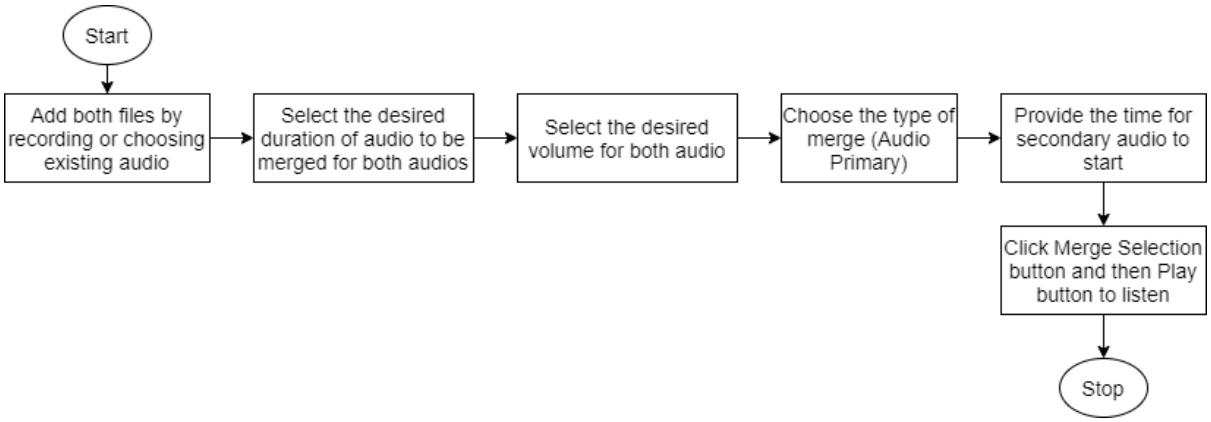


Figure 5 – Merging audio flowchart 2

5. Saving merged audio

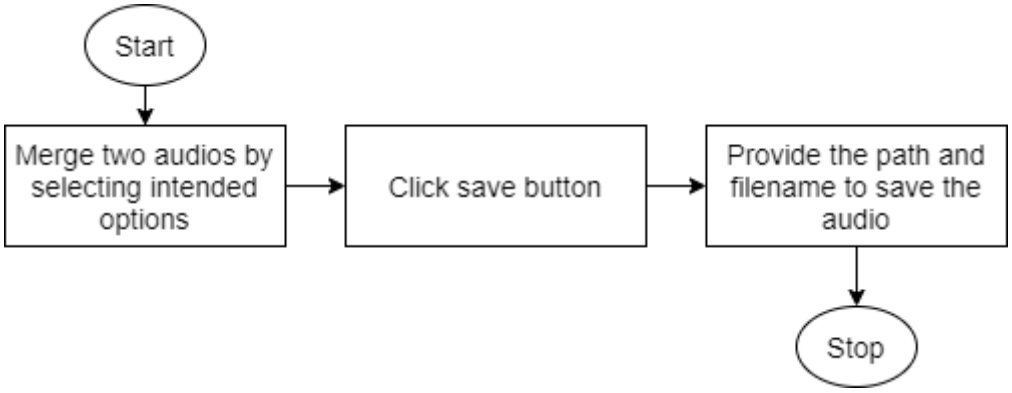


Figure 6 – Saving audio

3.1.2 Use Case Diagram

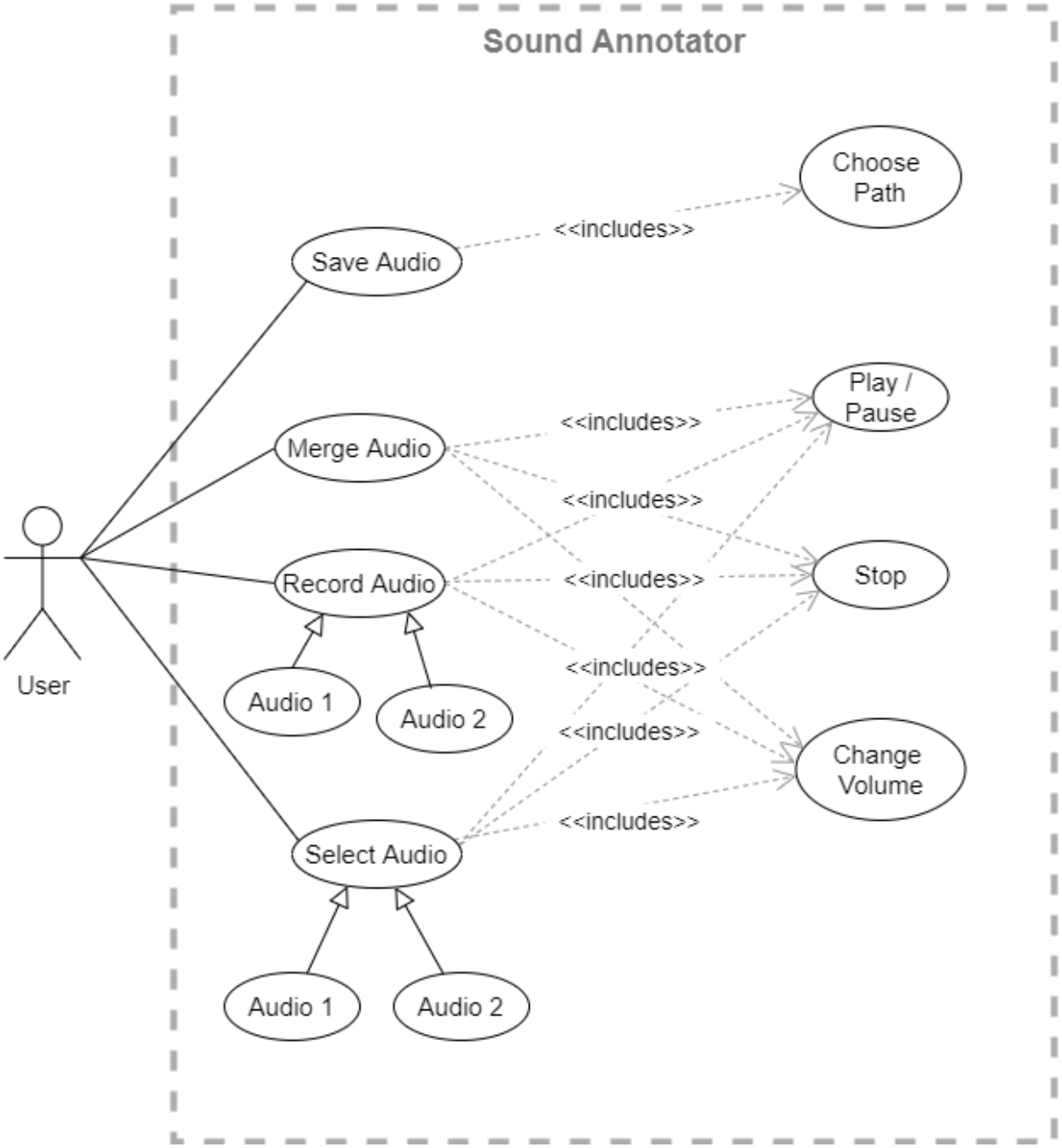


Figure 7 – Use case diagram

3.1.3 Documentation of Use Case

i. Select Audio

Name: UC1 Select Audio

Initiator: User

Goal: Select desired audio into the application

Pre-condition: The application is open

Post-condition: The desired audio gets selected

Assumptions: The audio is valid.

Main Success Scenario:

1. Users clicks Choose File button for Audio 1 or Audio 2.
2. User navigates to the desired audio file.
3. User clicks open.
4. The audio is chosen, displayed in relevant figure and can be played.

Includes: Play, Stop, Change Volume

ii. Record Audio

Name: UC2 Record Audio

Initiator: User

Goal: Record audio and use it in the application

Pre-condition: The application is open

Post-condition: The audio gets recorded and selected in the application

Assumptions: The recording device is connected and functional.

Main Success Scenario:

1. User clicks the record button.
2. User speaks or plays relevant sound in the background for recording.
3. User clicks the stop button at the end of recording.
4. The recording is saved, plotted in relevant figure and can be played.

Includes: Play, Stop, Change Volume

iii. Merge Audio

Name: UC3 Merge Audio

Initiator: User

Goal: User merges two audios

Pre-condition: Two audios are selected or recorded

Post-condition: Audios are merged

Assumptions: *None*

Main Success Scenario:

1. User selects/records both the audios and their volumes.
2. User provides the duration of audio to be merged for both audios.
3. User selects the desired type of merge.
4. User selects desired volume for the final audio.
5. User clicks the Merge button.
6. The audio is merged, displayed in relevant figure and can be played.

Includes: Play, Stop, Change Volume

iv. Save Audio

Name: UC4 Save Audio

Initiator: User

Goal: User saves merged audio

Pre-condition: Audios are merged.

Post-condition: The merged audio is saved in the computer and can be played with local media player.

Assumptions: *None*

Main Success Scenario:

1. User merges the audio and is satisfied with the final results.
2. User clicks save button.
3. User selects the desired path and clicks save.
4. The saved audio can be accessed through the explorer and played with local media player.

Includes: Choose Path

3.2 Wireframes

Screen Size

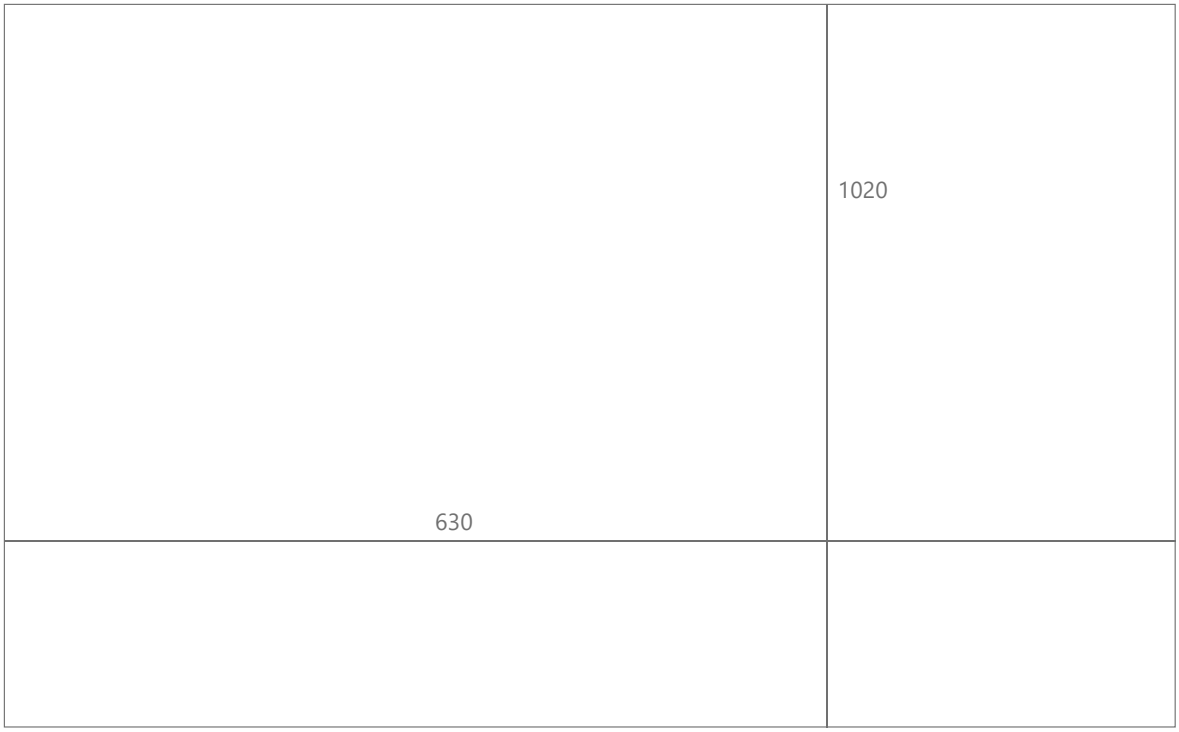


Figure 8 – Screen size wireframe

Screen UI

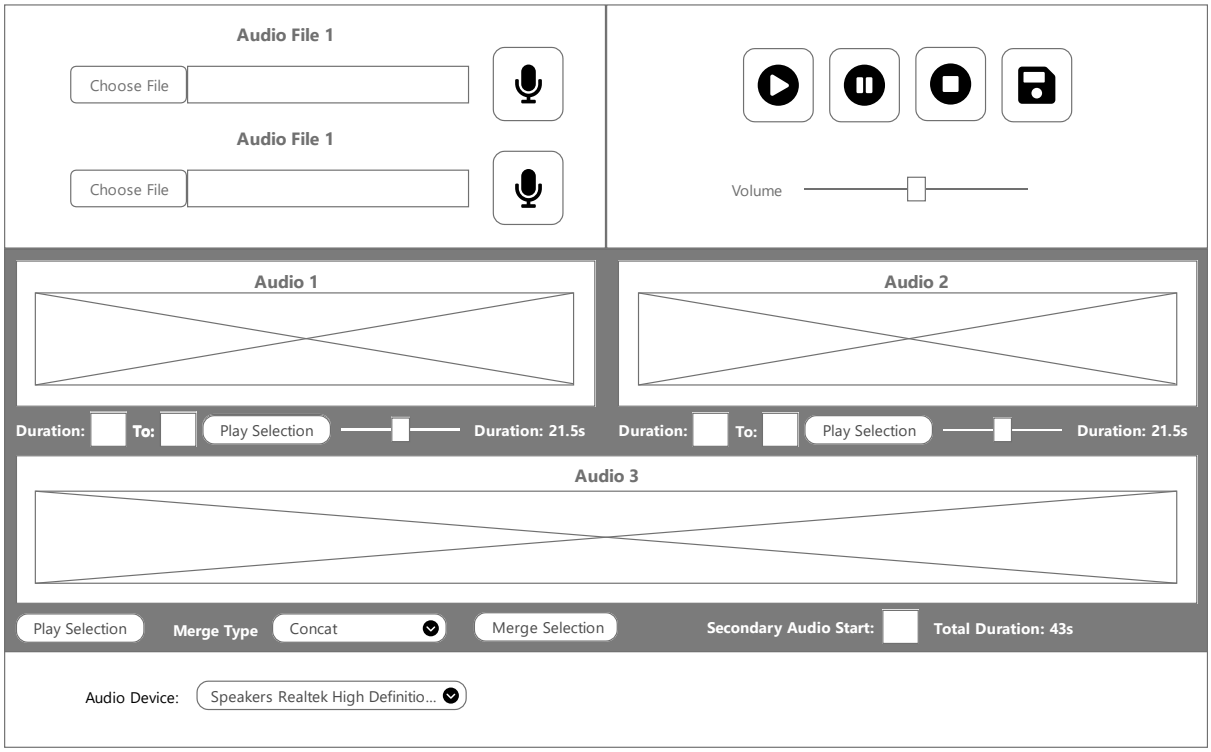


Figure 9 – Screen UI wireframe

Choose / Save

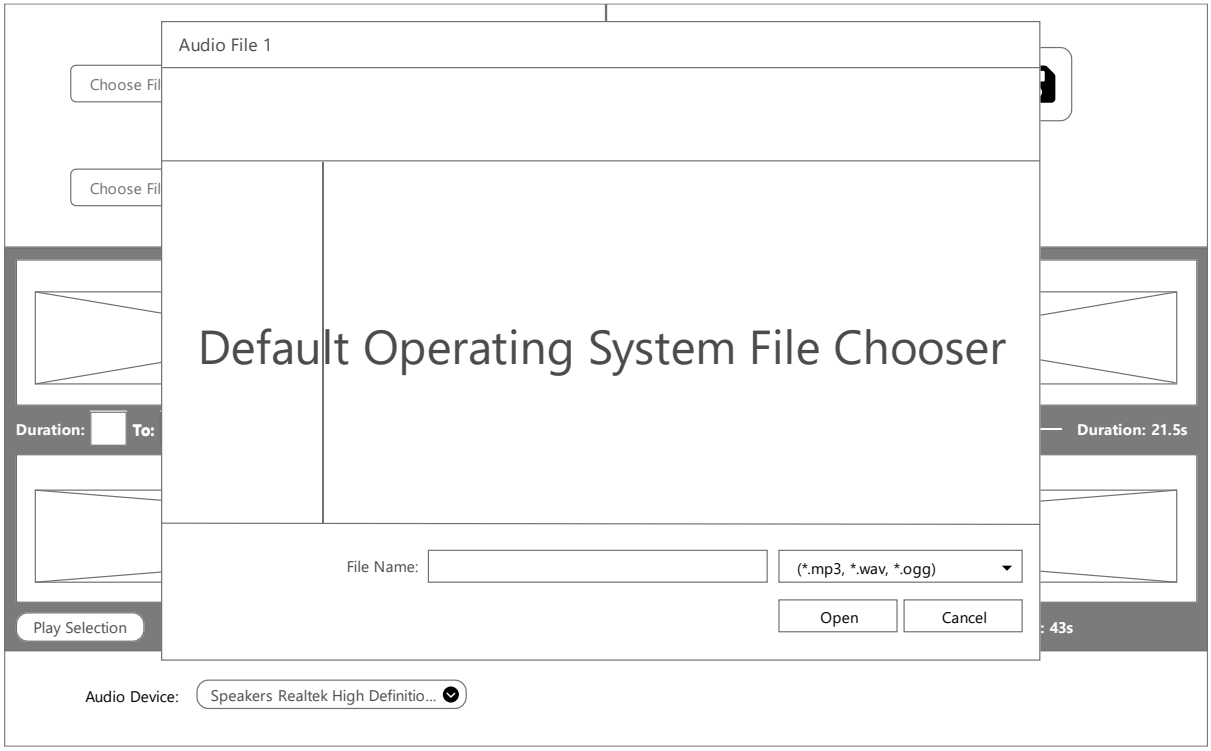


Figure 10 – Choosing/Saving audio wireframe

Final System Interface

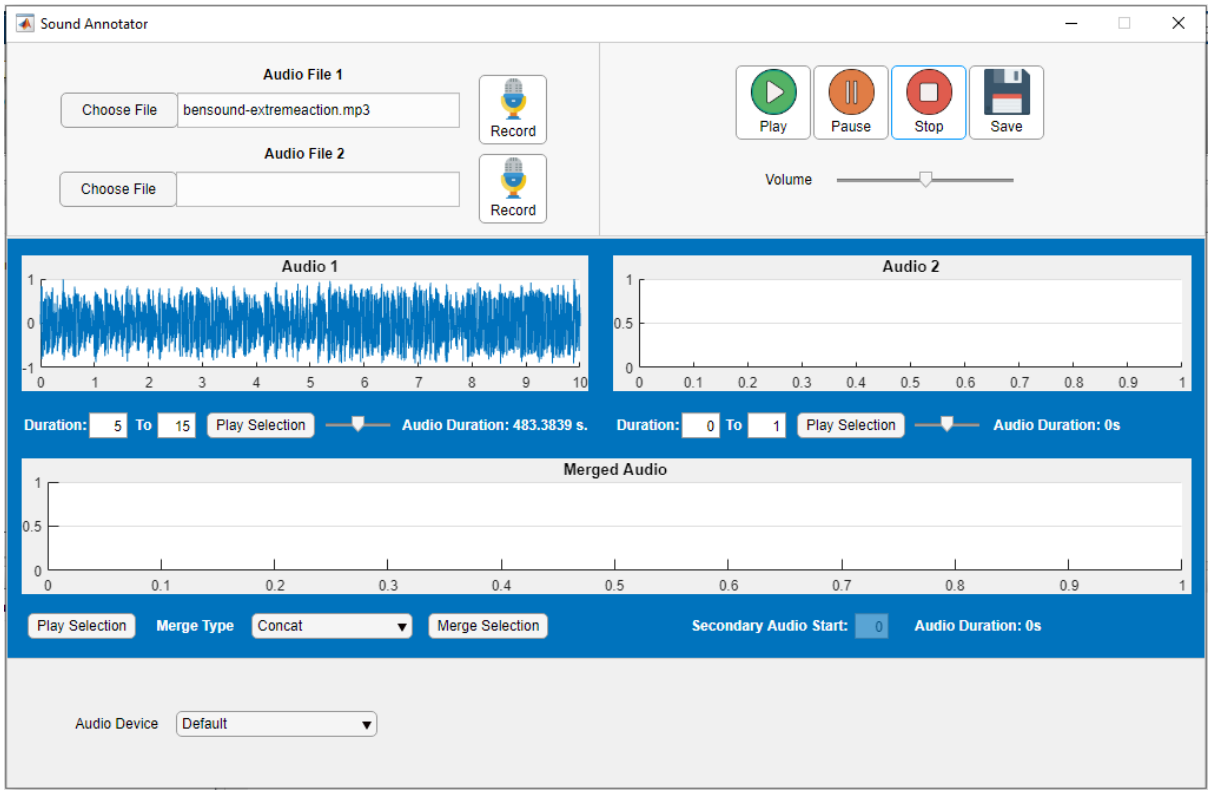


Figure 11 – Final system interface

3.3 System Event Diagrams (Dynamic Modelling)

Choose Audio File

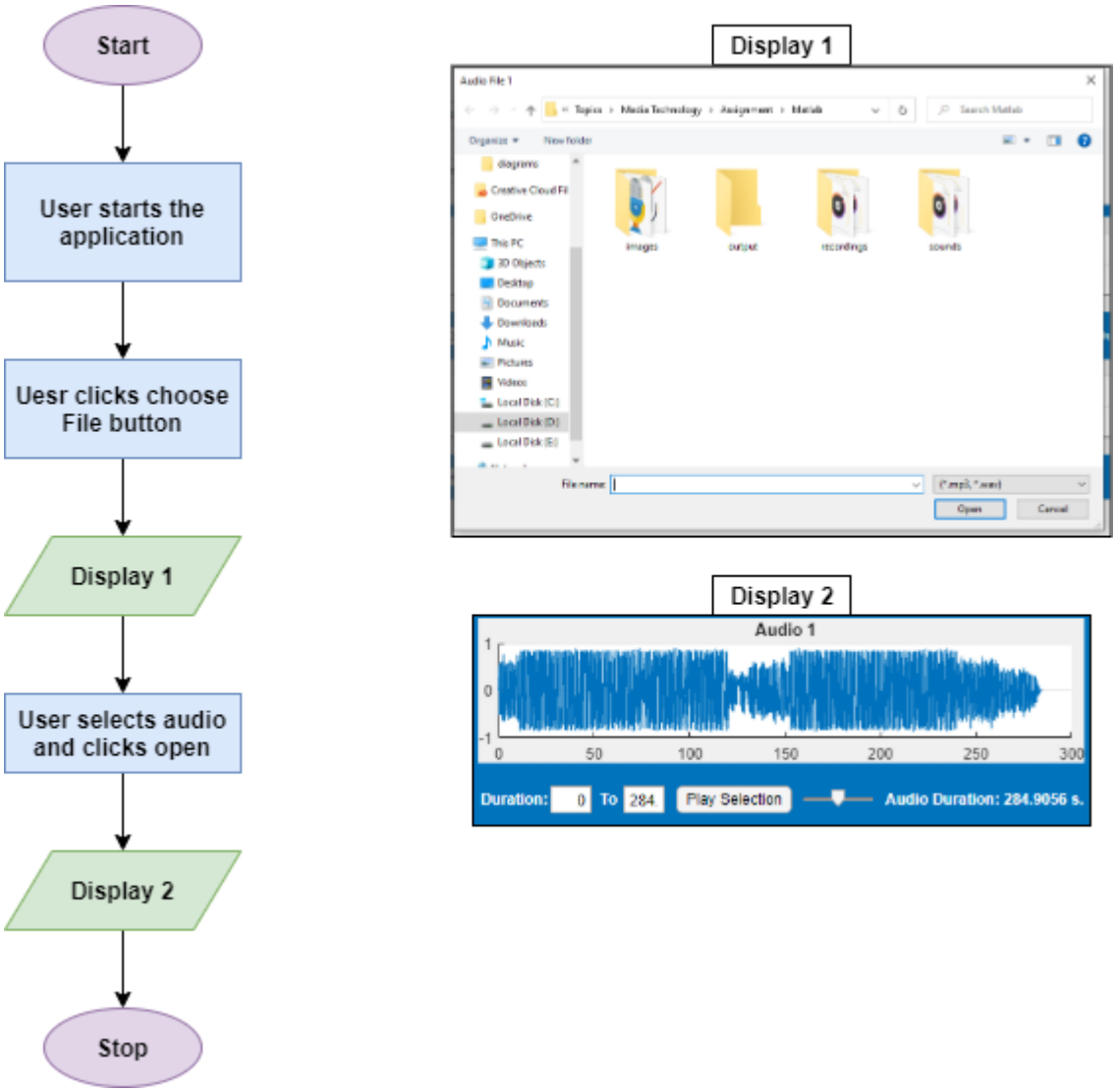


Figure 12 – Adding audio event diagram

Record Audio

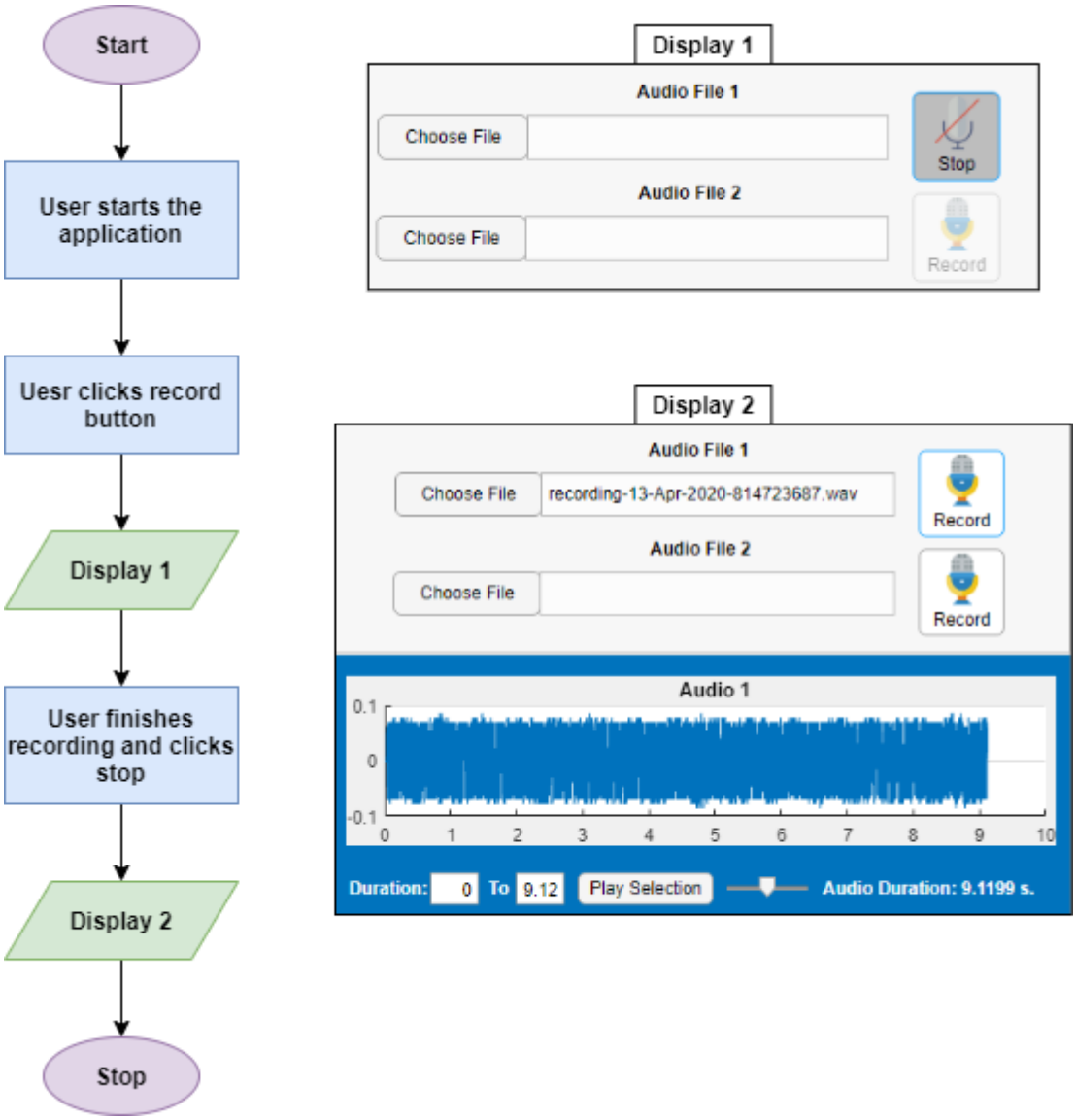


Figure 13 – Recording audio event diagram

Merge Audio & Save

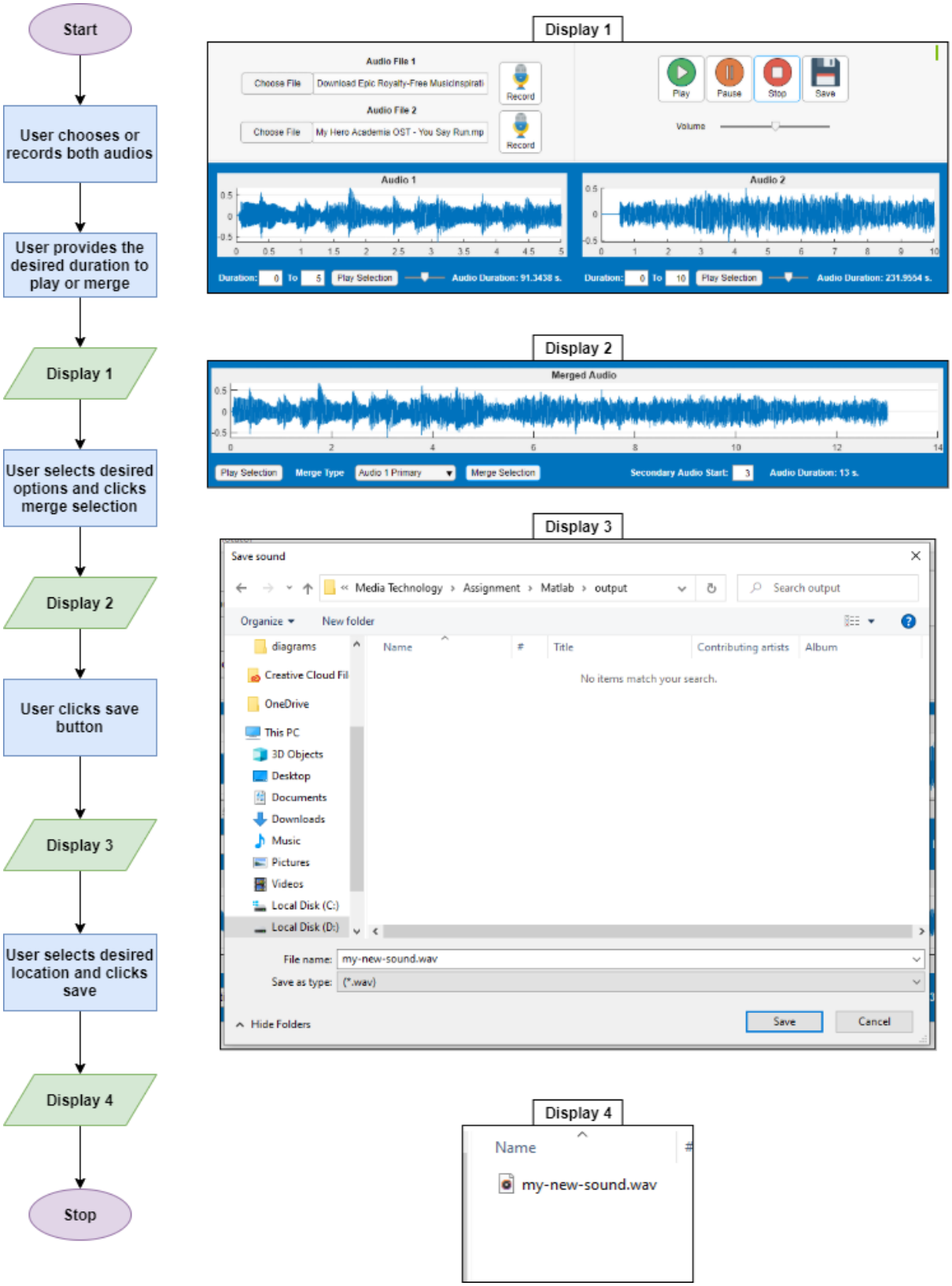


Figure 14 – Merging and saving audio event diagram

4. EVALUATION - KEY FEATURES & WEAKNESSES

This section will discuss the minimum as well as extra features implemented in the application. The focus will also be put on explanations of how these aspects were implemented and are related to media technology.

As part of the minimum specification, the following features have been implemented:

1. Importing two audio files

In order to import audio files into the interface, a small search for file picker in MATLAB documentation was enough. After some research, it was found that MATLAB provides "uigetfile" command to open a file picker dialog. Clicking the choose file button in the interface brings up this dialog and allows selection of relevant audio files. (uigetfile - Mathworks.com, 2020)

```
% Button pushed function: FileSelector1
function FileSelector1ButtonPushed(app, event)
    % https://www.mathworks.com/help/matlab/ref/uigetfile.html
    [filename filepath] = uigetfile({'*.mp3; *.wav'},'Audio File 1');
    %https://uk.mathworks.com/matlabcentral/answers/296305-appdesigner
    figure(app.UIFigure) %to bring back focus to the app
    if(isa(filename,'double')==1)
        return;
    end
    app.audio1 = strcat(filepath, filename);
    app.Audio1namedisplay.Value=filename;
    setupAudio1(app);
end
```

Figure 15 – Importing audio source code

There was an annoying event occurring after picking files by using the provided command. It made the application minimize each time a file gets picked. It was learned that this has happened to many users and a lot of solutions have been provided. One of the answers mentioned about manually putting the focus back on the application by using the figure() command (Friedrich, 2019). This solved the issue.

Likewise, the isa() command is used to check whether the file is picked or cancelled. If the file is picked, filename should be a string value and if it does not get picked, it needs to be a double. This was confirmed by displaying the values in the command window earlier. If the file does not get picked, the process should halt and not attempt to import any audio.

2. **Mixing one audio to a specific part of the second audio so that both audios can be heard adjacently**

This functionality has been implemented with the aid of provided lecture slides and some research across the web. Initially, as minimum specification, this was done by allowing selection of portions of both the audios to merge. The longer one would get truncated and the two audio files could be listened together. The min() function was used to find the length of smaller audio and then the audios are merged by using '+'. This was however improvised to provide more options for merging the audios and will be discussed as an advanced feature.

```
case "Truncate Larger Audio"
    len=min(length(y1),length(y2));
    y=y1(1:len) + y2(1:len);
```

Figure 16 – Mixing audio truncate

3. **Exporting the result as an audio file**

After using "uigetfile" for choosing audio files, it was thought that a similar command should exist for saving files. The "uiputfile" was discovered the same way in MATLAB documentation (uiputfile - Mathworks.com, 2020). This too required the use of figure() function to bring back focus. It was followed with the previously known audiowrite() function to save the audio in the selected location.

```
% Button pushed function: SaveButton
function SaveButtonPushed(app, event)
    if(app.audioMerged==0)
        return;
    end
    % https://www.mathworks.com/help/matlab/ref/uiputfile.html
    [nfname,npath]=uiputfile('.wav','Save sound','output.wav');
    figure(app.UIFigure) %to bring back focus to the app
    if isequal(nfname,0) || isequal(npath,0)
       return   % or whatever other action if 'CANCEL'
    else
       nwavfile=fullfile(npath, nfname);
       audiowrite(nwavfile,app.mergedY,app.mergedFS);
    end
end
```

Figure 17 – Saving audio source code

4. Interface supporting these interactions

This was done with the aid of different tools available in MATLAB App Designer. Before beginning the project, a separate, small course was studied which provided necessary knowledge for building GUI with MATLAB using both GUIDE and the App Designer. From the course, the advantages of the App Designer became very clear, and the course allowed to use the App Designer efficiently. The course was provided in Udemy, titled "MATLAB App Designing: The ultimate Guide for MATLAB Apps", and taught by Nouman Azam (Azam, 2019).

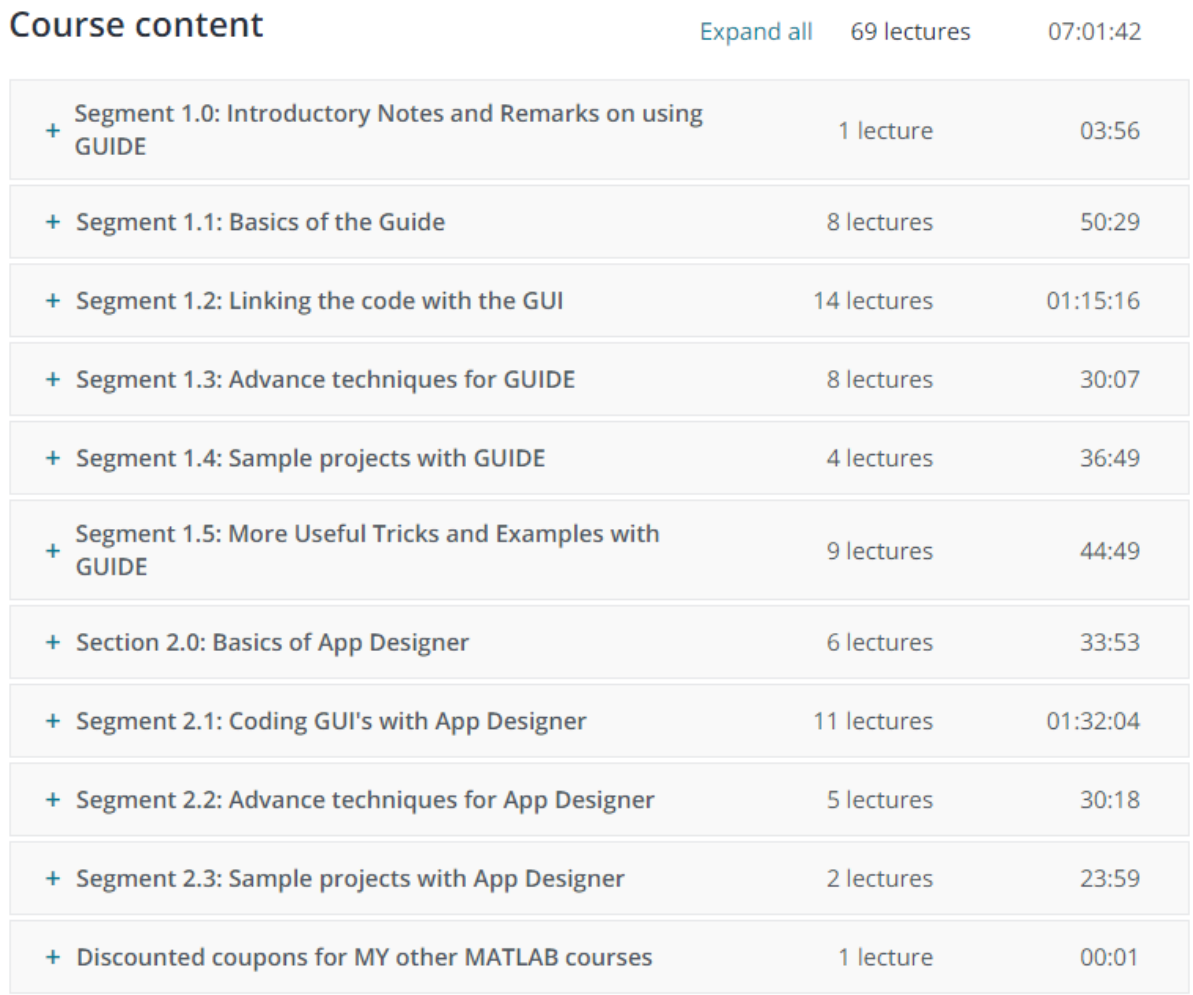


Figure 18 – Udemy GUIDE course

Additionally, the following features have been implemented:

5. Recording audio files

It was thought that allowing to insert a recording from within the application would be a relevant feature. Thus, an option was provided to allow instantaneously recording an audio from within the app to use instead of an existing audio file. The user can click Record button to start the recording and Stop when they are done recording. The recording is first saved in the recordings folder and picked by the application. For this, MATLAB audiorecorder was used (audiorecorder - Mathworks.com, 2020). From the documentation, it was learned that record() will start the recording and stop() will stop the recording. The audio can be retrieved by using the getaudiodata function.

```
value = app.RecordAudio1Button.Value;
global recorder;
if(value==1) % When record clicked
    app.RecordAudio2Button.Enable=0;
    app.RecordAudio1Button.Text="Stop";
    app.RecordAudio1Button.Icon = 'muted.png';
    recorder = audiorecorder;
    record(recorder);
else % When stop clicked
    app.RecordAudio2Button.Enable=1;
    app.RecordAudio1Button.Text="Record";
    app.RecordAudio1Button.Icon = 'microphone.png';
    stop(recorder);
    recordingFileName = generateRandomRecordingName(app);
    recordingName = strcat('recordings/', recordingFileName);
    audioFile = getaudiodata(recorder)
    audiowrite(recordingName,audioFile, 8000);
```

Figure 19 – Recording audio source code

It is also worth mentioning that the MATLAB “randi” function from the lectures, combined with the date command, were used to produce a randomly generated file name for each recording so that one recording does not replace the other. These recordings can be accessed later separately through the explorer or can be imported to the application through the file picker.

6. Different options for merging audios

Three different options have been provided for merging the audios. As discussed earlier, these are implemented as an advanced functionality. The first option is to concatenate the audios where the second audio is added after the first audio. The second option is the option discussed earlier where the larger audio is truncated. The third option is the most advanced one and allows to specify at what exact point of time should the secondary audio get merged. For example: if the Audio 1 is 50 seconds long, and the Audio 2 is 10 seconds long, the user can specify to merge the audio at 15th second of Audio 1 and only the portions 15-25 is merged. The remaining portions are padded with 0 for Audio 2 to make it a 50 second audio as well. This was referenced to an answer provided in MATLAB forums about padding an array with zeros (KSSV, 2016). Although selection of specific duration of audio is provided for both the audios, it was thought that it would be relevant to provide this feature as well.

```
switch app.MergeTypeSelect.Value
    case "Concat"
        y = [y1(:); y2(:)];

    case "Truncate Larger Audio"
        len=min(length(y1),length(y2));
        y=y1(1:len) + y2(1:len);

    case "Audio 1 Primary"
        % https://www.mathworks.com/matlabcentral/answers/312690-how-to-add-zeros-on-the-end-of-

        val=app.SecondaryAudioStartText.Value;
        app.SecondaryAudioStartText.Limits=[0,audioLength1+10];

        zb = zeros(floor(f2*val),1);
        if((audioLength2+val)>audioLength1)
            zpa = zeros(floor((audioLength2+val)-audioLength1)*f2,1);
            za=zeros(0,1);
        else
            za = zeros(floor(audioLength1-(audioLength2+val))*f2,1);
            zpa = zeros(0,1);
        end

        y2 = [zb;y2(:);za;];
        y1 = [y1(:); zpa;];
```

Figure 20 – Different types of merge

7. Selecting volume levels of particular audio

This feature allows selection of volume levels for both Audio 1 and 2 for playing them or before merging them. One audio could have a relatively higher volume compared to the other. This feature allows manually manipulating volume as desired. Sometimes, it could also be relevant to have the background music at a much lower volume and have the main focus on audio with someone speaking. The manipulation of volume was referenced to class lecture slides on Audio. Simply the slider value is retrieved and multiplied with the "y" value.

```
y1=y1*app.VolumeSliderAudio1.Value;
```

Figure 21 – Volume levels for audio 1

8. Selecting volume levels of final audio

Likewise, the volume levels can be adjusted for the final audio as well. Sometimes, some audio files could be very high or low in volume. This feature allows to manipulate this.

Figure 22 – Volume level for merged audio

9. Play/pause/stop audio

In order to implement the audio player feature, the audioplayer object was referenced in the MATLAB documentation (audioplayer - Mathworks.com, 2020). The play(), pause() and stop() commands that can be used on this object were used to implement this feature. These were integrated on callback functions of relevant buttons.

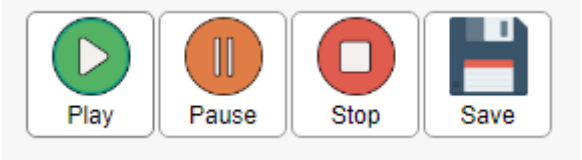


Figure 23 – Audio controls

These features are handled correctly such that they don't throw an error to null pointer reference whenever audios are not selected. Also, the state of audio is set globally so that if the audio is previously playing, the play button will resume the audio instead of playing it from the beginning.

```
function PlayButtonPushed(app, event)
    global audiostate;
    if(app.audioMerged~=0)
        if(audiostate=="Playing")
            resume(app.audioPlayerMerged);
        else
            play(app.audioPlayerMerged);
        end
        audiostate = "Playing";
    end
end
```

Figure 24 – Play button pushed source code

10. Selecting particular duration of audio to play or merge

We have looked that after recording or selecting an audio into the application, the audio can be played, paused, stopped, resumed and the volume levels can be changed. Similarly, the audio could be cropped to play only desired duration of the audio. In the screenshot below, the audio, which was originally 88.3519 seconds, was cropped to play only 5-15 seconds. The same duration of audio will be used while merging it with another audio as provided in the duration text fields.

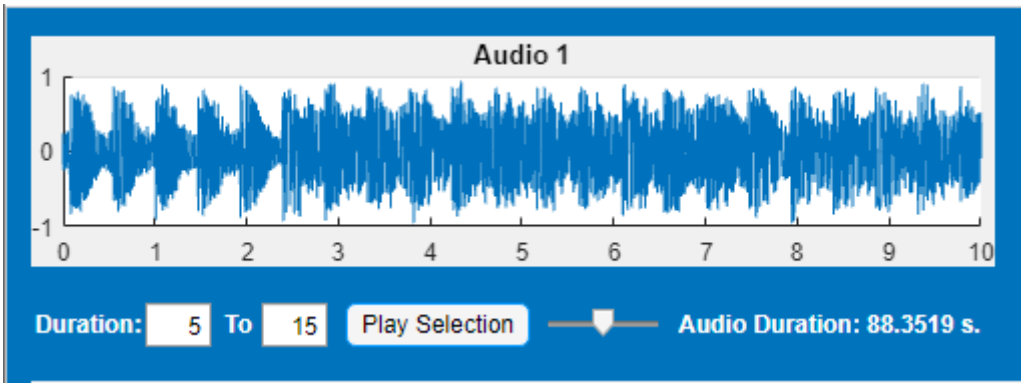


Figure 25 – Duration of audio

After retrieving the audio, its size was determined by using the FS (which is the sample rate) and the size function on y (which is the sampled data). Another answer by KSSV was referenced to select relevant portions of the audio (KSSV, 2018). The audio which lies between the selected text fields is selected and filtered in y. Only this portion of audio is displayed in the figure for allowing the users to know which portion of the audio has been selected visually.

```
[y, FS] = audioread(app.audio1);
% https://www.mathworks.com/matlabcentral/answers/377742-how-do-i-extract-a-few-seconds-from-a-g
[m,n]=size(y);
dt=1/FS;
t=dt*(0:m-1);
idy = (t>=app.Audio1MinTextField.Value) & (t<app.Audio1MaxTextField.Value);
y = y(idy);
y=y*app.VolumeSliderAudio1.Value;
app.audioPlayer1=audioplayer(y, FS);
```

Figure 26 – Selecting particular duration of audio source code

11. Intuitive GUI

As mentioned before, the GUI was prepared after studying the course at Udemy. The features of App Designer which allows easy manipulation of the GUI and generates object-oriented code both of which can be easily switched between in the code view and the design view were used effectively to prepare the final interface. The functions, properties and callbacks were utilized effectively for preparing the application.

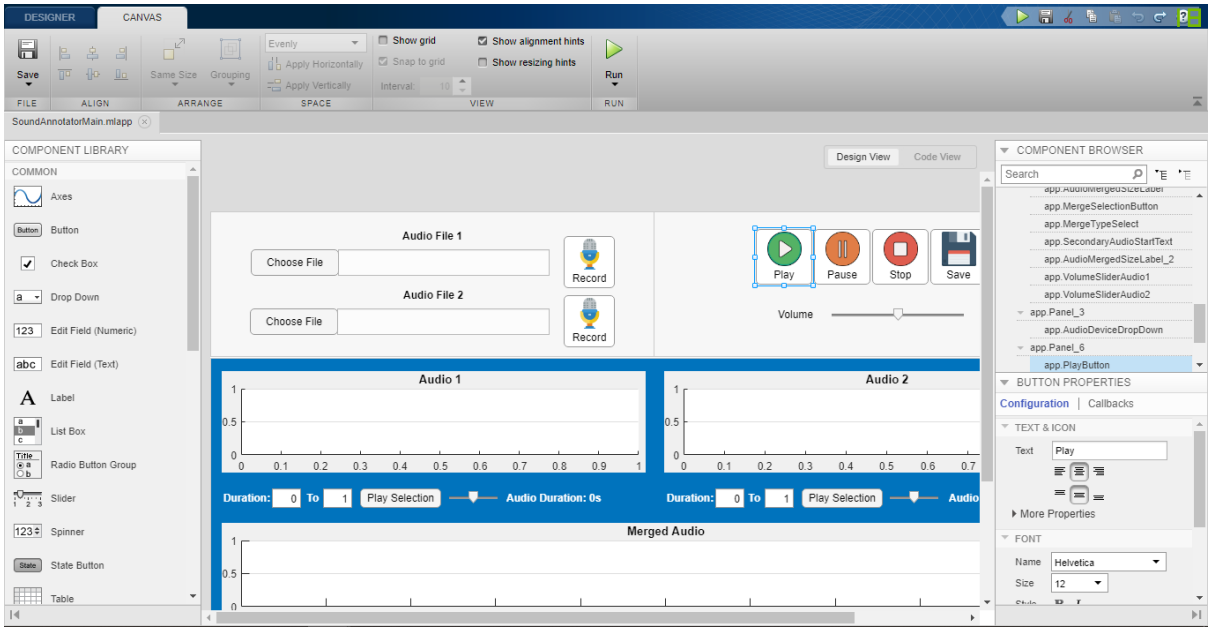


Figure 27 – Interface preparation using App Designer

12. Preventing Errors

Functions such as isa() have been used to check whether the global variables or class properties are double or an array. For example: it is necessary to determine whether both audio files are selected before attempting a merge operation. Thus, without selecting both audio files, clicking the merge button will not have any effect which would otherwise throw an error.

```
function MergeSelectionButtonPushed(app, event)
    if(isa(app.audio2,'double')==1|isa(app.audio1,'double')==1)
        return;
    end
```

Figure 28 – Preventing errors source code

13. Plotting Audio Waves

The audio waves are plotted and updated each time an audio is recorded, selected, changed in duration and volume or merged. This is done to allow visual representation to the user about what is going on and whether they are getting exactly what they think. For example: if two audio files are selected, and volume level for audio 2 is reduced to minimum. Merging these two files will show exactly in the plot about significantly low volume in the second audio. This feature improves usability and is a must-have. The correct way to plot the audio files according to seconds was referenced to an answer provided by Wayne King (King, 2011).

```
app.audioPlayer1=audioplayer(y, FS);
% https://www.mathworks.com/matlabcentral/answers/22112-how-to-plot-wav-file
t = 0:dt:(length(y)*dt)-dt;
plot(app.UIAxesA1,t, y);
```

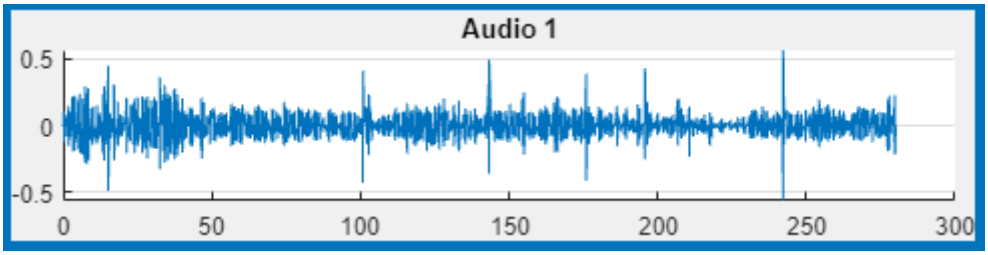
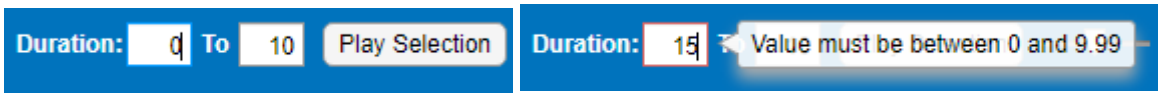


Figure 29 – Plotting audio in axes

14. Minimum and Maximum Duration Constraints

After recording or selecting any audio, it can be played by clicking the play selection button. The user is allowed to choose a duration of audio to be played by entering values in the text fields. There should be some constraints while allowing this feature. The minimum value should not exceed the maximum value and the maximum value should not exceed the total length of the audio. The following code ensures that the maximum value of these text fields are correctly set. Each time the value of maximum text field is updated, the max limit for the minimum text field is also updated.




```
app.audioPlayer1=audioplayer(y, FS);
dt = 1/FS;
t = 0:dt:(length(y)*dt)-dt;
audioLength1=t(length(t));
plot(app.UIAxesA1,t, y);
app.AudioSize1Label.Text=strcat("Audio Duration: ", strcat(num2str(audioLength1), ' s.'));
app.Audio1MinTextField.Limits=[0,audioLength1-0.01];
app.Audio1MaxTextField.Limits=[0.1,audioLength1];
app.Audio1MaxTextField.Value=audioLength1;
```

```
% Value changed function: Audio1MaxTextField
function Audio1MaxTextFieldValueChanged(app, event)
    value = app.Audio1MaxTextField.Value;
    app.Audio1MinTextField.Limits= [0, value-0.01];

end
```

Figure 30 – Minimum maximum duration source code

Some weaknesses of the solution:

- Selection of invalid file format by manually changing the selection to all files results in an error.
- Changes the sampling frequency of sounds which causes sound quality degradation.
- Some repeated code in application to audio 1 and 2 separately. This could be solved by using Object-Oriented Programming features - implementing Audio class and objects.
- Cannot select more than two audios at a time. Allowing dynamic selection of as much audios as desired could be better.
- The audio timer is not displayed and there is no way to know at what time the current audio is playing.

5. TESTING

5.1 Testing

This section will include the test plans and results for the black box approach used as part of testing the system. The test cases and results will be listed in a table. The test screenshots will be listed in the Appendix.

	Test	Steps	Expected Result	Actual Result
1	Running the application	• Run the "startup.m" file	The application starts	Same as expected
2	Loading audio file (invalid)	• Click choose file • Select all files from the selection list • Select an invalid file (different format)	The application should not allow this to happen	Allowed to happen but not allowed to proceed further
3	Loading audio file (valid)	• Click choose file • Select all files from the selection list • Select a valid audio	The audio should be picked and plotted in axes.	Same as expected
4	Recording audio	• Click record button • Click stop when done	The recording should be saved and plotted in axes	Same as expected
5	Playing loaded audio	• Click play selection button after recording or loading audio	The loaded or recorded audio should be played	Same as expected
6	Changing volume	• Change audio volume • Click play button	The volume should be changed as intended.	Same as expected
7	Selecting audio duration	• Provide duration of audio	The selected duration of audio	Same as expected

		• Click play selection button	should be played and plotted	
8	Merge two audios (concat)	• Select/record both audio files • Select Concat in merge type • Click merge selection	The merged audio should be displayed in the graph and can be played by clicking play selection	Same as expected
9	Merge two audios (change volume of audios)	• Change volume of selected/recorded audios • Click merge selection	The merged audio should show the change in audio levels in the graph and it should be noticeable while playing the audio.	Same as expected
10	Merge two audios (change duration of audios)	• Change duration of selected/recorded audios • Click merge selection	The merged audio should contain only the selected duration of both audios.	Same as expected
11	Merge two audios (truncate)	• Provide varying audio durations • Click merge selection	The larger audio should get truncated and the merged audio should have the length of smaller audio.	Same as expected
12	Merge two audios (with primary)	• Select one audio as primary • Provide the secondary audio starting time	No audio should get truncated and only conflicting duration of audios should be played together.	Same as expected

13	Play merged audio	• Merge audios • Click play button	Should play the merged audio	Same as expected
14	Pause merged audio and then play again	• Pause the playing audio • Click play button again	The audio should resume from the paused point	Same as expected
15	Stop merged audio and then play again	• Stop the playing audio • Click play button again	The audio should be played from start	Same as expected
16	Change volume of the merged audio	• Change volume level • Click merge selection	The volume of the merged audio should change and be noticeable on the plot as well as while listening.	Same as expected
17	Save the finalized audio	• Merge audios • Click the save button • Provide relevant filename and path for saving the audio.	The merged audio should be saved in the selected location and should be visible through the file explorer.	Same as expected

6. CONCLUSIONS

6.1 Takes From the Assessment and Module

To summarize, the CSY3010 – Media Technology module has provided a lot of knowledge about different medias and how they are stored, manipulated and transferred in computers. From producing simple vector images to compressing, encoding and transferring large video files containing numerous images, about audio, video, colors in computing, image processing, compression techniques, streaming and synchronization, it has taught a lot in the limited two terms. Skills gained from this module can potentially be implemented across different domains and should boost the CV of an individual for jobs in the media industry.

Personally, this was a highly motivating subject which was of interest since long back. The project as part of this assessment was highly fascinating to work on. Learning that MATLAB provides such ease of creating GUI applications with the aid of App Designer and getting accustomed to MATLAB code structure from existing programming knowledge was the most enjoyable part. Learning as you go, most of the time, the MATLAB documentation provides exactly what you are searching for. Even if you don't find it in the documentation, the online forums have a very active community. As MATLAB is very powerful in directly manipulating medias, it was learned that coding in MATLAB saves a lot of time and lines of code than the most common programming languages.

6.2 What Could be Done With More Time

The system as part of this project allowed merging two audio files and different extra features. This was done within a limited time. Several extra features could be added if more time was available. First and foremost, the existing weaknesses mentioned in the Evaluation could be solved. This introduces features such as allowing more than two audios to be merged, audio controls, etc. Additionally, features such as allowing selection of video along with audio could be added where the user selects a video and adds audio to specific parts of the video. Similarly, adding different sound effects to the imported audio could be another feature. Almost all the media players provide the ability to select different modes of audio such as Bass, Electric, etc. That would add more flavor to the system in application to Media Technology. In a nutshell, the media technology module has provided a lot of knowledge required to implement these extra features and it would have been very viable to implement these if more time was available.

REFERENCES

Media Technology Definition - Webpage

Learn.org (2020). *What Is Multimedia Technology?* [online] Available at: https://learn.org/articles/What_is_Multimedia_Technology.html [Accessed 7 April 2020].

Digital Audio Definition - Webpage

Webopedia (2020). *What is Digital Audio?* [online] Available at: https://www.webopedia.com/TERM/D/digital_audio.html [Accessed 7 April 2020].

Sound Processing with MATLAB – Book

Weeks, M. (2011). *Digital Signal Procesing Using MATLAB And Wavelets*. Sudbury: Jones and Bartlett Publishers. [Accessed 7 April 2020].

MATLAB with Arduino – IEEE Journal

Silva, S., Soares, S., Valente, A., & Marcelino, S. T. (2015). *Digital sound processing using arduino and MATLAB.* 2015 Science and Information Conference (SAI). Available from DOI: 10.1109/sai.2015.7237295 [Accessed 7 April 2020].

Vehicle Monitoring System with MATLAB – IEEE Journal

Valle, J. M. G., Garcia, J. C. C., & Cadaval, E. R. (2017). *Electric vehicle monitoring system by using MATLAB/App Designer.* 2017 International Young Engineers Forum (YEF-ECE). Available from DOI: 10.1109/yef-ece.2017.7935642 [Accessed 7 April 2020].

MATLAB GUI App – ResearchGate Journal

Antonios. A.S. & Anastasios, A. (2009). *Matlab GUI application for teaching control systems*. Proceedings of the 6th WSEAS International Conference on ENGINEERING EDUCATION. pp. 208-211.

MATLAB File Exchange

Mathworks.com (2020). *File Exchange – MATLAB Central*. [online] Available at: https://www.mathworks.com/matlabcentral/fileexchange [Accessed 7 April 2020].

MATLAB GUIDE vs App Designer

Mathworks.com (2020). *Comparing GUIDE & App Designer – MATLAB & Simulink*. [online] Available at: https://www.mathworks.com/products/matlab/app-designer/comparing-guide-and-app-designer.html [Accessed 9 April 2020].

MAGIX Music Maker - Software

MAGIX (2020). *Music Maker [OFFICIAL] Download free music software MAGIX!* [online] Available at: https://www.magix.com/us/music/music-maker/ [Accessed 8 April 2020].

FL Studio - Software

FL Studio (2020). *FL Studio* [online] Available at: https://www.image-line.com/flstudio/ [Accessed 8 April 2020].

Audacity – Software

Audacity (2020). *Audacity ® Free, open source, cross-platform audio software for Windows, Mac OS X, GNU/Linux and other operating systems*. [online] Available at: https://www.audacityteam.org/ [Accessed 8 April 2020].

MATLAB File Selection - Documentation

Mathworks.com (2020). *Open File Selection Dialog Box - MATLAB Uigetfile*. [online] Available at: https://www.mathworks.com/help/matlab/ref/uigetfile.html [Accessed 9 April 2020].

Window Minimizes - Forum

Friedrich (2019). *Appdesigner Window Ends Up In Background After Uigetfile - MATLAB Answers - MATLAB Central*. [online] Available at: https://uk.mathworks.com/matlabcentral/answers/296305-appdesigner-window-ends-up-in-background-after-uigetfile#answer_390627 [Accessed 9 April 2020].

MATLAB File Saving - Documentation

Mathworks.com (2020). *Open Dialog Box For Saving Files - MATLAB Uiputfile*. [online] Available at: https://www.mathworks.com/help/matlab/ref/uiputfile.html [Accessed 9 April 2020].

Course from Udemy – Online Course

Azam, N. (2019). MATLAB App Designing: The Ultimate Guide For MATLAB Apps. [online] Udemy. Available at: https://www.udemy.com/course/matlab-graphical-user-interface-using-guide-codes-include/ [Accessed 4 April 2020].

MATLAB Audio Recording - Documentation

Mathworks.com (2020). *Object For Recording Audio - MATLAB*. [online] Available at: https://www.mathworks.com/help/matlab/ref/audiorecorder.html [Accessed 9 April 2020].

Padding Array With Zeros - Forum

KSSV (2016). *How to add zeros on the end of a signal. - MATLAB Answers - MATLAB Central*. [online] Available at: https://www.mathworks.com/matlabcentral/answers/312690-how-to-add-zeros-on-the-end-of-a-signal [Accessed 10 April 2020].

MATLAB Audio Player - Documentation

Mathworks.com (2020). *Object For playing Audio - MATLAB*. [online] Available at: https://www.mathworks.com/help/matlab/ref/audioplayer.html [Accessed 9 April 2020].

Extracting Audio - Forum

KSSV (2016). *How do i extract a few seconds from a given audio file? - MATLAB Answers - MATLAB Central*. [online] Available at: https://www.mathworks.com/matlabcentral/answers/377742-how-do-i-extract-a-few-seconds-from-a-given-audio-file [Accessed 9 April 2020].

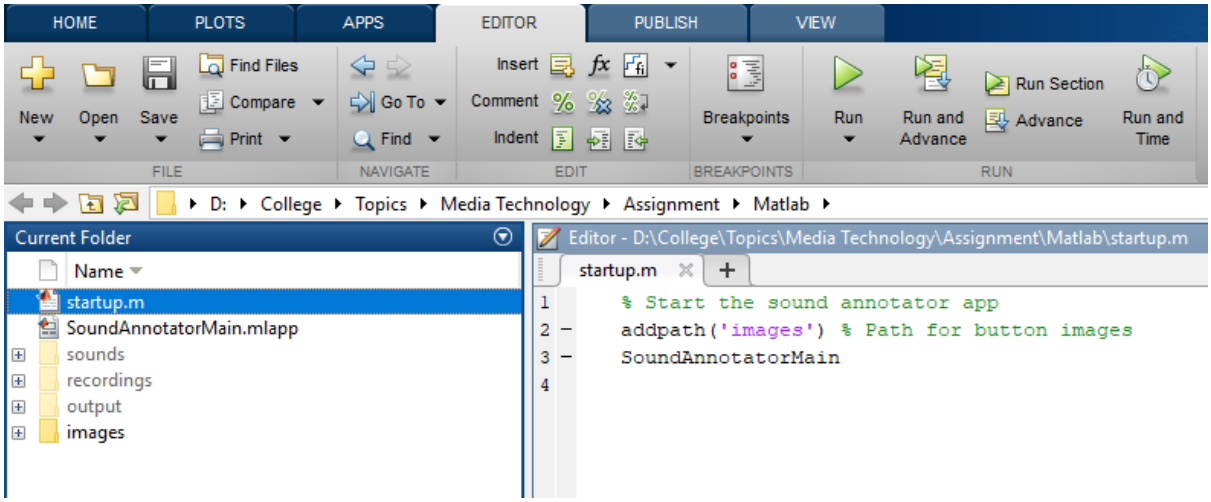
Plotting Sound Wave - Forum

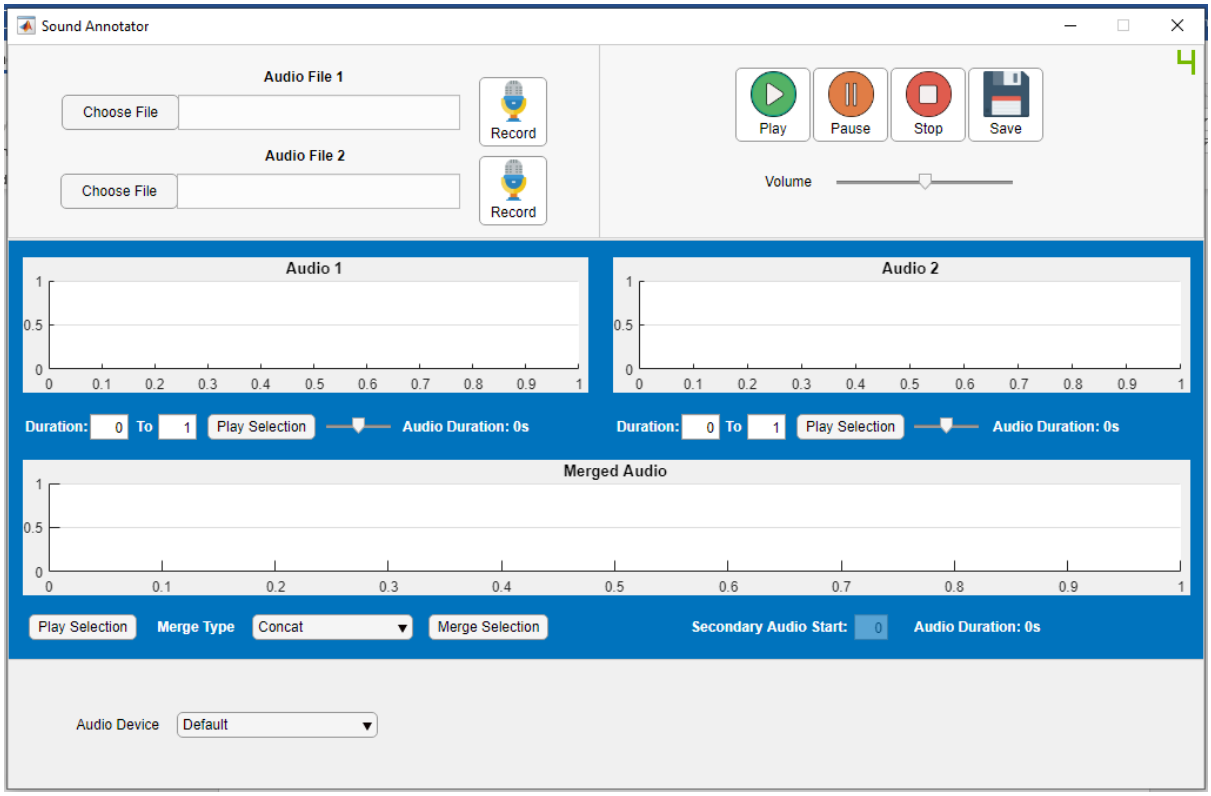
King, W. (2016). *How to plot WAV file - MATLAB Answers - MATLAB Central*. [online] Available at: https://www.mathworks.com/matlabcentral/answers/22112-how-to-plot-wav-file [Accessed 9 April 2020].

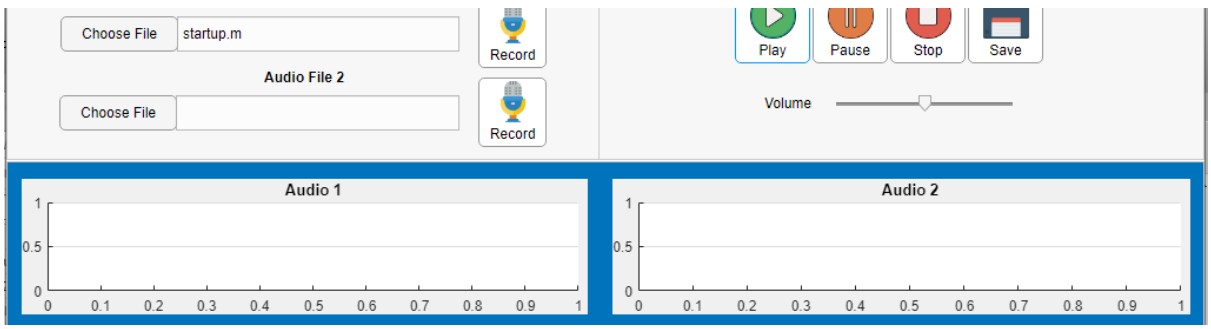
Free Icons for Buttons From: https://www.flaticon.com/

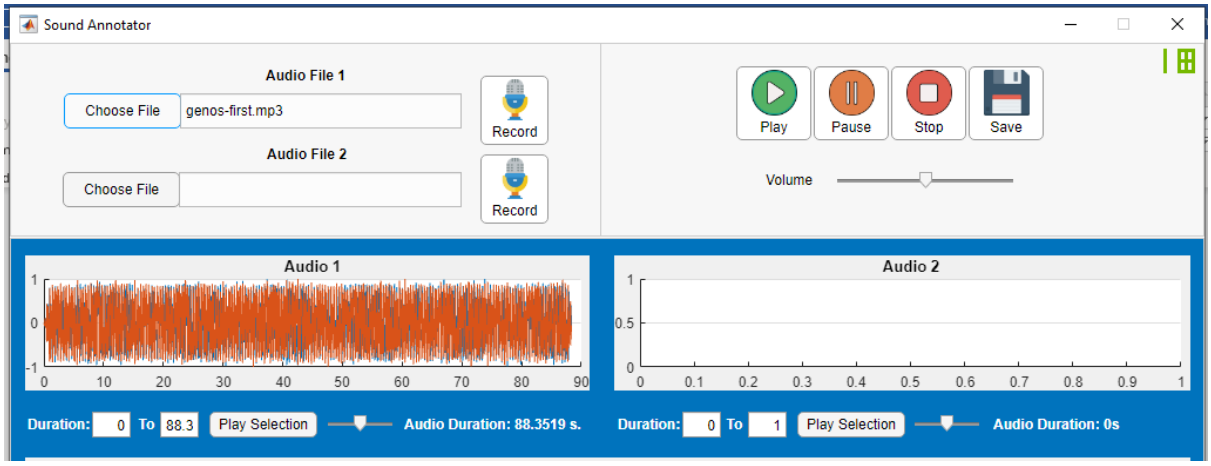
Free Sounds From: https://freesound.org/

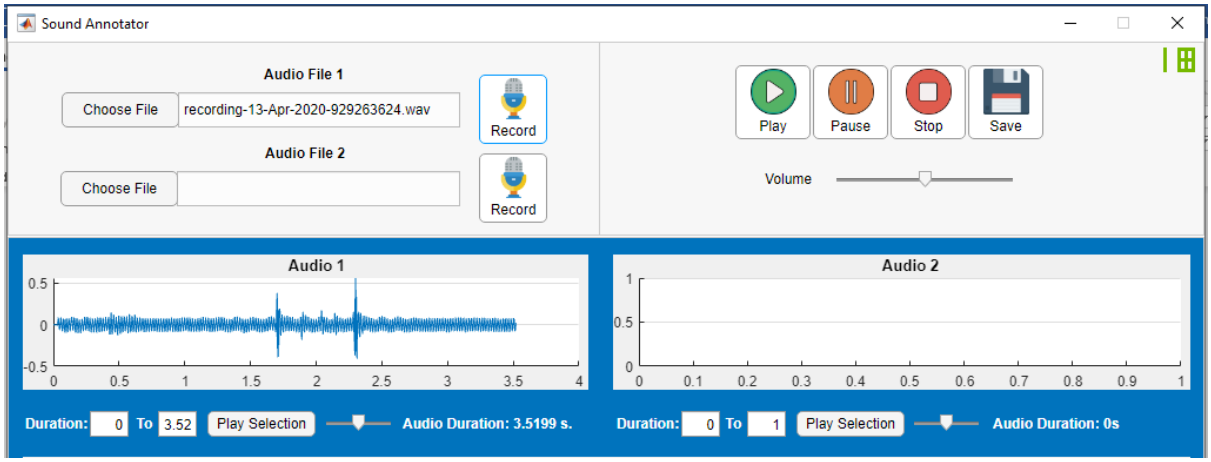
APPENDIX I – TEST SCREENSHOTS





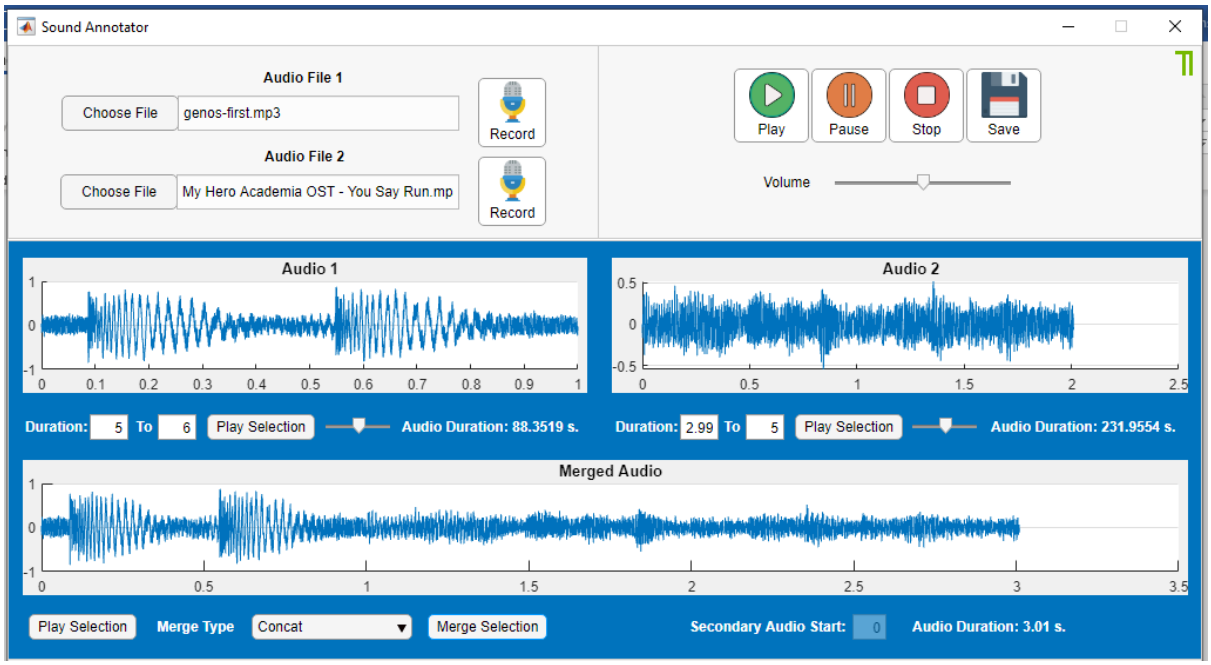


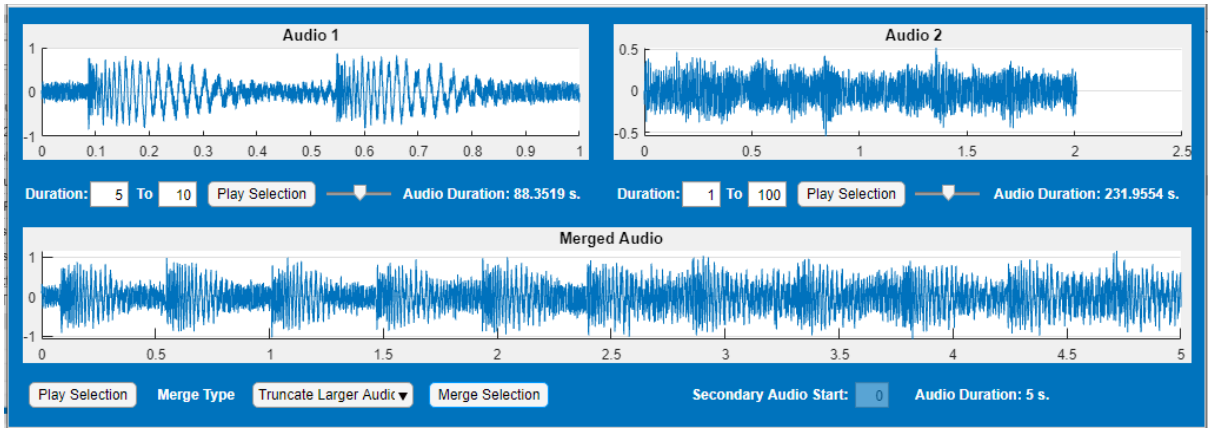

Sound Annotator
Audio File 1
Choose File
genos-first.mp3
Record
Audio File 2
Choose File
Record
Play
Pause
Stop
Save
Volume
Audio 1
Audio 2
Duration: 0 To 88.3 Play Selection
Audio Duration: 88.3519 s.
Duration: 0 To 1 Play Selection
Audio Duration: 0s

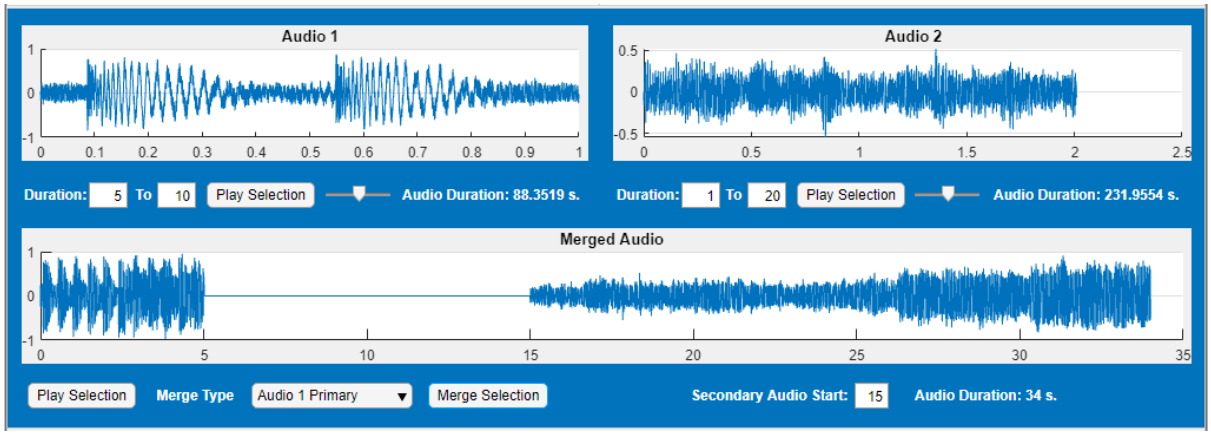

Sound Annotator
Audio File 1
Choose File
recording-13-Apr-2020-929263624.wav
Record
Audio File 2
Choose File
Record
Play
Pause
Stop
Save
Volume
Audio 1
Audio 2
Duration: 0 To 3.52 Play Selection
Audio Duration: 3.5199 s.
Duration: 0 To 1 Play Selection
Audio Duration: 0s

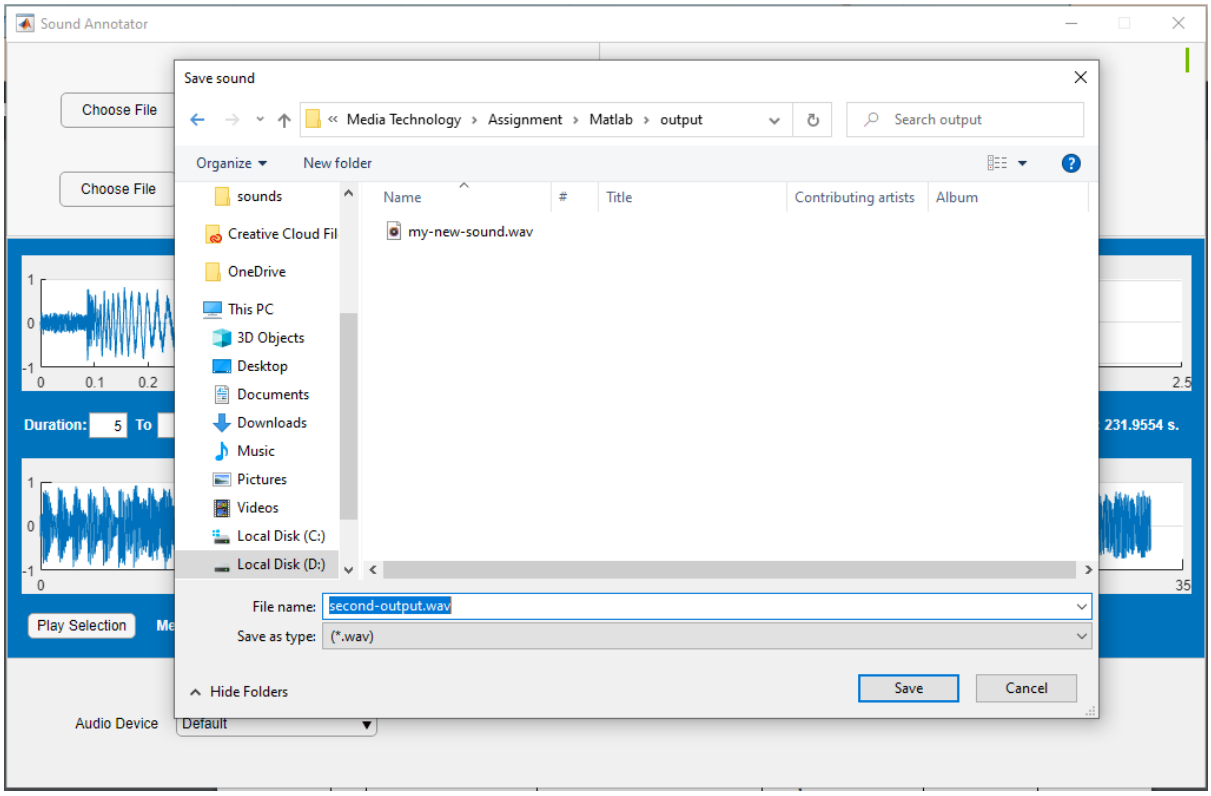

Sound Annotator
Audio File 1
Choose File
genos-first.mp3
Record
Audio File 2
Choose File
My Hero Academia OST - You Say Run.mp
Record
Play
Pause
Stop
Save
Volume
Audio 1
Audio 2
Duration: 0 To 88.3 Play Selection
Audio Duration: 88.3519 s.
Duration: 0 To 232 Play Selection
Audio Duration: 231.9554 s.
Merged Audio
Play Selection
Merge Type
Concat
Merge Selection
Secondary Audio Start: 0
Audio Duration: 320.3072 s.

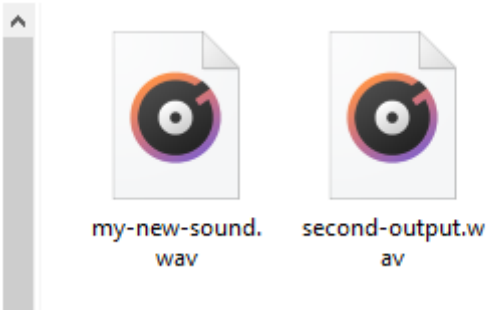

Sound Annotator
Audio File 1
Choose File
genos-first.mp3
Record
Audio File 2
Choose File
My Hero Academia OST - You Say Run.mp
Record
Play
Pause
Stop
Save
Volume
Audio 1
Audio 2
Duration: 0 To 88.3 Play Selection
Audio Duration: 88.3519 s.
Duration: 0 To 232 Play Selection
Audio Duration: 231.9554 s.
Merged Audio
Play Selection
Merge Type
Concat
Merge Selection
Secondary Audio Start: 0
Audio Duration: 320.3072 s.


Sound Annotator
Audio File 1
Choose File
genos-first.mp3
Record
Audio File 2
Choose File
My Hero Academia OST - You Say Run.mp
Record
Play
Pause
Stop
Save
Volume
Audio 1
Audio 2
Duration: 5 To 6 Play Selection
Audio Duration: 88.3519 s.
Duration: 2.99 To 5 Play Selection
Audio Duration: 231.9554 s.
Merged Audio
Play Selection
Merge Type
Concat
Merge Selection
Secondary Audio Start: 0
Audio Duration: 3.01 s.


Audio 1
Audio 2
Duration: 5 To 10 Play Selection
Audio Duration: 88.3519 s.
Duration: 1 To 100 Play Selection
Audio Duration: 231.9554 s.
Merged Audio
Play Selection
Merge Type
Truncate Larger Audio
Merge Selection
Secondary Audio Start: 0
Audio Duration: 5 s.


Audio 1
Audio 2
Duration: 5 To 10 Play Selection Audio Duration: 88.3519 s.
Duration: 1 To 20 Play Selection Audio Duration: 231.9554 s.
Merged Audio
Play Selection Merge Type Audio 1 Primary Merge Selection Secondary Audio Start: 15 Audio Duration: 34 s.


Sound Annotator
Choose File
Choose File
Save sound
Media Technology › Assignment › Matlab › output
Search output
Organize
New folder
sounds
Creative Cloud Fil
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Name
#
Title
Contributing artists
Album
my-new-sound.wav
File name: second-output.wav
Save as type: (*.wav)
Hide Folders
Save
Cancel
Duration: 5 To
231.9554 s.
Play Selection
Audio Device
Default


my-new-sound.wav
second-output.wav