|  |  |  |  |
| --- | --- | --- | --- |
| **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: | Sudish Basnet  18413692 |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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.

* Sudish Basnet

**Demo video link: https://northampton.mediaspace.kaltura.com/media/Media+Technology+As2++2020/1\_nuu4zxr1**

Contents

[1 Introduction 3](#_Toc39405377)

[1.1 Report Introduction 3](#_Toc39405378)

[1.2 Project Introduction 3](#_Toc39405379)

[1.3 Project Aims and Objectives 3](#_Toc39405380)

[1.2 Project Development Methodology 4](#_Toc39405381)

[1.2.1 Problem Domain 4](#_Toc39405382)

[1.2.2 System specification Phase 4](#_Toc39405383)

[1.2.3 System design and testing Phase 4](#_Toc39405384)

[2 Requirements Engineering 4](#_Toc39405385)

[2.1 Elicitation Activities 4](#_Toc39405386)

[2.1.1 Comparable System 4](#_Toc39405387)

[2.1.2 Academic Literature Review 8](#_Toc39405388)

[2.1.3 Ethnography Research 10](#_Toc39405389)

[3 Requirements Specification 10](#_Toc39405390)

[3.1 Functional Requirements 10](#_Toc39405391)

[3.1.1 System functionalities 10](#_Toc39405392)

[3.2 Performance Requirements 11](#_Toc39405393)

[3.3 Design Constraints 12](#_Toc39405394)

[3.4 Commercial Constraints 12](#_Toc39405395)

[4 System Analysis & Design 13](#_Toc39405396)

[4.1 Preliminary Design Stages 13](#_Toc39405397)

[4.1.1 Significant Event Analysis 13](#_Toc39405398)

[4.2 Detailed Static System Designs 14](#_Toc39405399)

[4.2.1 Class diagram 14](#_Toc39405400)

[4.2.2 Use case diagram 14](#_Toc39405401)

[4.3 Detailed Dynamic System Designs 15](#_Toc39405402)

[4.3.1 Dynamic Diagrams 15](#_Toc39405403)

[5 System Interface Designs 16](#_Toc39405404)

[5.1 Draft Interface Designs 17](#_Toc39405405)

[5.1.1 System Wireframe 17](#_Toc39405406)

[5.1.2 System mock-up 20](#_Toc39405407)

[5.1.3 System Navigation and Event Diagram 22](#_Toc39405408)

[6 Design Revisions 26](#_Toc39405409)

[7 System Build and Technical Notes 26](#_Toc39405410)

[7.1 System events and class attributes formation 28](#_Toc39405411)

[7.2 System technical notes recap 41](#_Toc39405412)

[8 Test Strategy 41](#_Toc39405413)

[8.1 Unit Testing: 42](#_Toc39405414)

[8.2 Black Box Testing 42](#_Toc39405415)

[9 System Evaluation 43](#_Toc39405416)

[9.1 Heuristic Evaluation 43](#_Toc39405417)

[9.2 System Pilot Trials 44](#_Toc39405418)

[10 Project Conclusions 44](#_Toc39405419)

[11 References 45](#_Toc39405420)

[12 Project Gantt Chart 46](#_Toc39405421)

[Figure 1 : Audio Joiner 6](#_Toc39405422)

[Figure 2 : Audio Editor 7](#_Toc39405423)

[Figure 3 : Online Converter 8](#_Toc39405424)

[Figure 4 : Apowersoft Free Online Editor 9](#_Toc39405425)

[Figure 5 : TWISTED WAVE 9](#_Toc39405426)

[Figure 6 : System class diagram 16](#_Toc39405427)

[Figure 7 : System use case diagram 17](#_Toc39405428)

[Figure 8 : System main GUI wireframe 19](#_Toc39405429)

[Figure 9 : Tone Generation GUI wireframe 20](#_Toc39405430)

[Figure 10 : Audio Recording GUI wireframe 21](#_Toc39405431)

[Figure 11 : System main GUI mockup 22](#_Toc39405432)

[Figure 12 : Audio Record GUI mockup 23](#_Toc39405433)

[Figure 13 : Tone Generation GUI 24](#_Toc39405434)

[Figure 14 : System Navigation and Buttons flowchart 25](#_Toc39405435)

[Figure 15 : Mixing or Overlapping two audio flowcharts 25](#_Toc39405436)

[Figure 16 : Merging Audio Flowchart 26](#_Toc39405437)

[Figure 17 : Reverse Audio Flowchart 26](#_Toc39405438)

[Figure 18 : Audio Trim Flowchart 27](#_Toc39405439)

[Figure 19 : Record Audio Flowchart 27](#_Toc39405440)

[Figure 20 : Tone Generation Flowchart 28](#_Toc39405441)

[Table 1 : Overall Comparable System Overview 10](#_Toc39405442)

[Table 2 : Academic literature review 11](#_Toc39405443)

[Table 3 : System functionalities 12](#_Toc39405444)

[Table 4 : Acceptance Performance Tests 14](#_Toc39405445)

[Table 5 : System event analysis 15](#_Toc39405446)

# 1 Introduction

## 1.1 Report Introduction

This is the report for the first assignment of module CSY3010 for second term of year 3 provided by Northampton University. This report justifies all the work done to complete the assignment. Report consist of several section starting from introduction to report and project for the system implementation, strategies that will be followed out with proposed requirement engineering followed by several system designs and the testing for the system as well as system evaluation so that the system can be used for the given purpose after the completion.

## 1.2 Project Introduction

This project is about the development of a sound annotator using MATLAB where the given scenario is about a post-production company which adds voice commentary to the existing football match . The basic criteria for the system are to import two audio files which can be used to mix as background and the front sound for the football match. The mixed version of sound will then be exported to required directory. With the functional requirement system should have a good user interface which make users interaction smoother. The extra functionalities that will be covered for this system will be like generating new tones which can be added to the audio as the opening or as preferred, reversing the audio if needed, slow down the audio speed if needed, joining the two audio as required.

## 1.3 Project Aims and Objectives

The main aim for this project is to create a software for mixing two audio sounds which is for the football match so that the commentary can be added to the preferred sound. The interface should be as friendly for users so that they can able to know the exact meaning of each navigations an able to import two audio files which is then mixed and then exported to preferred directory. Beside the main aim, the project is building several more functionalities to the system so that it will have great positive impact on the users.

To meet the project aim, there are several strategies which will be followed out without which the system cannot be completed. Those strategies are as below:

* Understand the brief provided for the basic functionalities.
* Background research on existing system like this.
* Know the problem domain with different elicitation strategies and cover their solution.
* Utilize requirement engineering so that system will not fail after the completion.
* Use several draft diagrams for the finalization of system user interface.
* Implement the system with collected information.
* Carryout several functionalities if can be implement.
* Test the system with basic methods for system accuracy.
* Evaluate the system.
* Finalize the system.

.

## 1.2 Project Development Methodology

### 1.2.1 Problem Domain

The problem with existing system available is either we can add audios or mix audios which is under the limitation when you try to make new sounds with the existing sounds.

### 1.2.2 System specification Phase

With the analyzation of research and brief, system with the basic feature of sound mixing is finalized with the other key features for more smoothness in system where there won’t be any restriction to users so every individual can access every features provided by the system.

### 1.2.3 System design and testing Phase

On completion of background research and outcomes of several solution and ideas, designing of system is started where draft designs like system wireframes, mockups, and several flowchart showing the system workflow is determined upon which new system is designed and implemented. To make the user interface more friendly, different signs and symbols will be used throughout every clickable event . Color schemes are selected accordingly to create a suitable working environment.

# 2 Requirements Engineering

To determine and document the requirement for the system so that it will be easier to maintain and build with every key features of solution system to be built. This section consists of elicitation activities and requirement specification.

## 2.1 Elicitation Activities

### 2.1.1 Comparable System

1. **Audio Joiner**

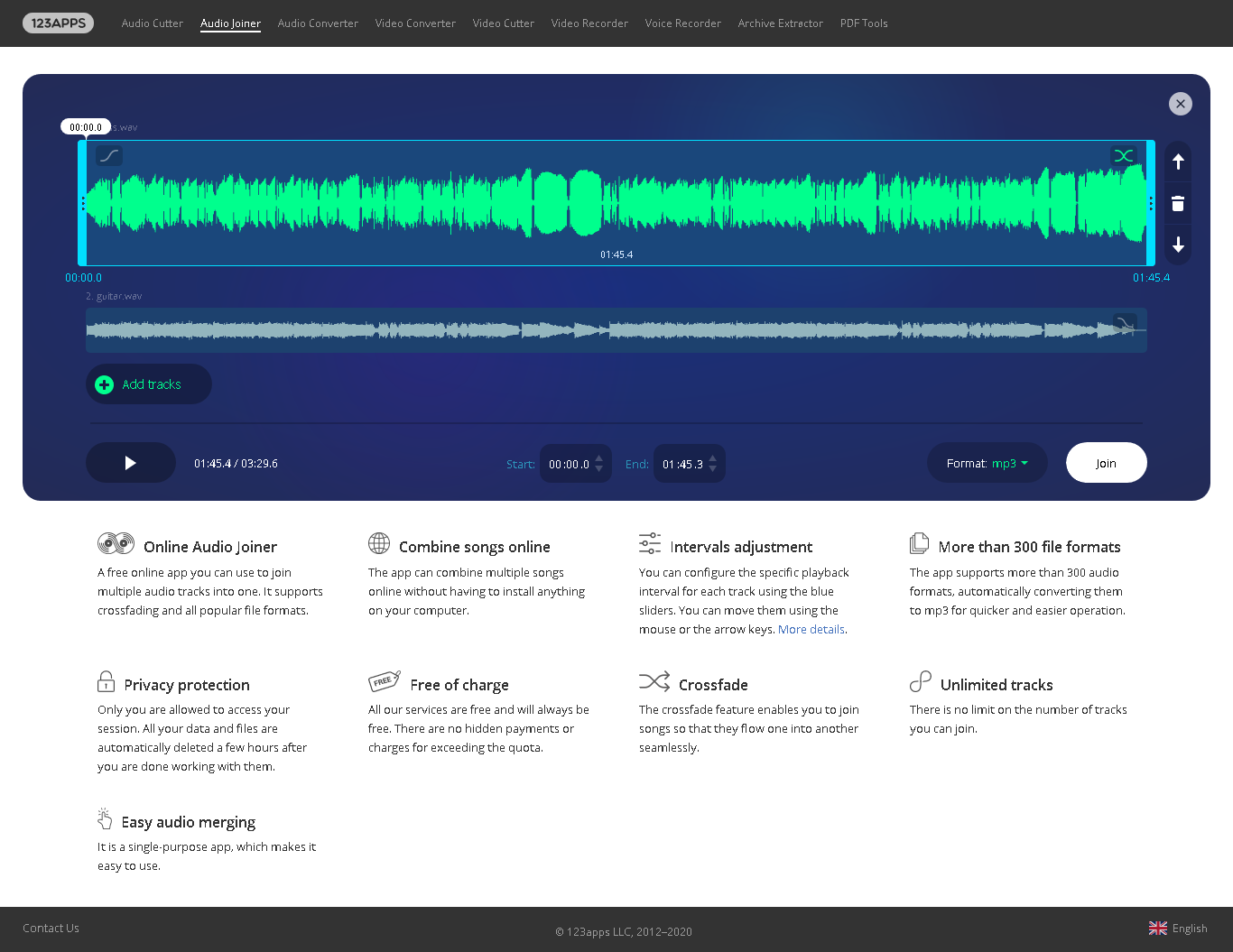


Figure 1 : Audio Joiner

The application can be assessing at URL (https://audio-joiner.com/) which is an online web app with the motive of merging multiple audio. The GUI is very attractive and users for this app is rapidly increasing. Performance speed is slow.

1. **Audio Editor**

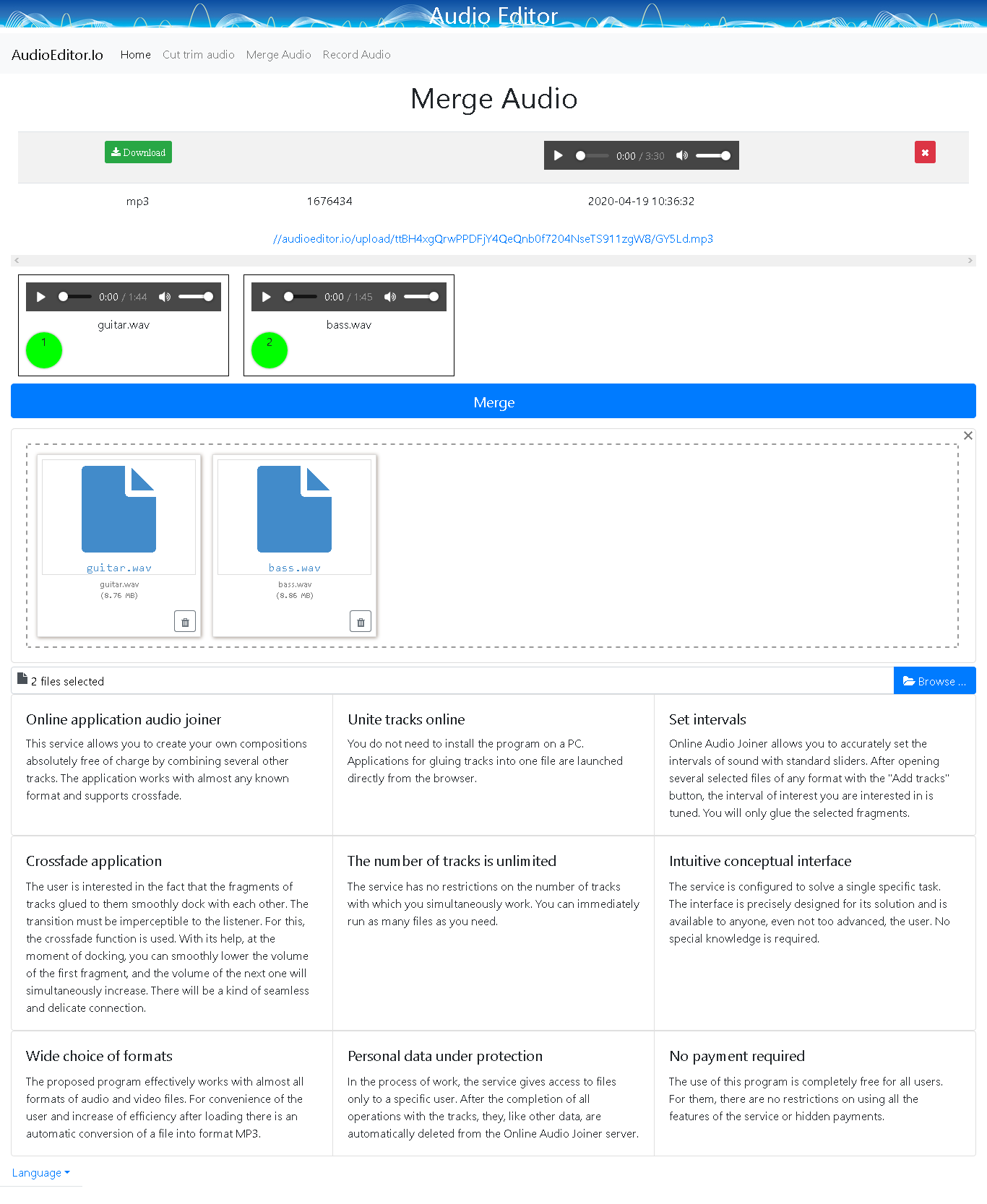
****

Figure 2 : Audio Editor

The system can be assess at URL(<https://audioeditor.io/merge-audio/>). The system GUI is average where the system functionality is to merge audio. Performance speed is good.

1. **Online Converter**

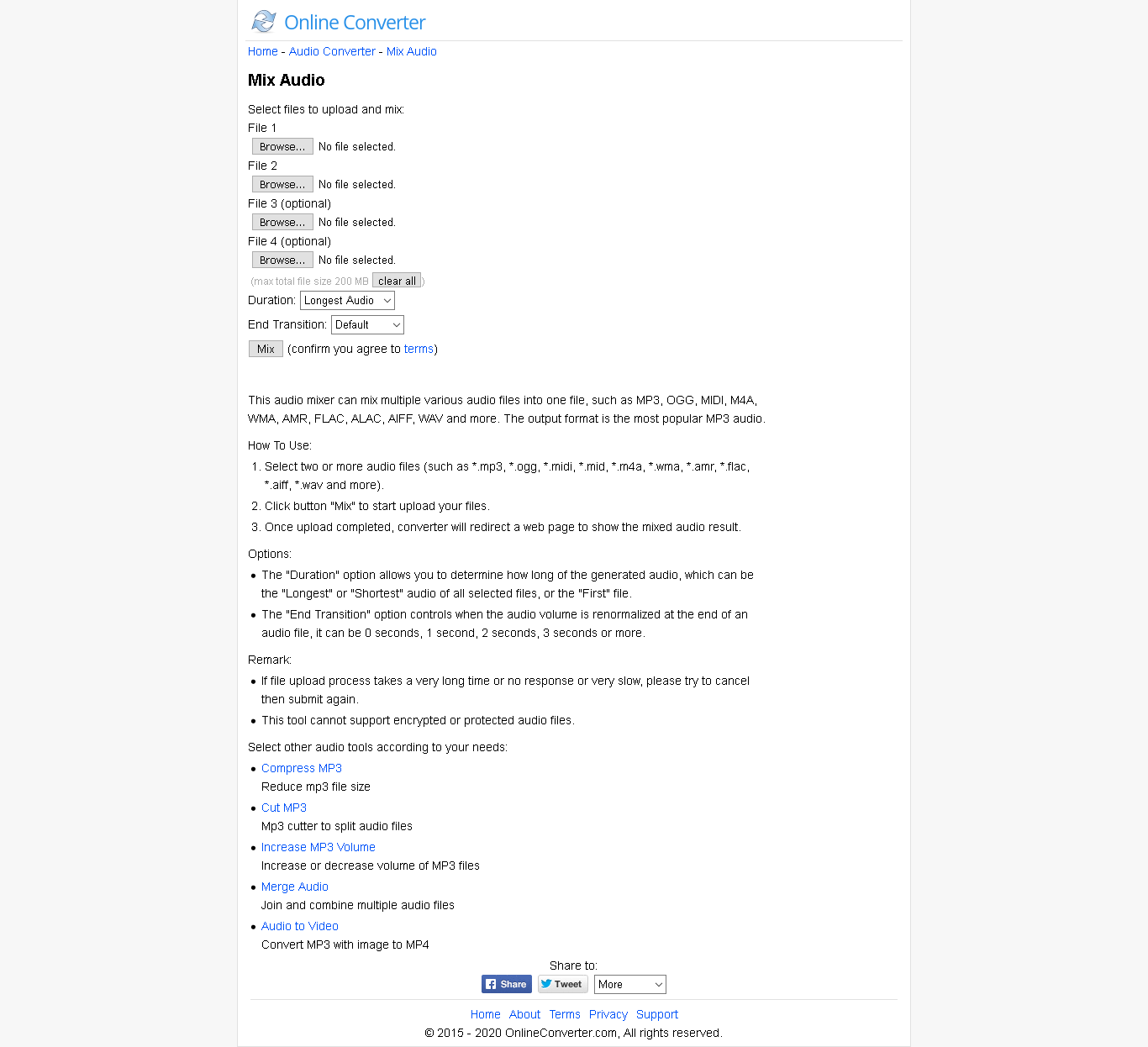


Figure 3 : Online Converter

The system purpose is to mix the multiple audio sounds which can be imported and then downloaded after the completion. The GUI not much attractive and the performance speed is a bit slower. The system can be assess at URL (<https://www.onlineconverter.com/mix-audio>).

1. **Apowersoft Free Online Audio Editor**

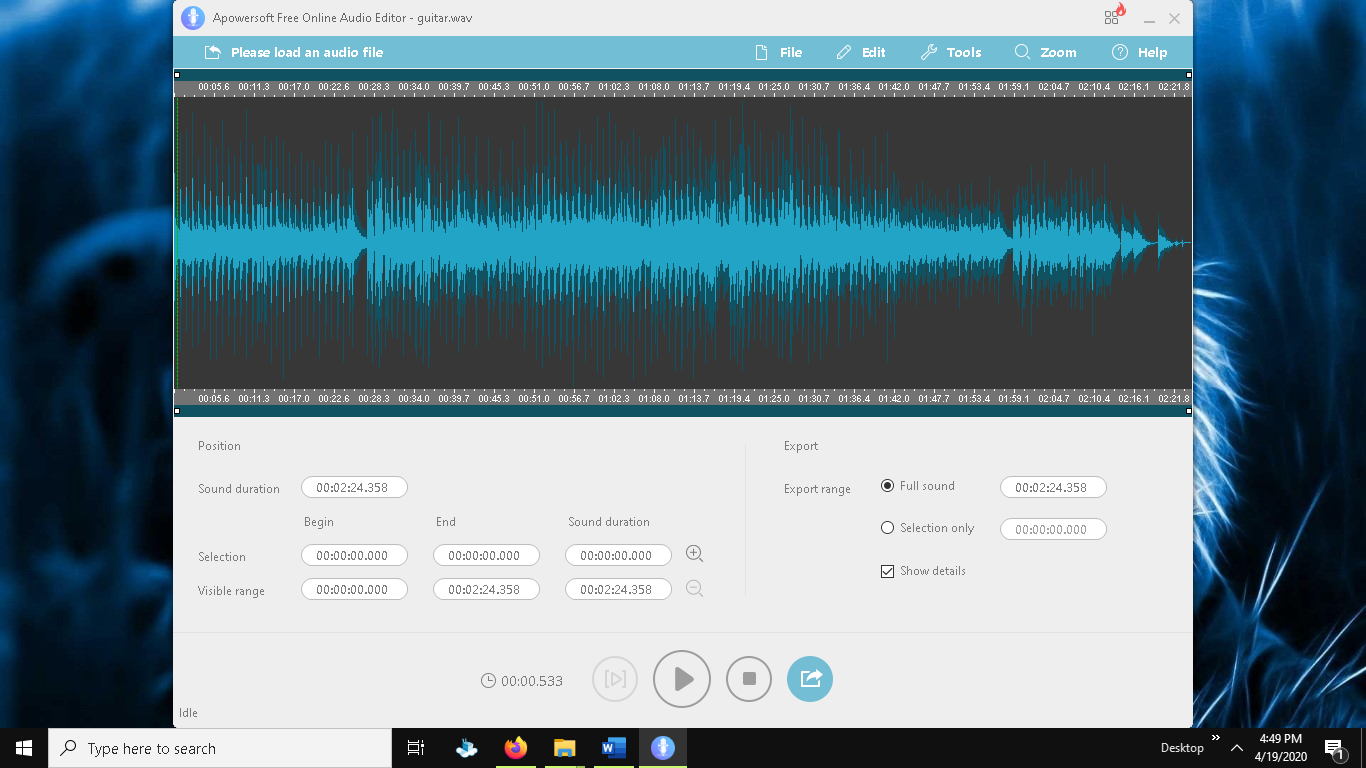


Figure 4 : Apowersoft Free Online Editor

The source for download or online access can be found at URL(<https://www.apowersoft.com/free-online-audio-editor>) which can be installed accordingly. The GUI is attractive, and performance is a bit slower.

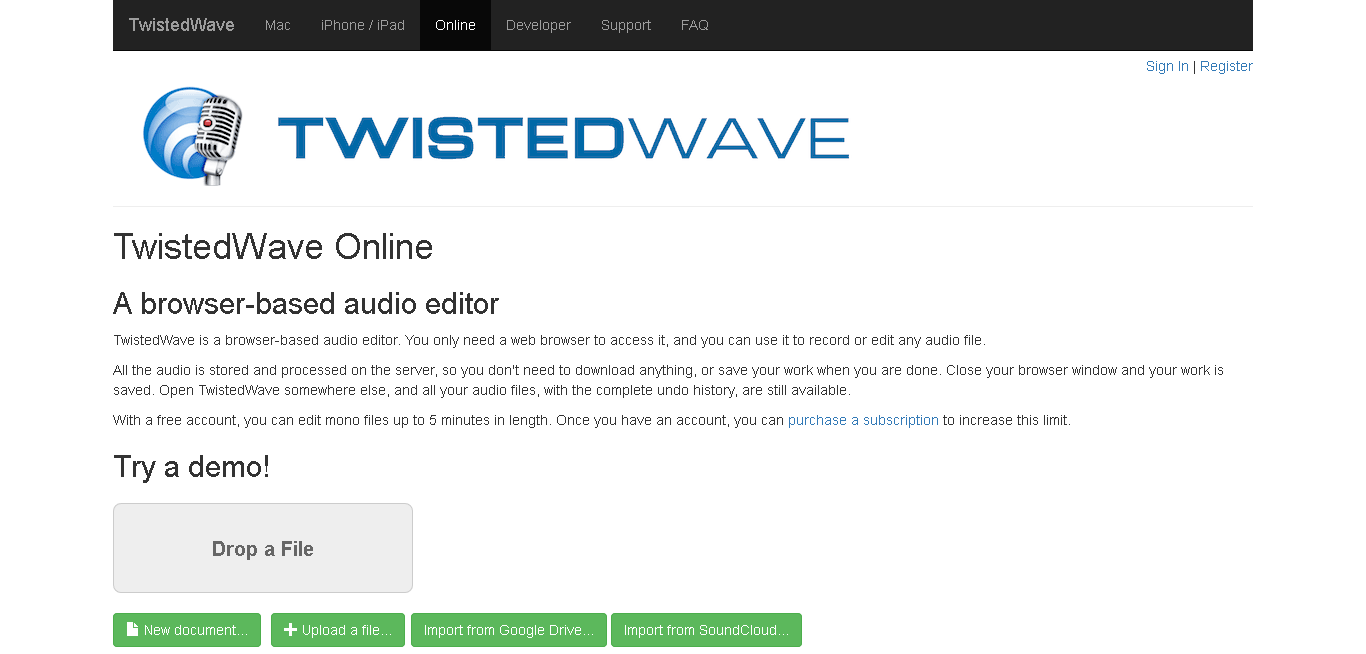
1. **TWISTED WAVE**

Figure 5 : TWISTED WAVE

This system can be found online as well as I the desktop version to which is access through URL(https://twistedwave.com/online) . The motive for this application is to edit audio like changing volume, reversing audio.

Different comparable system with their individual software motive are taken in account for the comparable system as part of elicitation activity for the project and the outcomes from the system with some check in is listed below in table.

Table 1 : Overall Comparable System Overview

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Existing system functionalities and features | Audio Joiner | Audio Editor | Online Converter | Apowersoft Free Online Audio Editor | TWISTED WAVE |
| GUI | Attractive | Average | Below Average | Attractive | Below Average |
| Merge Sound | Yes | Yes | No | No | No |
| Mix Sounds | No | No | Yes | Yes | No |
| Play merged sound in system | No | Yes | No | Yes | No |
| Performance Speed | Slow | Fast | A bit slow | A bit slow | Quite fast |
| Voice Record | No | No | No | No | No |
| Generate Tone | No | No | No | No | No |
| Reverse Audio | No | No | No | No | Yes |
| Limit and Play audio sections | Yes | Yes | No | Yes | No |
| Change speed of audio file | No | No | No | Yes | Yes |
| Download/ Export Sound | Yes | Yes | Yes | Yes | Yes |
| Volume Change | No | No | Yes | No | Yes |

### 2.1.2 Academic Literature Review

Moreover, on research of several articles and journals for development of sound annotator, researcher have carried out few points so that it can ease the software development.

Table 2 : Academic literature review

|  |  |  |
| --- | --- | --- |
| Title | Source | Subject relevant topics |
| Audio Mixing. | (effects, 2020) | * The invention relates to a method and an apparatus for mixing audio signals. * According to the invention, a set of audio channels are arranged in a mixing stack and a mixed audio signal is produced from the topmost channels above a predetermined threshold level in the stack. * Further, it is desirable to maximize the number of audio channels to mix, even if the current number of active participants are low. * Another problem with the above described prior art is that such a scheme for mixing audio streams in certain common situations will result in an annoying switching behaviour in the background noise. |
| A Complete School Management System | (Semantic Annotation and Retrieval of Music and Sound Effects - IEEE Journals & Magazine, 2020) | * We consider the related tasks of content-based audio annotation and retrieval as one supervised multiclass, multilabel problem in which we model the joint probability of acoustic features and words. * We also show that our audition system is general by learning a model that can annotate and retrieve sound effects. |

### 2.1.3 Ethnography Research

To Ethnography Research to get deep information about how the existing system works, Apowersoft Free Online Audio Editor is chosen which is use for mixing or overlapping two audios. Below are the process and the time required for each process in the system.

* Open the online link of the provider
* Download or choose option to use online
* The GUI opens after the installation or choosing online platform
* Select files to upload which acquired 10sec for uploading 4 mb file each
* Click on merging files which takes more than 10 sec
* User have option either to export or play there

For the merging of audio sounds, Audio Joiner is chosen for ethnography research where the carried steps are given below:

* Open the online platform link which responses fast
* Upload or drag and drop multiple audio files which takes more than 15 sec for 4mb of each file.
* User can select the audio format
* User can now join the audios and play according to given time sections
* User can export the new audio which is bigger than actual size.

# 3 Requirements Specification

## 3.1 Functional Requirements

### 3.1.1 System functionalities

The audio editor which is developed for the CSY3010 module as part of assignment, it consists of many key functionalities so that it can ease the company who asked to develop solution system for their needs so that they can mix the two audios which is the basic requirement of the company. Beside the basic functionality, several other requirements have been carried by the developer which are listed below according to the events area.

Table 3 : System functionalities

|  |  |
| --- | --- |
| **Area** | **Event Description** |
| **Home panel** | * Upload track 1 * Upload track 2 * Record Audio * Play , Pause, Stop Track * Select the mice for input operation of audio * Select for speaker for output operation of sound * Manage the volume * Mange the playback speed * Reverse the track selected * Trim the track selected * Generate tone accordingly * Input play duration section and play it * Adjust the section for the mixing of two track with mix button * Join the track * Save the track edited |
| **Audio record panel** | * Record new audio * Replay the recorded audio * Use FIR filter to audio and replay * Denoise the audio recorded to remove noise * Save the sound with the option of saving normal audio or denoised audio |
| **Tone Generation** | * Manage the level of volume for the tone sample * Select the filter for the keyboard music * Create new tone with the given keyboard * Save the new tone |

## 3.2 Performance Requirements

When it comes to the performance of the system then it is the complex and important attribute for the system. In this world of big competitions between many application systems, performance plays a great role for the success of system solution and it is not about how big are the system functionalities but what performance it has with the available functionalities. In order to ensure the system performance following key points are considered while developing applications.

#### 3.2.1 Speed

Generally, speed refers to the processing speed of the system that how fast it can load to new panel and how fast the access of data is so that it can lead to increase in system efficiency . The factors affecting speed in system are define below.

1. **Response time**

The designed system is expected to respond within less than 2 seconds if it meets all the requirements like device performance and the memory available to run for the system. To estimate the system actual performance speed, system is run at several servers as pilot system.

1. **Throughput**

Normally the system is run by single user but the system can be opened multiple for multitasking and for this case , case study have been done and implemented system in such a way that it can handle couples of tasks at a same time within the expected time for processing.

#### 3.2.2 Capacity

System is designed to be run individually which means that the capacity of the system is defined by the capacity of the sever on which it runs and normally it will be the computer of the company. Therefor the capacity can be defined by the user.

#### 3.2.3 Reliability

For this solution system, it is more reliable due to its development for individually sever so that it can be crashes only if the system goes overload with many events which is rare case but I that case, it will not affect the data they are working with and users can access the system anytime they want as they will not be covered with boundaries.

#### 3.2.4 Usability

Considering the fact that the users who are going to see maybe unskilled too so that the system user interface is designed accordingly with the preference of un-skill users so that they will be able to distinguish every navigation, buttons or diagrams with certain description about them by test or images. Whereas the color combination for system is designed eye friendly and common so that it will not affect users while long run. To make system less complex it is dived into few section which is all controlled by main panel.

### 3.3 Design Constraints

Client have given less specification for the system development regarding postproduction company which adds voice commentary for a pre-recorded football match. Since they are working for adding two audio files, they concluded that the system should allow the user to combine two audio with the feature of importing files and then mixing them together at specific section of main audio file which is then to be exported to the destination directory. They have suggested to implement more functionalities for the system according to the choice of developer but need to be relevant to the topic given.

### 3.4 Commercial Constraints

Since, company have not shared about the requirement attributes, following strategies will be evaluated by the developer for test cases after the completion of system development which is the showed to the company.

Table 4 : Acceptance Performance Tests

|  |  |
| --- | --- |
| Performance test for system | Strategy |
| Multi-Tasking | * Performance for system depends on the number of tasks the system doing * For multiple task response, testing is done |
| Throughput Speed | * It is for the purpose of calculating the rate of response by the system for event. * Allowing system evaluation with selected person to justify for this case which will be shown in evaluation. |
| Response Time | * Response time is the rate for the throughput for certain events . * It is also evaluated during evaluation with selected candidate. |
| Capacity | * For this system capacity depends upon the sever so that the server is responsible for the storage of system files and event files which means it is relatively higher. |
| Reliability | * It defines the smoothness of the system events * Testing is done through the evaluation which is justified by user’s response. |

# 4 System Analysis & Design

## 4.1 Preliminary Design Stages

In this section, the first design for the system will be defined through textual analysis so that system attributes and events can be known for proper workflow.

### 4.1.1 Significant Event Analysis

Table 5 : System event analysis

|  |  |
| --- | --- |
| Event | Attributes Involved |
| Mixing Audios | Track 1  Track 2  Selection time |
| Merge Audios | Track 1  Track 2  Starting time of sample  Ending time of sample |
| Reverse Audio | Track 1/ Track 2 |
| Record Audio | Temp Audio File |
| Tone Generation | Temp Tone  Volume slider  Keyboard |
| Trim Sound | Track 1/ Track 2  Starting time  Ending time |

## 4.2 Detailed Static System Designs

### 4.2.1 Class diagram

UML Class diagram are the graphical representation or simply blueprint for object-oriented system where it helps to visualize the object-oriented system and construct it. Below figure contains the class, attributes and functions used with their respective relationship which helps to understand system more without complexity.

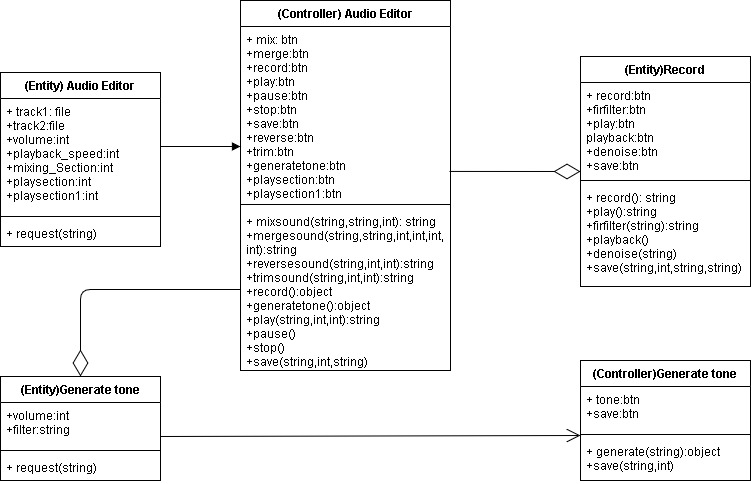


Figure 6 : System class diagram

### 4.2.2 Use case diagram

Use case diagram are the behavioral diagrams for a system through which it indicates every possible event and the workflow by the individual user for the success of respective event considering action taken for them. Below diagram properly shows the system events and actions by actor.

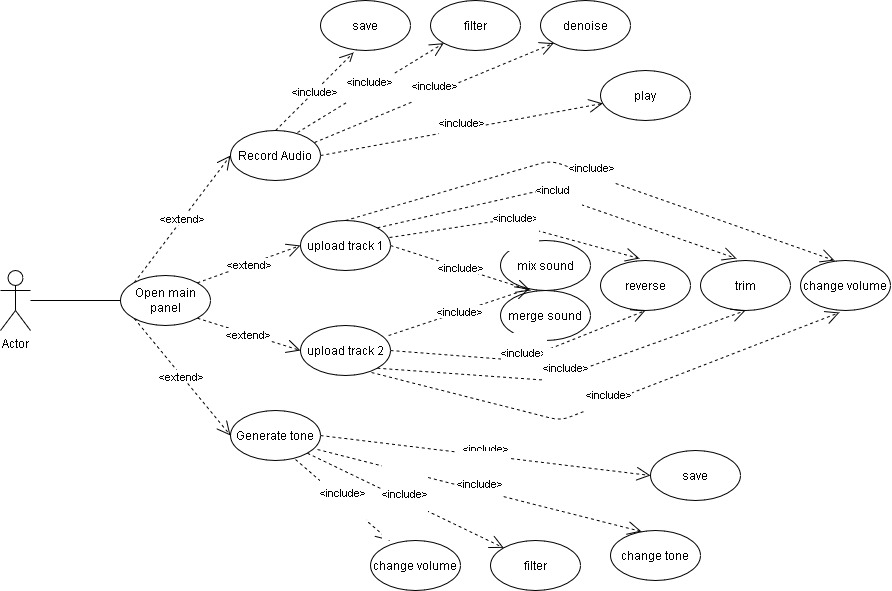


Figure 7 : System use case diagram

## 4.3 Detailed Dynamic System Designs

### 4.3.1 Dynamic Diagrams

|  |
| --- |
| **Scenario: Mixing / Overlapping Audios** |
| 1. User starts the system 2. User selects the track 1 and upload audios file 3. User selects the track 2 and uploads another audio file 4. User selects track 1 as main track 5. User selects the section where to overlap track 2 6. User clicks the mix button 7. The axes change and user play the track 1 with new sample |

|  |
| --- |
| **Scenario: Recording Audios** |
| 1. User starts the system 2. User clicks on record new audio 3. User is redirected new GUI 4. User clicks on record audio 5. User speaks and stop the recording after few instances of time 6. User replays the audio and hear it 7. User filter the audio with FIR effect and listen with replay 8. User denoise the audio and hear it 9. User clicks the save button 10. User gets options either to save normal audio or denoised audio 11. User selects the option save denoise audio 12. Event ends |

# 5 System Interface Designs

System interface design gives the insight view of the solution system before the actual developments to know how the system will work and how the system is to be developed during the implementation carrying different conditions and possible solution for suitable events. This section consists of two parts where one part defines the layout for the system where another part justifies for the system events or simply system workflow.

## 5.1 Draft Interface Designs

### 5.1.1 System Wireframe

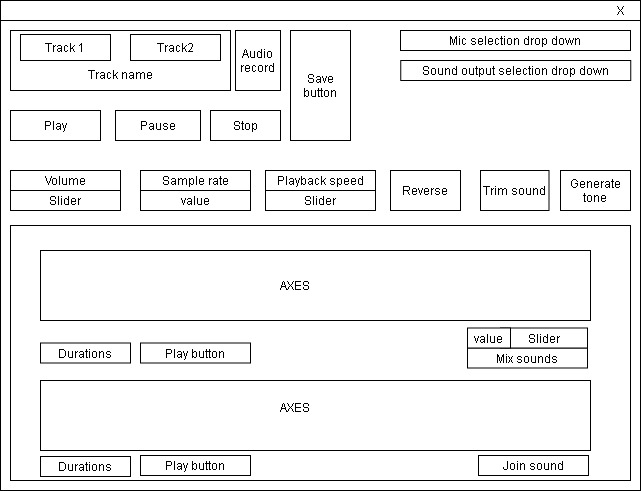
System wireframes are the initial phase for designing the system layout which makes the actual development of system more ease. Below are the wireframes justifying the planned layout for the solution system.

Figure 8 : System main GUI wireframe

Above is the wireframe for the system main GUI which is same for every event beside tone generation and audio recording where the slider may change the position, the axes can be changed according to frequency and the input field can be changed.

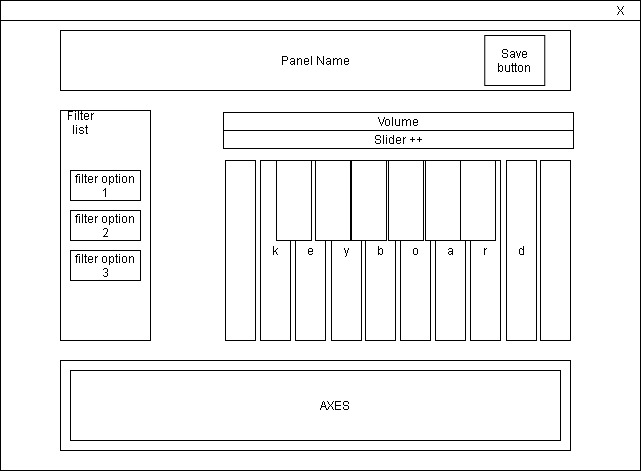


Figure 9 : Tone Generation GUI wireframe

Above is the wireframe for tone generation which will be the attached with main GUI and the axes will be changed accordingly asper the tone of keyboard and other will remain same beside the movement of slider for volume.

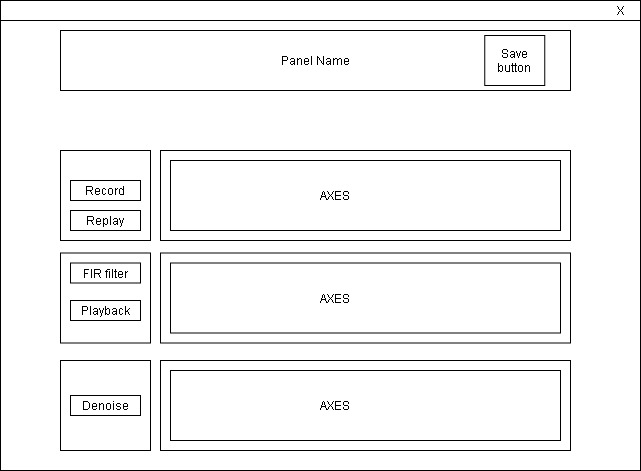


Figure 10 : Audio Recording GUI wireframe

Above is the wireframe for audio recording GUI whereas same as other GUI axes will be changed according to audio recorded and filtered where the buttons are same and will trigger certain events in the left 3 panel. The first panel contains header nae and the save button for this system.

### 5.1.2 System mock-up

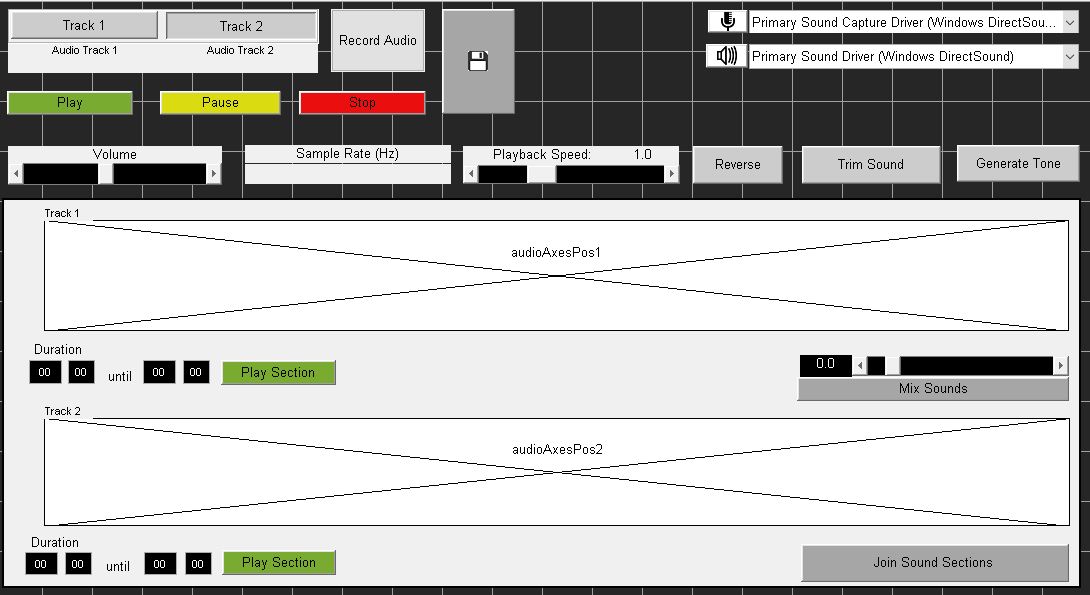


Figure 11 : System main GUI mockup

In the above mockup, the figure is about the main GUI which connects two more GUI for the proper functionalities of system where each of them is independent. The GUI consist of several buttons for triggering events and two axes for the plotting of audio frequencies which gives inside view of system event acting to tracks. The mockup contains button like record an generate tone which will trigger for new GUI above the main GUI. All the buttons in the system are designed with dark gray color to remember and the slider and given foreground with black color whereas the drop-down buttons with white background with couple of clickables drag lists. The main button here are the two tracks and then the play, pause, stop function and the basic functionality required by client which is Mix Sounds button with the slider of selecting certain section and then the join sound button which is extra.

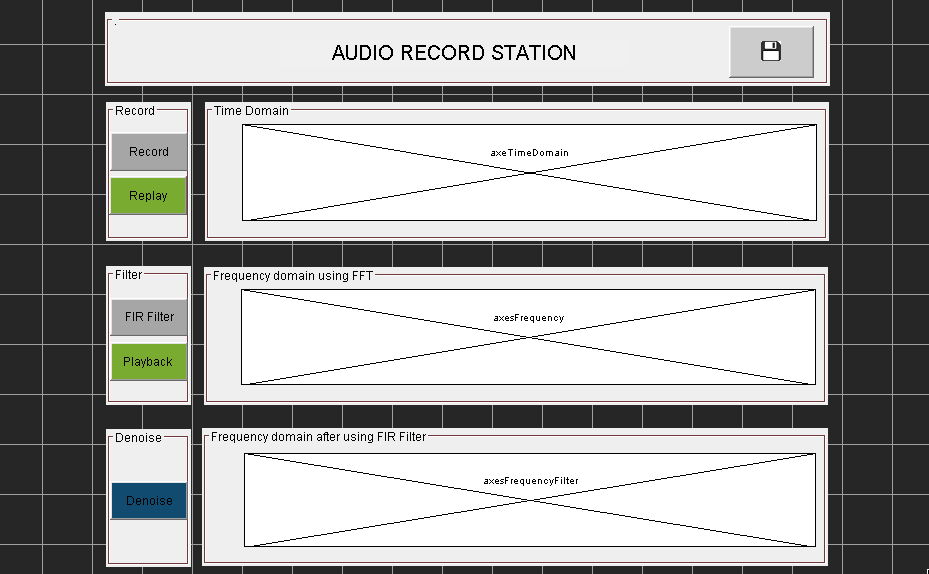


Figure 12 : Audio Record GUI mockup

This is the attached GUI for the system implemented to edit audio files. This GUI is about audio recording which then can be filtered accordingly. The axes are for the plotting of frequencies of sound where the small three boxes with buttons are the events triggers for this GUI and in the above panel beside panel title, it is a button to save file recorded.

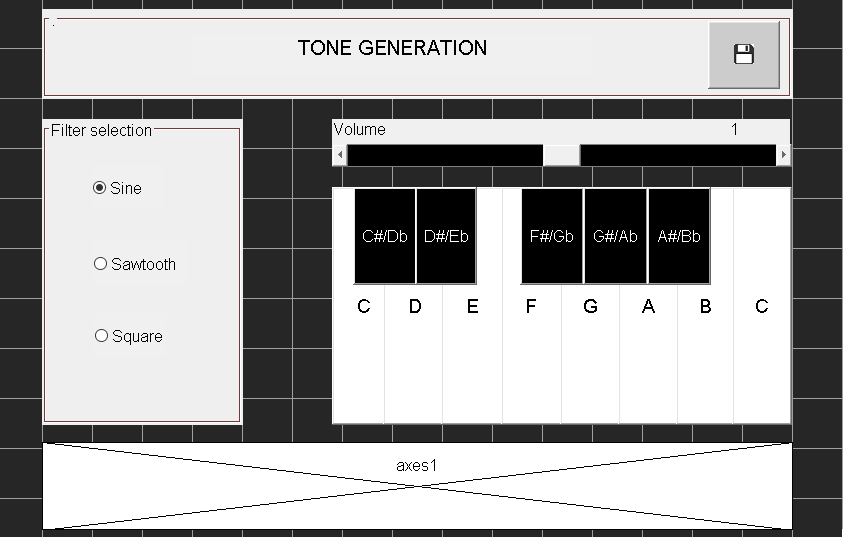


Figure 13 : Tone Generation GUI

This is the another attached GUI which is triggered after clicking a button in main GUI and this GUI consist of keyboard which produces tones and can be filtered accordingly with the help of filter selection buttons in the left panel. The axes panel plots frequencies and the volume slider maintain volume. The above heading panel consist panel title and the save button for the tone generated.

### 5.1.3 System Navigation and Event Diagram

In this section, the navigation for this system and workflow is designed with the flowchart which consist of menu bar options and buttons involved in the GUI which are responsible for certain event trigger in the system by the action of user.

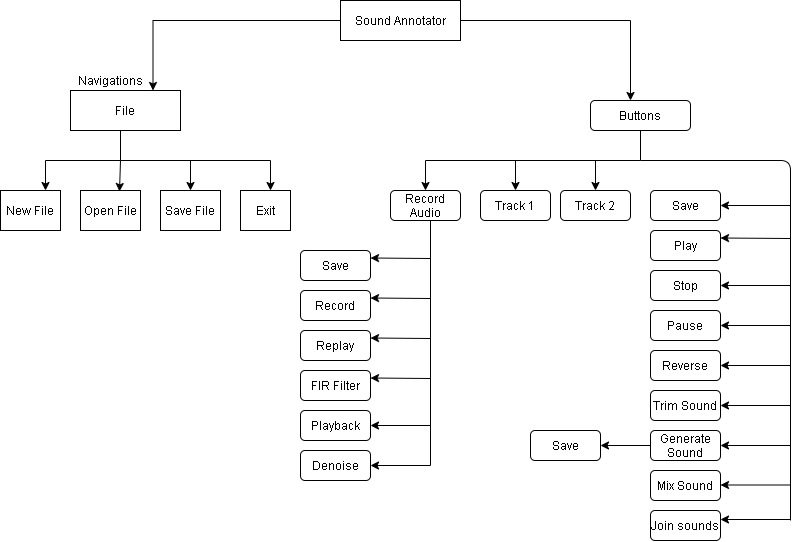


Figure 14 : System Navigation and Buttons flowchart

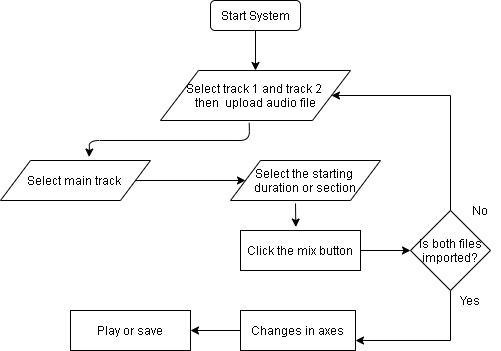


Figure 15 : Mixing or Overlapping two audio flowcharts

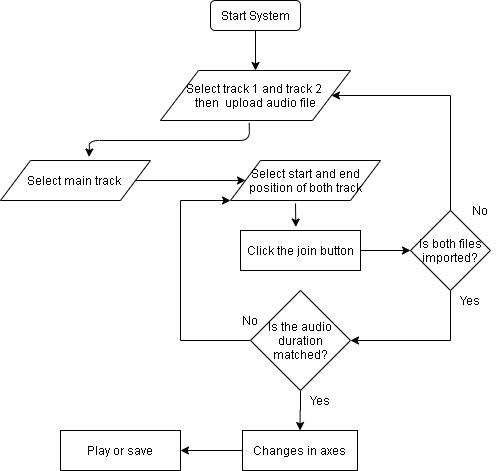


Figure 16 : Merging Audio Flowchart

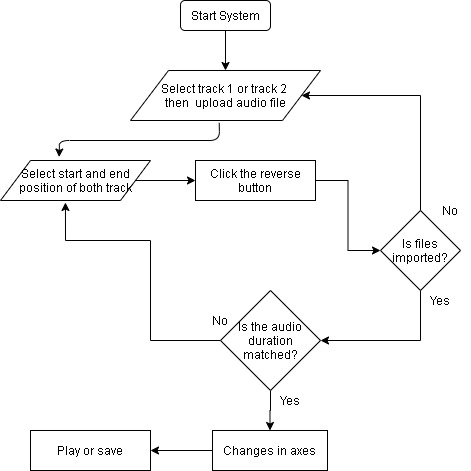


Figure 17 : Reverse Audio Flowchart

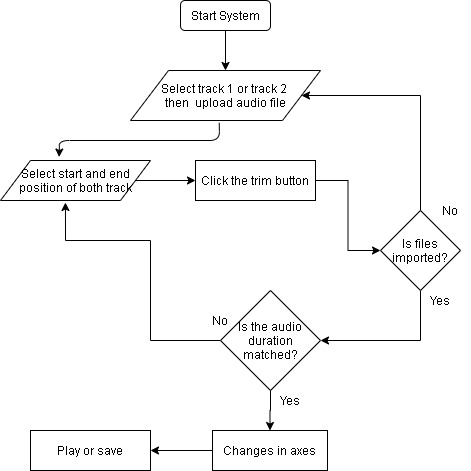


Figure 18 : Audio Trim Flowchart

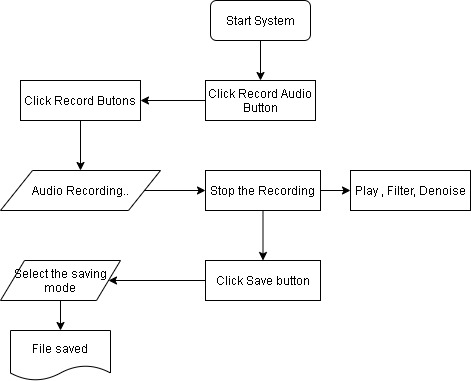


Figure 19 : Record Audio Flowchart

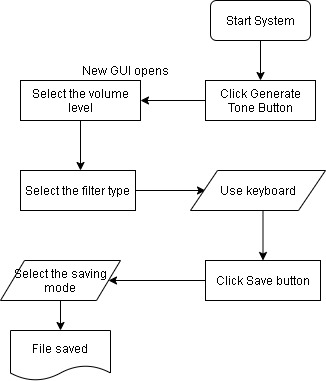


Figure 20 : Tone Generation Flowchart

# 6 Design Revisions

For the designing of GUI for the system, the first step was to create the system graphical notation so that system functionalities and event can be discovered after which wireframes are built after knowing the common attributes for the system designing and then with the help of wireframe system mock-ups were prepared upon which the system development for designing is to be carried and the system GUI was designed accordingly upon which for the work flow of the system it was consulted with the event diagram and workflow of system with the help of functional requirement.

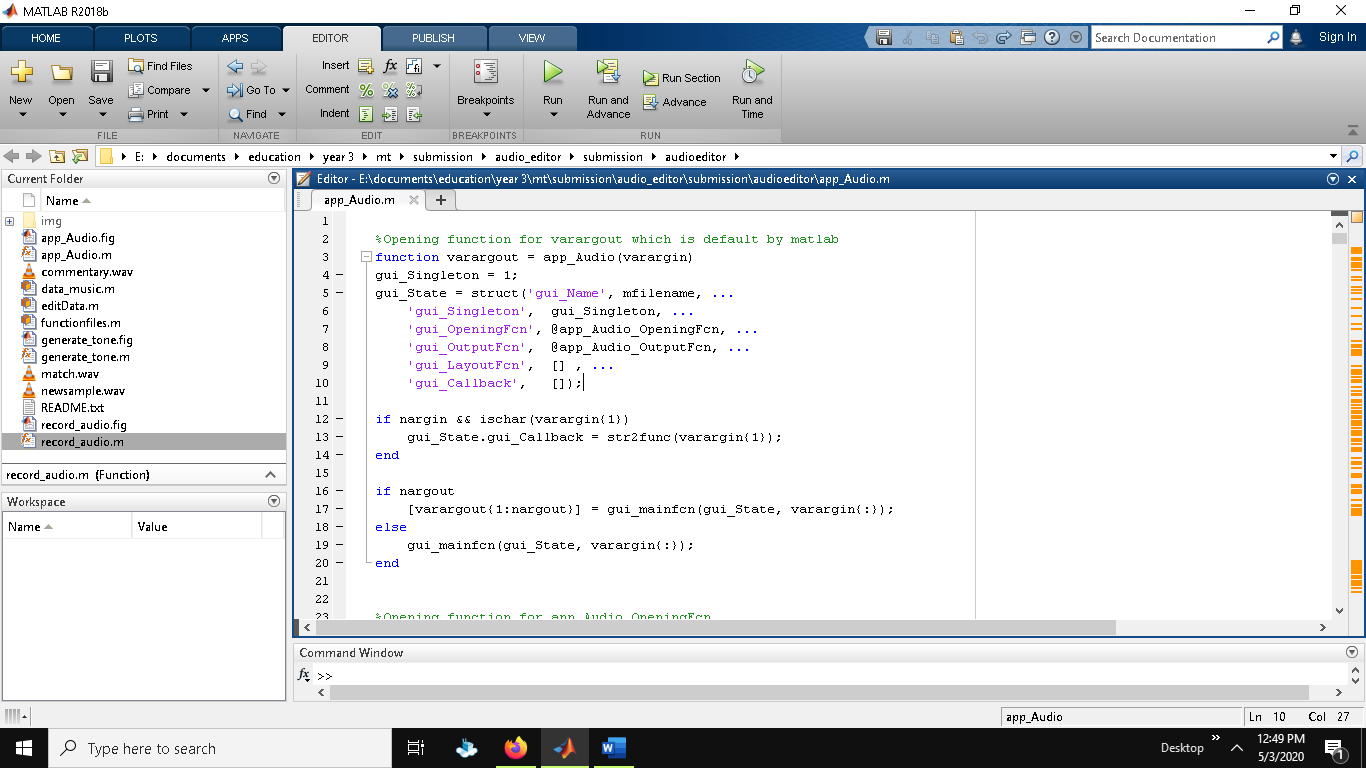
First the main GUI was created upon which the given requirement by client was completed for mixing two sound and then several other events were carried on the same panel and then two buttons for triggering new GUI were placed. Afterwards, GUI for audio recording and Tone generation is carried accordingly with the reference of system draft designs and the attributes collected with ensure every event takes place correctly.

# 

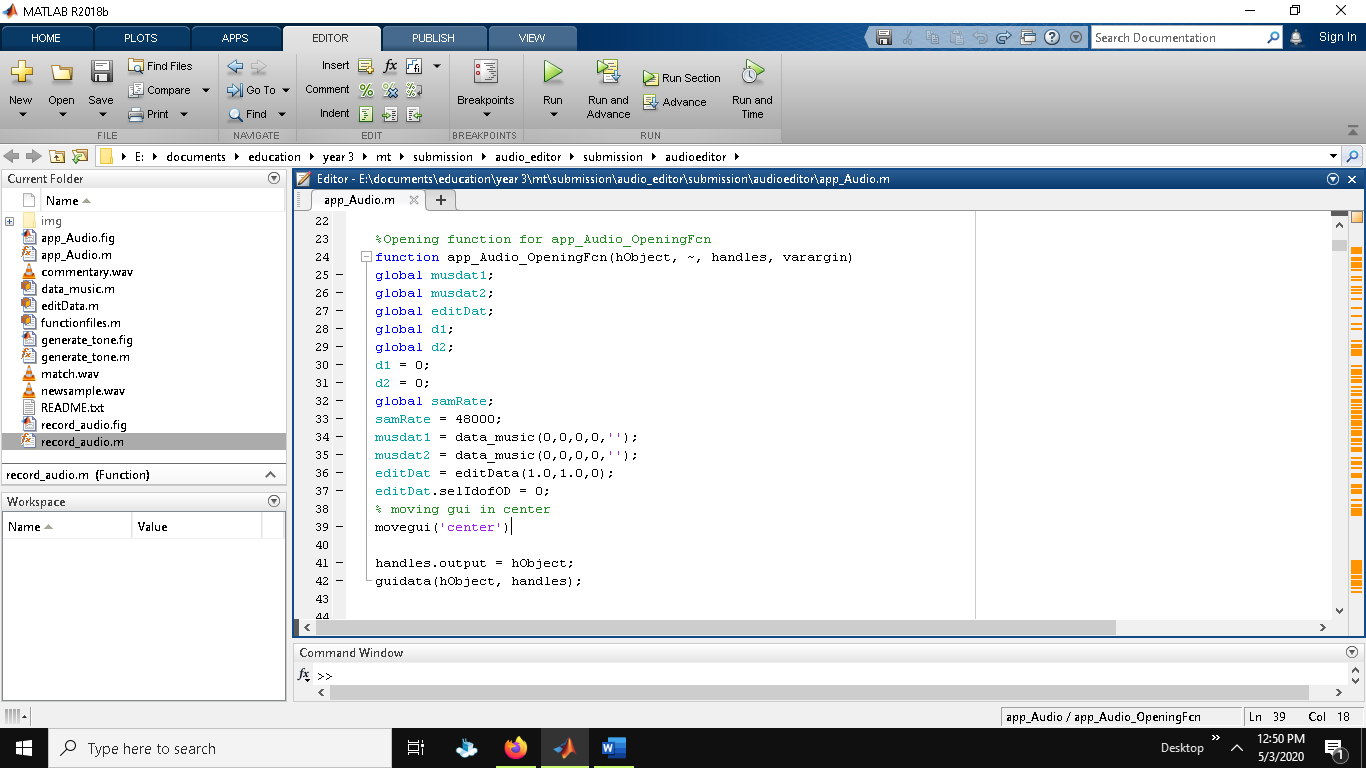
# 7 System Build and Technical Notes

With the completion of system requirement engineering and design finalisation for system workflow and events, system is developed with proper standard code. In the to make the system less complex to understand, 3 GUI is developed so that if anyone want to look through the code then, they can knot the what actually the lines defines and for every possible new functions comments are given. For the purpose of getting deep knowledge about MATLAB raw codes are cited from (ittus/Matlab-Voice-Record, 2020) ,(priichan/Audio-File-Editor, 2020) and (Andor, 2016) so that developer can have very insight view about MATLAB and develop according to his idea to implement the existing system function according to the need of system rarely for some cases.

The system files consist of 9 files where 3 of them are .fig files for each 3 GUIs and 3 are their respective .m files system is designed with MATLAB app designer. For making less complex structure, main GUI is linked with 3 more files which are for editor data ( properties defined here are volume, playback Speed, outputDeviceSelId, musicData) ,useful functions for every event triggering and then music data file for storing the information about the music data like the sound stream, total sample and samples rate.

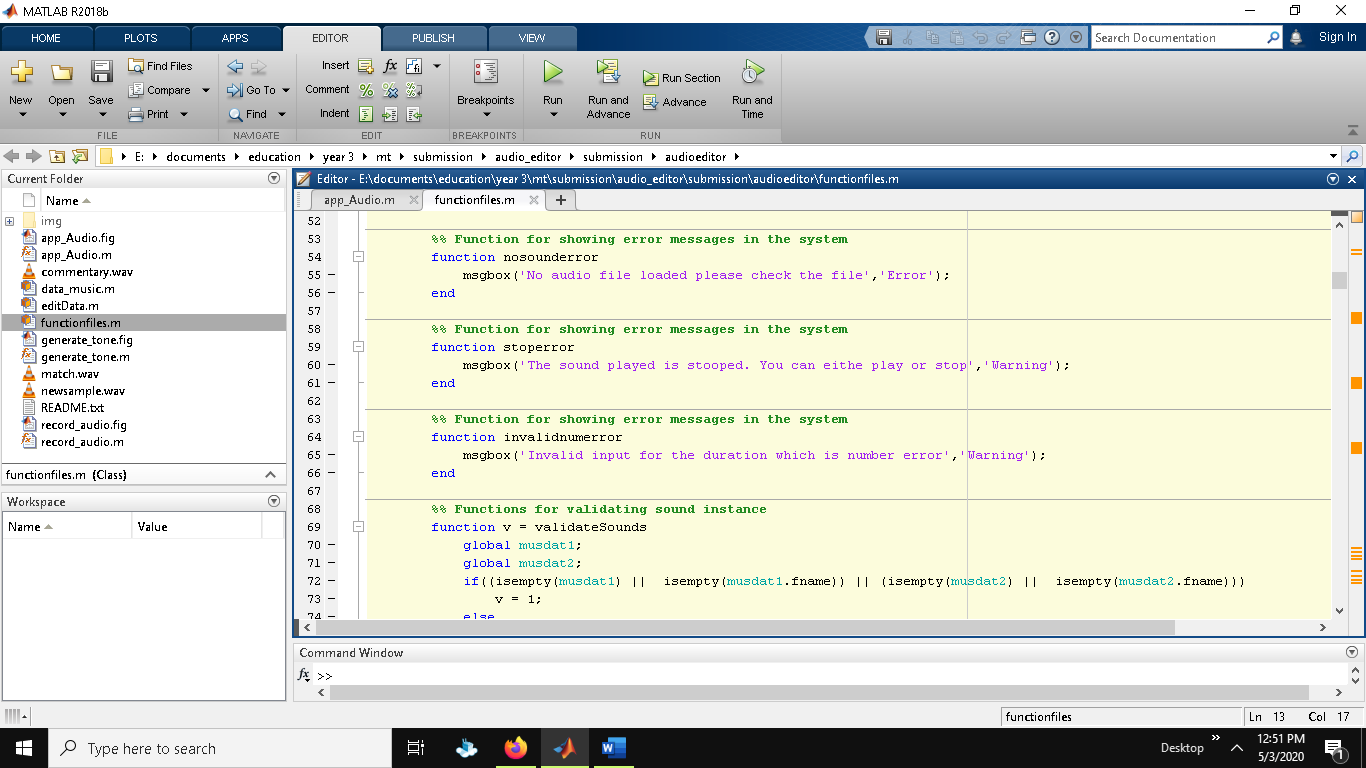
Below are the functions and classes with each variables role explained detailly which are in use for this system implementation. Below section starts with detailing from the main GUI and then audio recording and tone generation GUI.

Above is the varargout function for this system which is overall same for all the 3 GUIs beside the name of the functions for opening, closing. This function is automatically generated when we design system with MATLAB app designer which will be use while opening and calling during GUI call.

Below is the opening function for main GUI where it will create entry for several global variables with their own purpose where musicdata1 and 2 are for the purpose of retrieving the data of track uploaded and same for another global variable data1 and 2 which will be use later to overlap the audio by temporarily adding zeros to the section for matching array dimension of audios. The global variable sample rate is defined 48000 for now which is a typical waveform for recording and using audio sounds. The movegui(‘center’) function will let the GUI to open in center of window and all the object data will be assisted by handles.

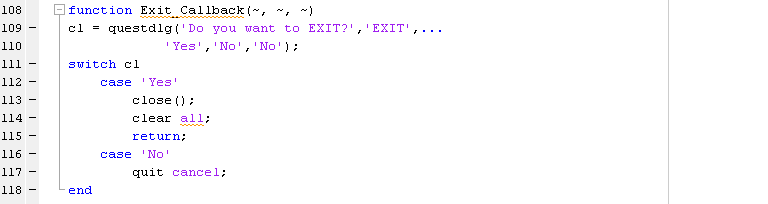
## System events and class attributes formation

**Error function**

Functions to show dialog box for system errors are same for all with the use of msgbox which contains string inside it.

**Closing function**

Below is the closing function for GUI which uses switch cases either to exit or not and respond accordingly which can be done by deleting object too. The function is same for all the 3 GUI for closing.

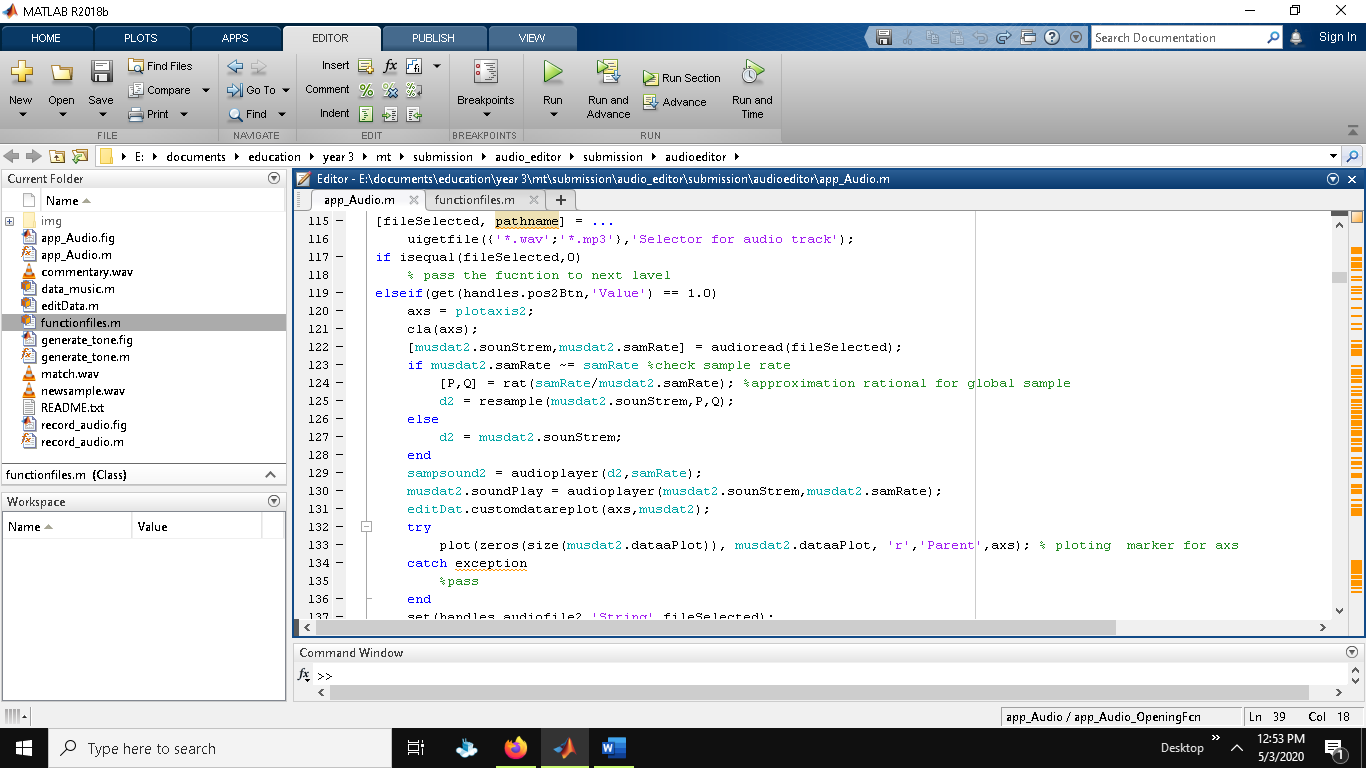
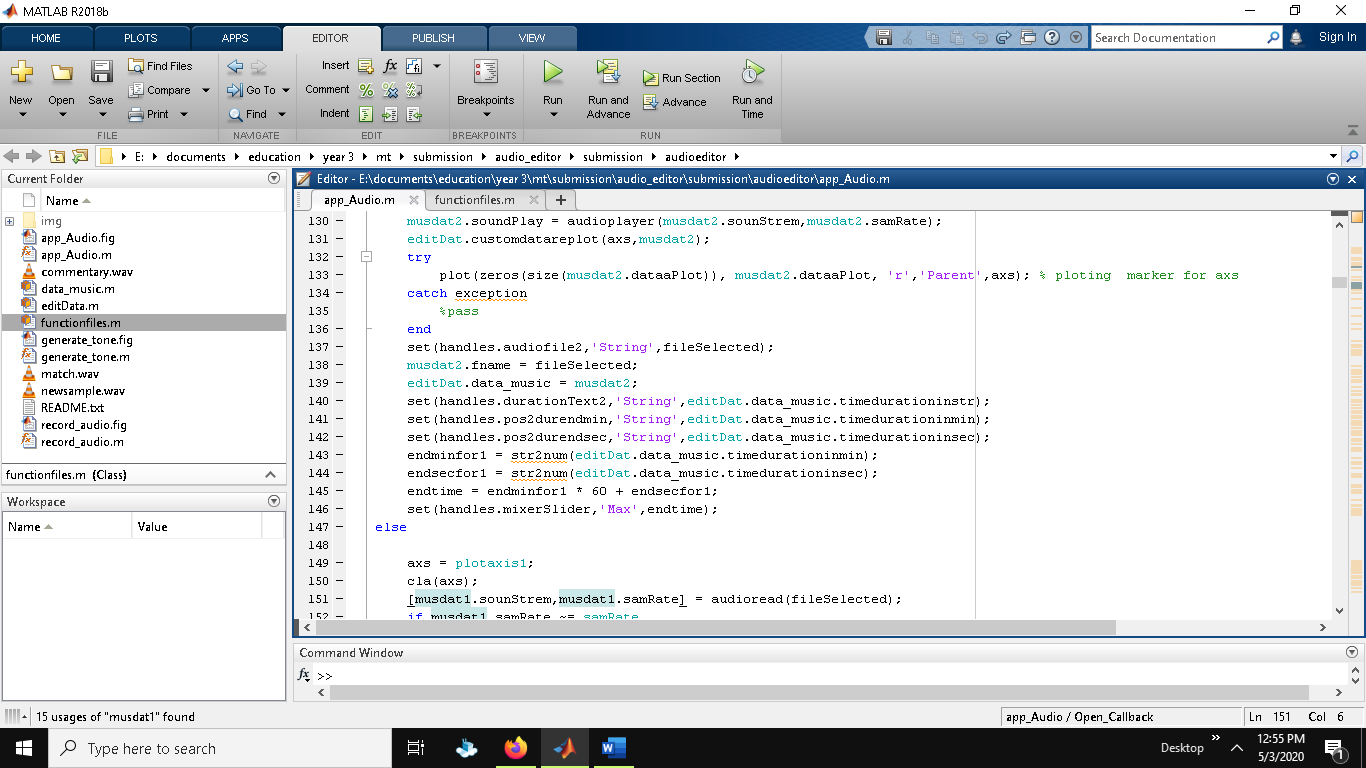


**File opening function**

After the selection of file to upload either track 1 or track 2 the opening function is called where at first all the global variables describe above are called so that data can be given for each. At first file selected and the pathname is decided and currently wav and mp3 file are allowed which then on opening of file it is checked that either file is selected or not upon the selection of that there is another conditional statement given to check either the track 1 is selected or track 2 according to button value.

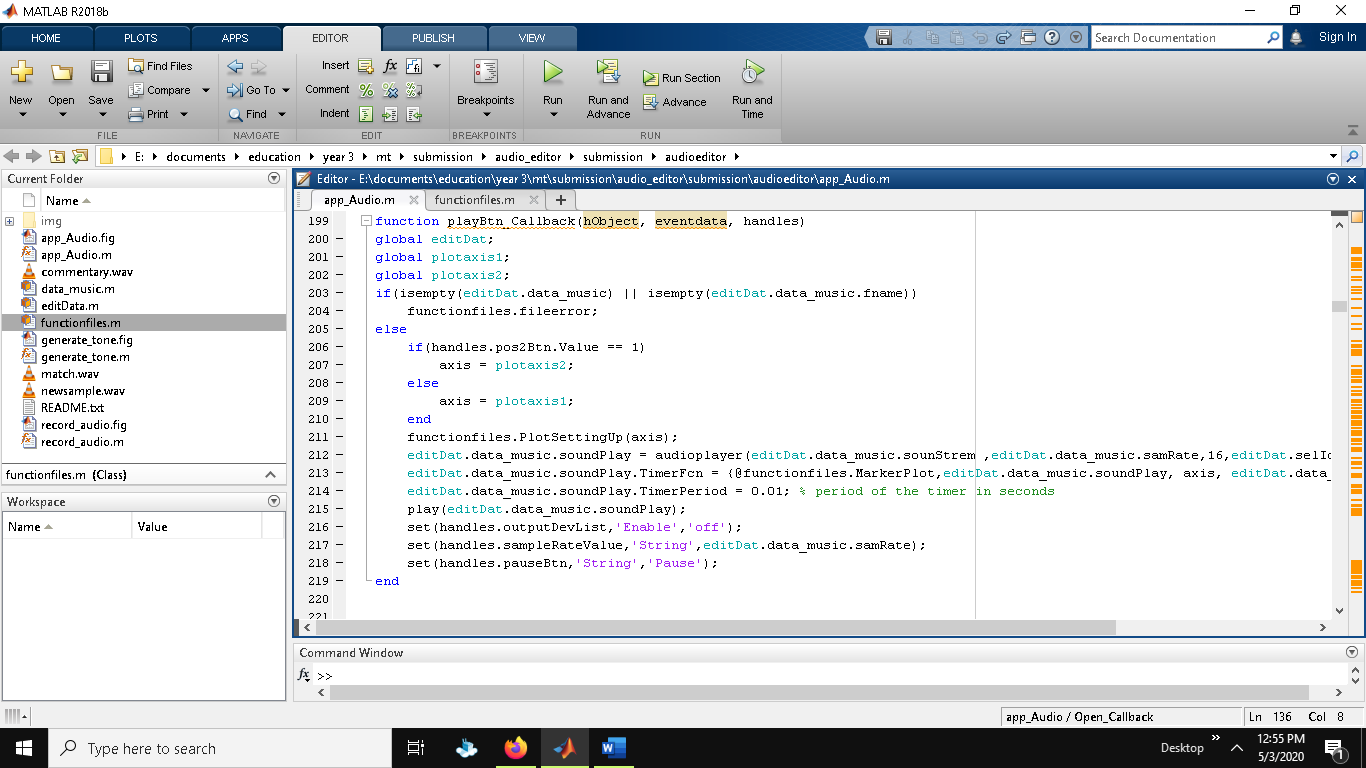
When the button position matched then data is plotted to every global variable with the help of audioread fucntion which is directly assigned to sound stream and sample rate of music data object upon call on which it will give all the information about the sound where there are two musicData object musicData1 and 2 for each track. After reading the audio file the sample is measure and check if the global sample rate matched or not and in case of not matching it will resample the music data with the help of calculating global sample rate by actual data sample rate. After that audioplayer function is use for global sound variable which will be use during the overlapping of sound later although we can use the previous audioread sample but to make code less complex, global variables are used wisely. Now the music data can be plotted to individual axis and to confirm that disp() function is used to view in command line. With the plotting of axis, all the field for audio data like duration are updated to set() function with the retrieve data.

The process for track 1 and 2 is same beside the variables name where 1 is replaced with 2 like musicData1, musicData2 and sound1 , sound2 etc.



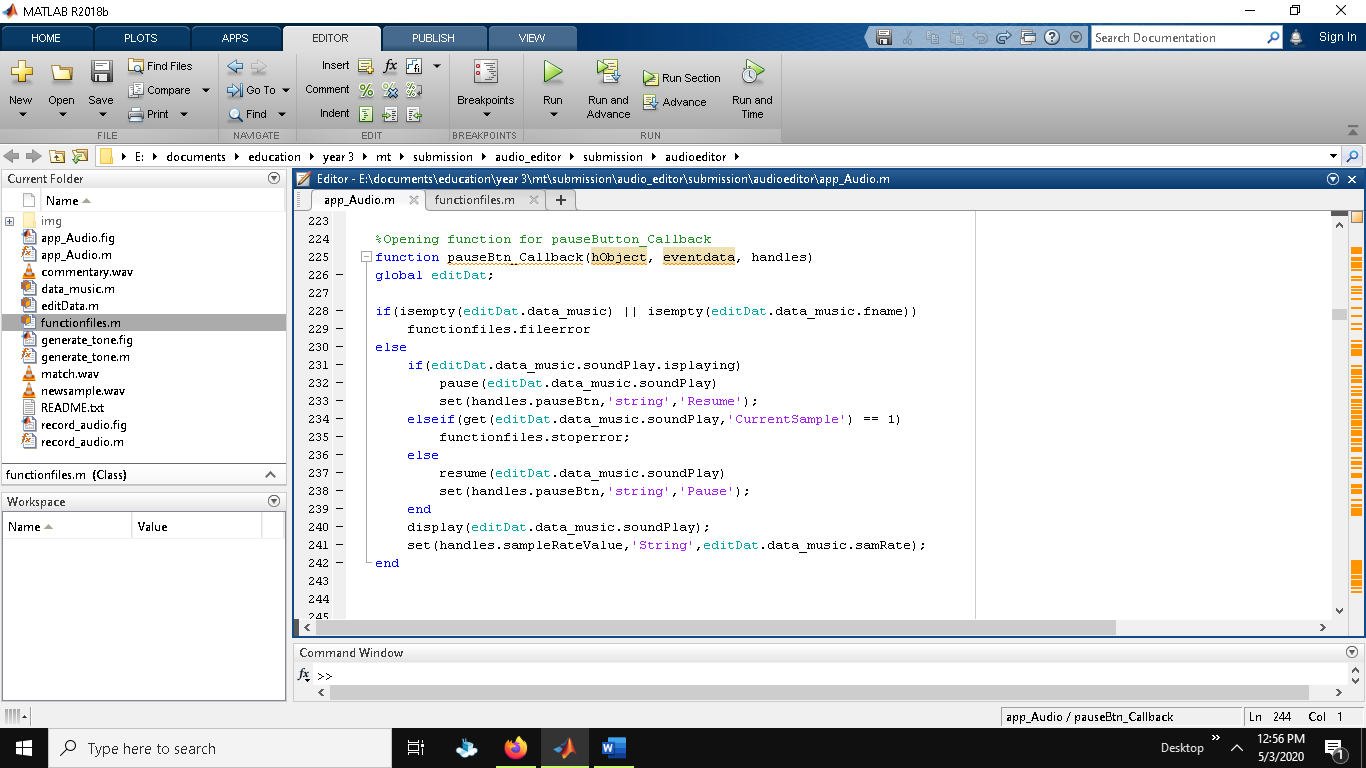
**Play Audio**

To play the audio audioplayer() function is used with the information of music data provided which is determined by the button selected and then matching the axis . before playing audio, it is first checked with conditional statement checking either the music data is empty or not and respond whether to throw error or pass.



**Pause Audio**

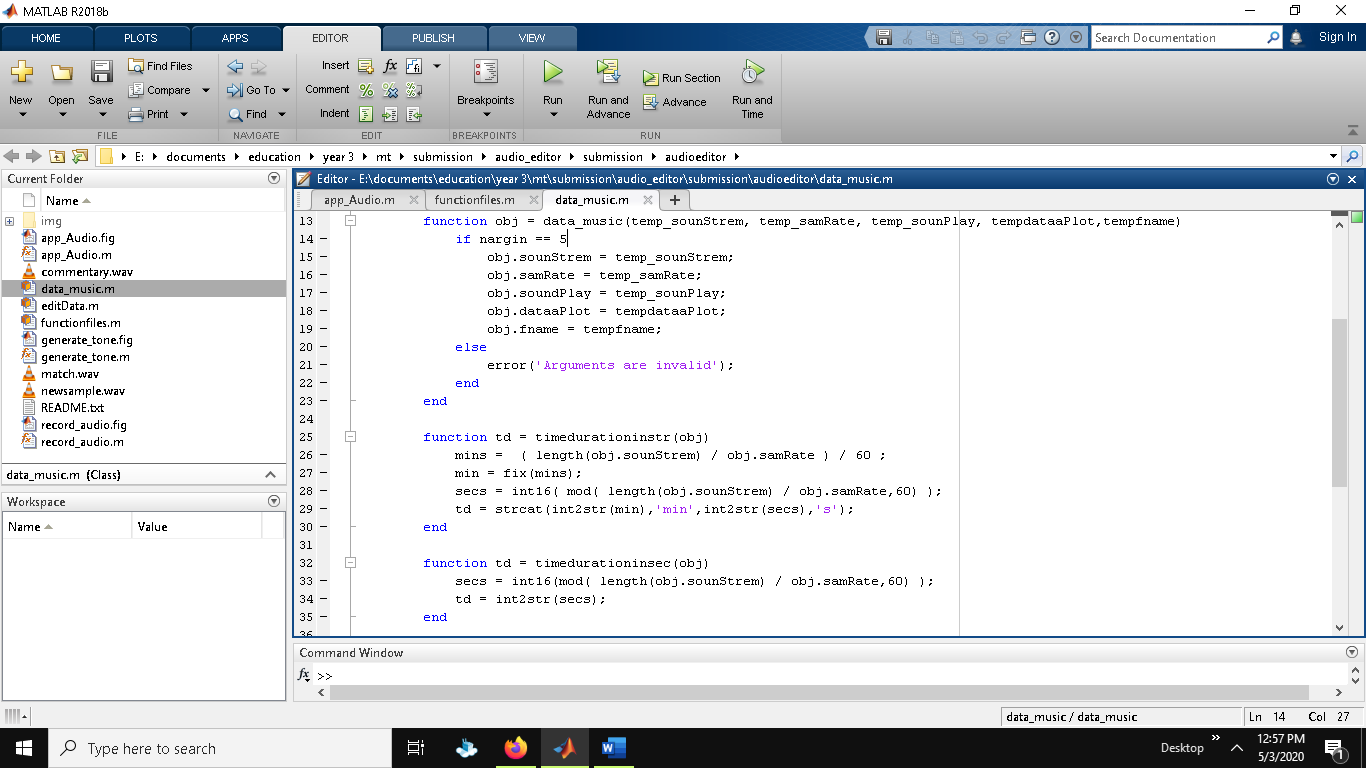
On click of pause button the function verifies the music data either empty or not then pass to next level checking either the audio is playing or not in editor data so that he pause button string is set to resume. If the resume button is clicked similarly it resume the audioplayer with resume function passed to editData.



**Stop audio**

To stop the paying audio, it is as same as the pausing the audio explained above beside plotting of data according to axis provided.

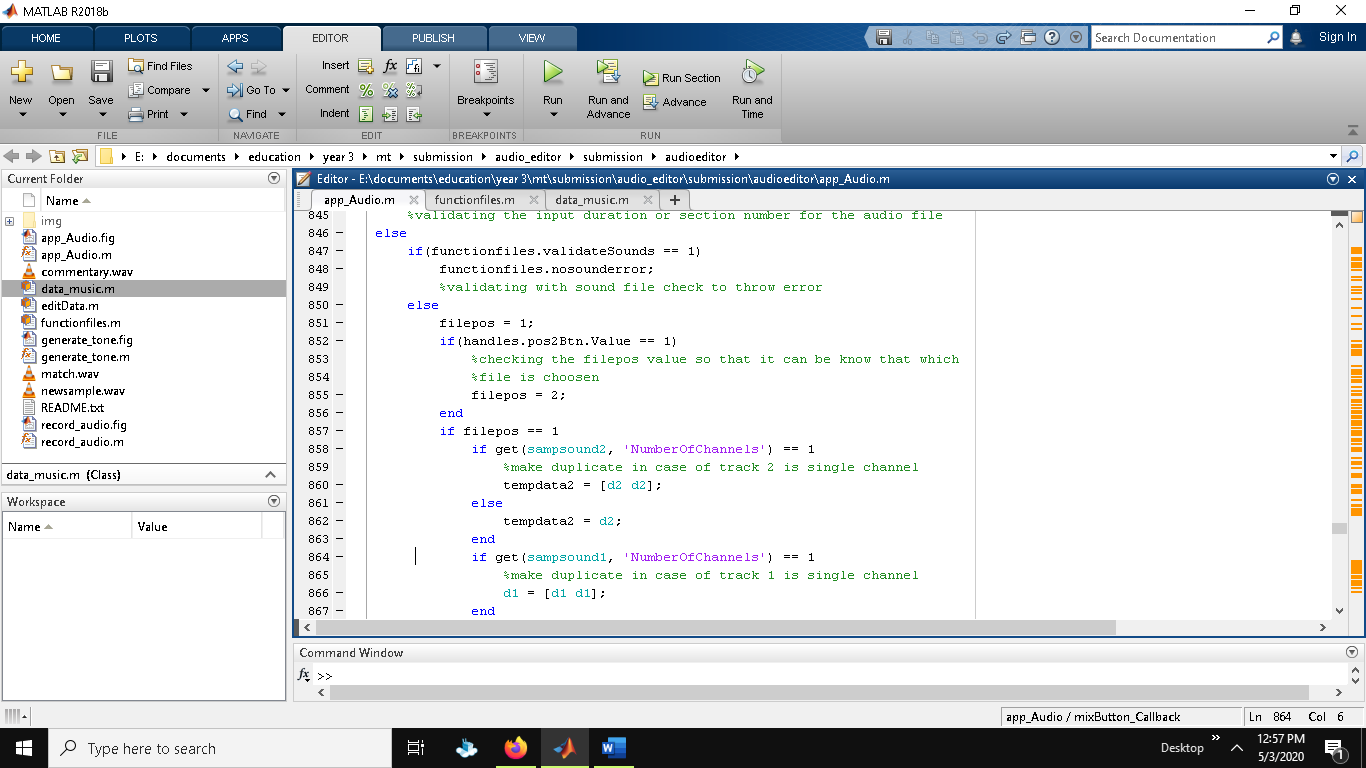
**Data\_music Class**

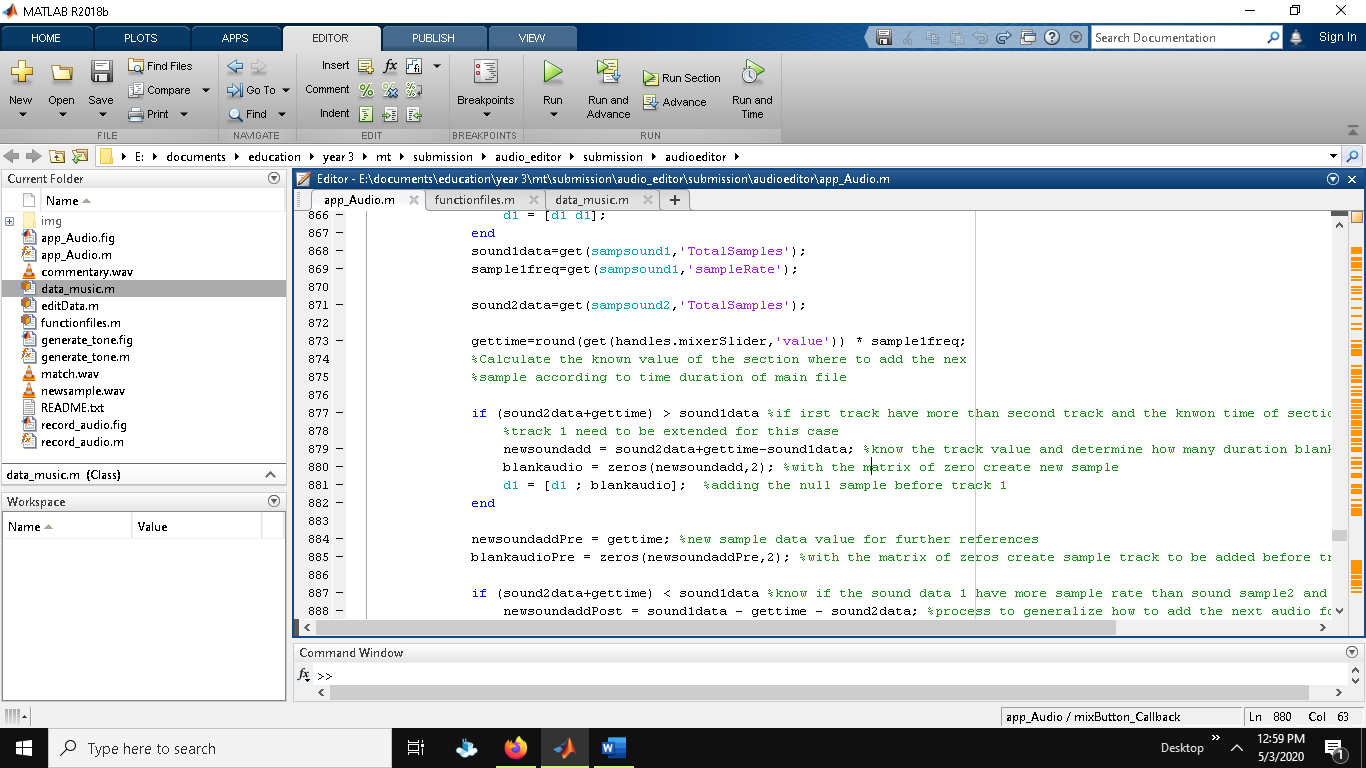
This class contains property like sounStrem, samRate, soundPlay, dataaPlot, filename, duration which are used outside the class to get the sample duration as per request. Below is the function for setting temporary data for properties which then carried out in further section to get the actual duration after knowing the start time and end time of sample audio. Nothing less below functions just contains the calculation done for knowing the duration which can either be second, minutes and then the send data is calculate with the help of different math operators to covert from one format to same format of all the objects data.

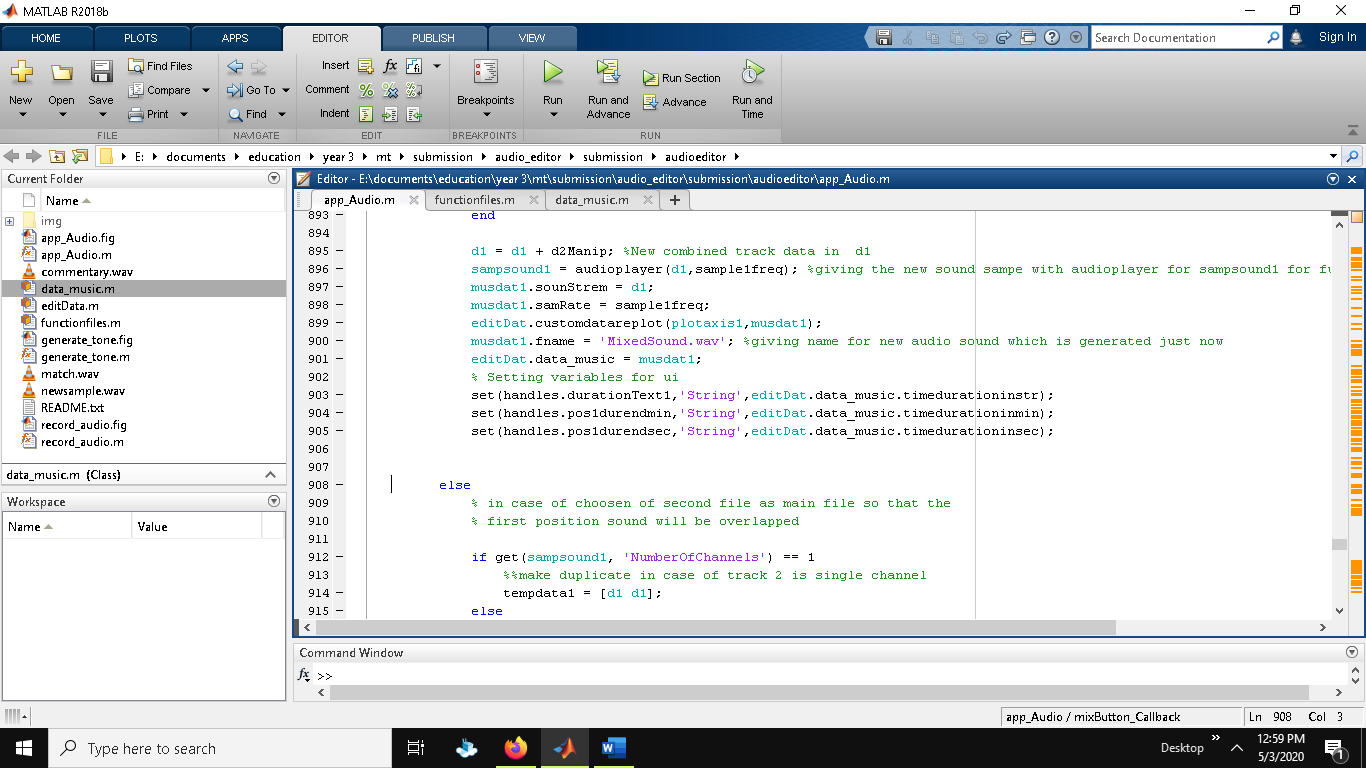
**Mix audios**

Mixing audio is not simple as playing and importing audio as both the samples needs to have matching dimension matrix otherwise it throws error because the audio is not same. In this section of development, it throws many errors normally they were of dimensions due to uploading one big size audio and one small size audio so with the research done for the implementation it is finally solved by knowing the channel of audio and then matching the dimension after which silence is added with the use of zeros for the smaller file for certain time frame.

In the above provide code, at first it is verified either sound is validated or not after which position variable is assigned to know the clicked button so that system will know the main audio and the overlapping audio. With the help of global sound2 which is assigned when uploading file with the help of audioplay, the channels is checked and grouped in array if channel is not equal to 1 for each sound sample. Sn1 and FS1 are the sound samples and frequencies on which the final audio will be set.



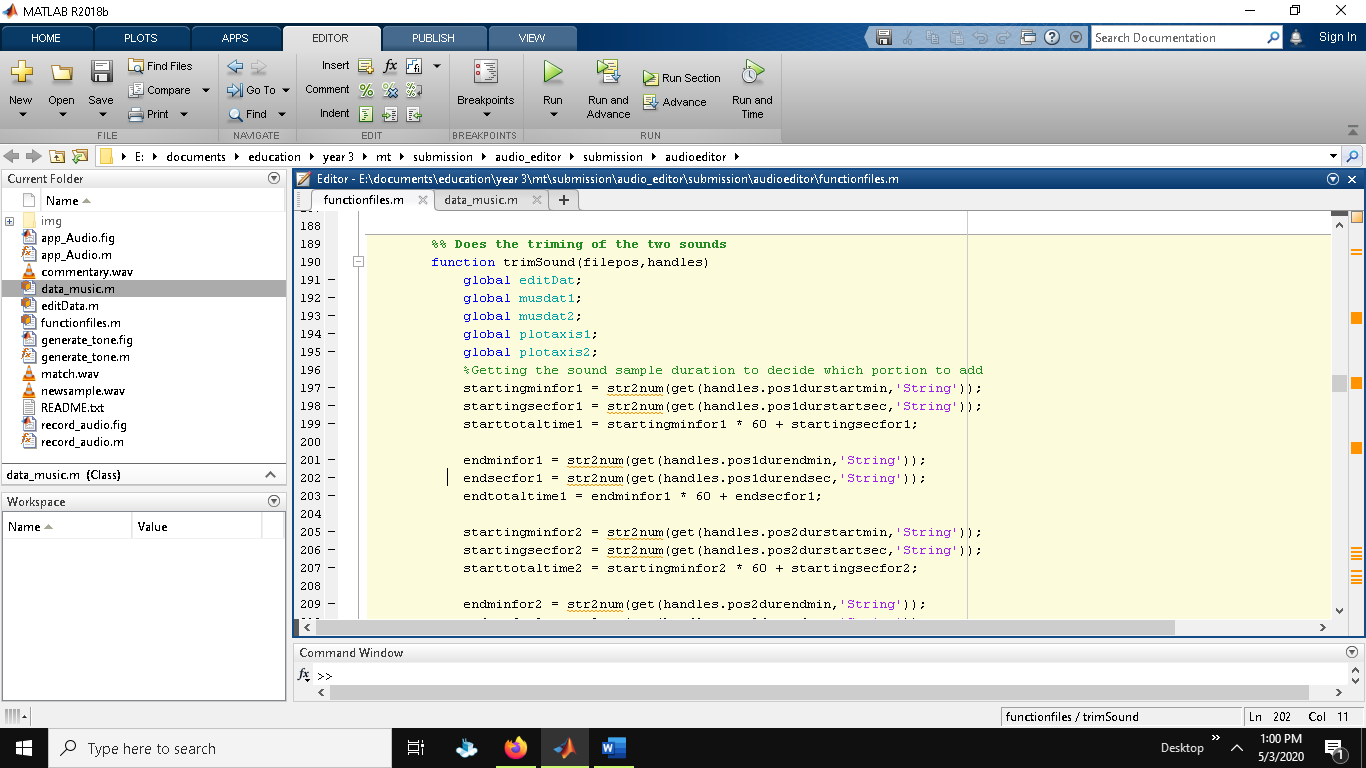
Below is the after code for the implementation of mixing audio. The snd2 variable is the second overlapping sound sample and the insertTime is the slider value for knowing where to start the overlapping of second sound. After this two conditional statement is carried to know either the sum of overlapping sound sample and insert time is greater or smaller than actual main sound so that silence to audio can be added as required at given time with the help of loop and temporary variable.

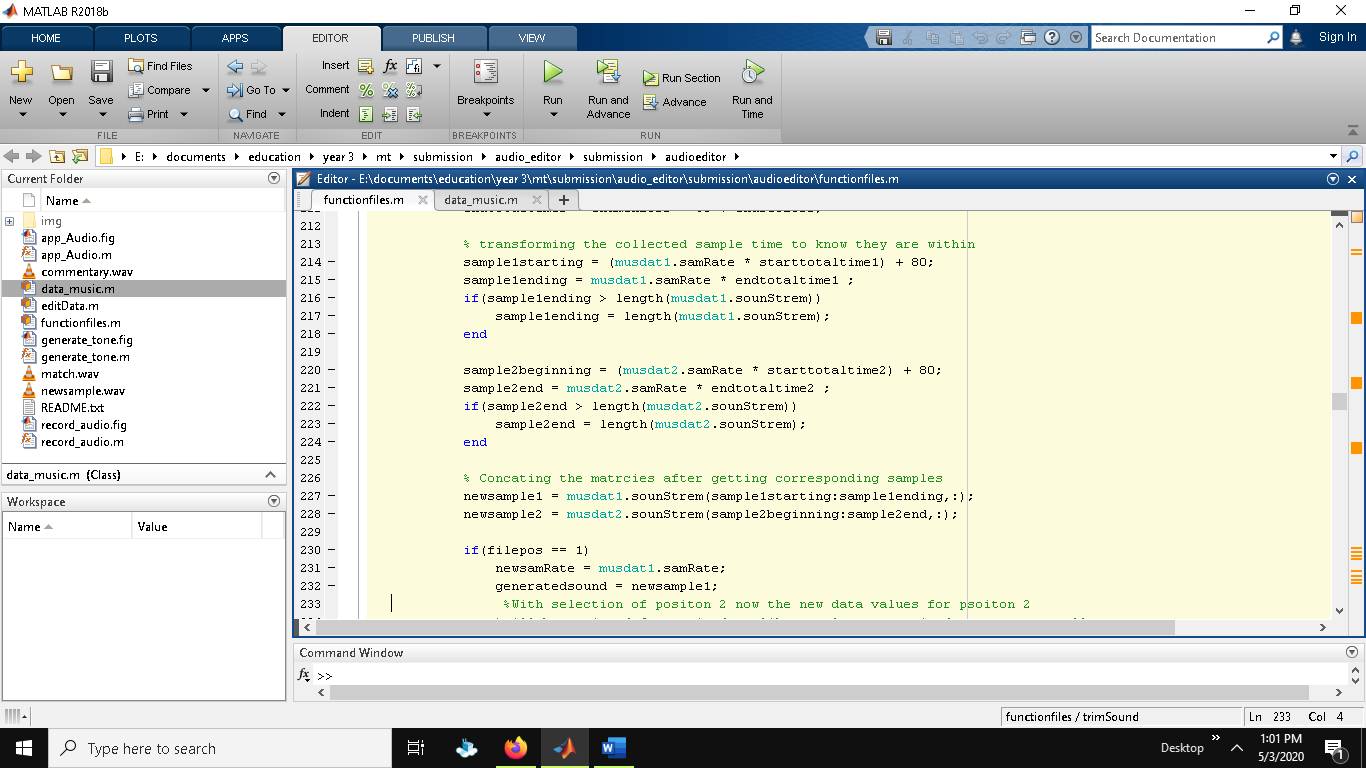
After calculating the required temporary variable for sound 2 it can be now added to main data which is track one and finally data1 is assigned the new data as well as other variabes like sound1 with new sound with same frequency. After this step the other process are to ploting axis and updating the present data in display which is same for all the condition for this system with the use of set method for desired destination.

This process of merging sound for both cases is same beside variable name which ends to 2 if we consider making the second track the main track.

**Trimming Audios**

For trimming useful functions file is also used where from main file calls the useful functions files sending data needed for editing like handles and position. After calling the function below is the code use for trimming audio sound.

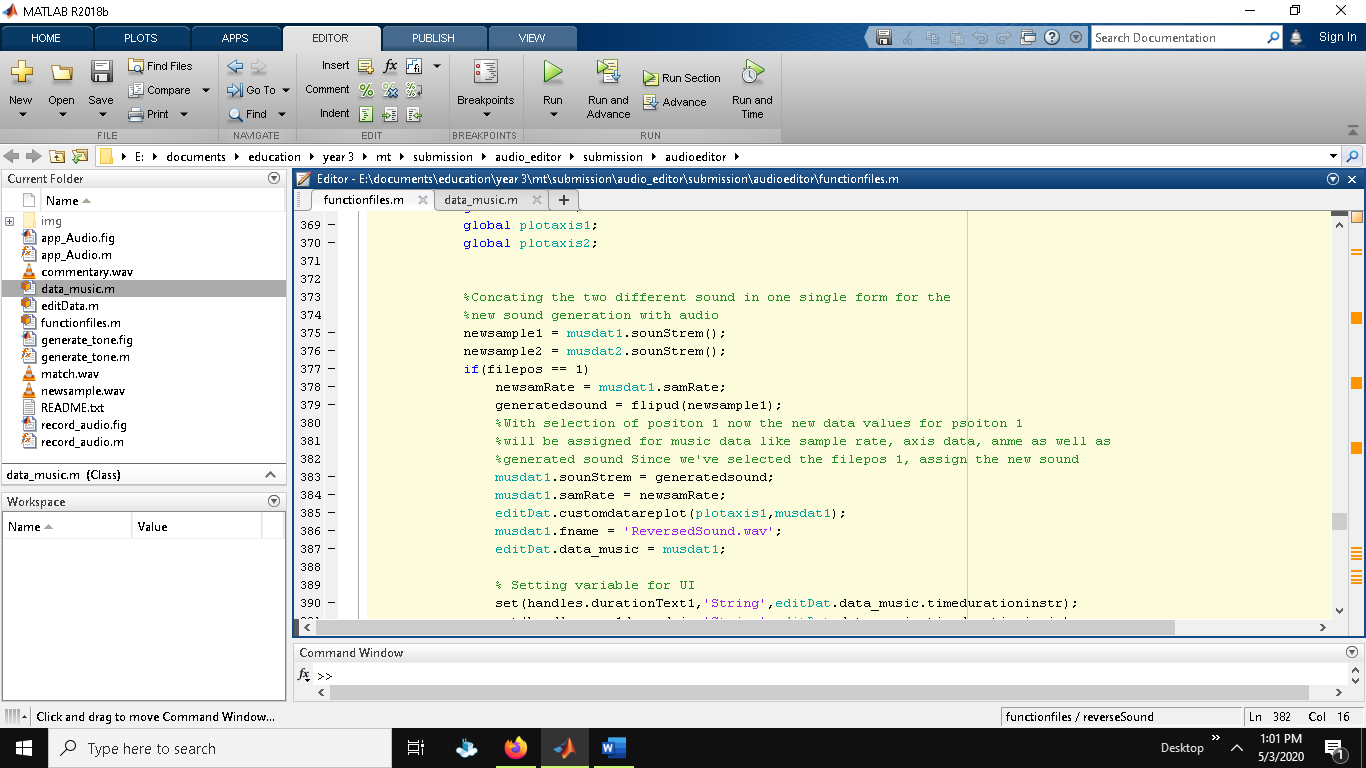


In the above code, at first global variables are declared as like in mixing audio and then the section time that user have inputted in the duration input field is calculated with the string to number conversion method and getting the data with handles use which is then the section of time to trim is calculate in second and further process will carried out which is shown below.

In the above section when the beginning and end sample is calculated it is then use to get the length of the audio file which is the populated to piece1 and piece 2 which is two different track with new sample according to duration provided. After this position is checked where new sample rate and sound is updated to existing sound accordingly. The other process for updating axes, time duration is as same as last part of code shown in mixing the audio .

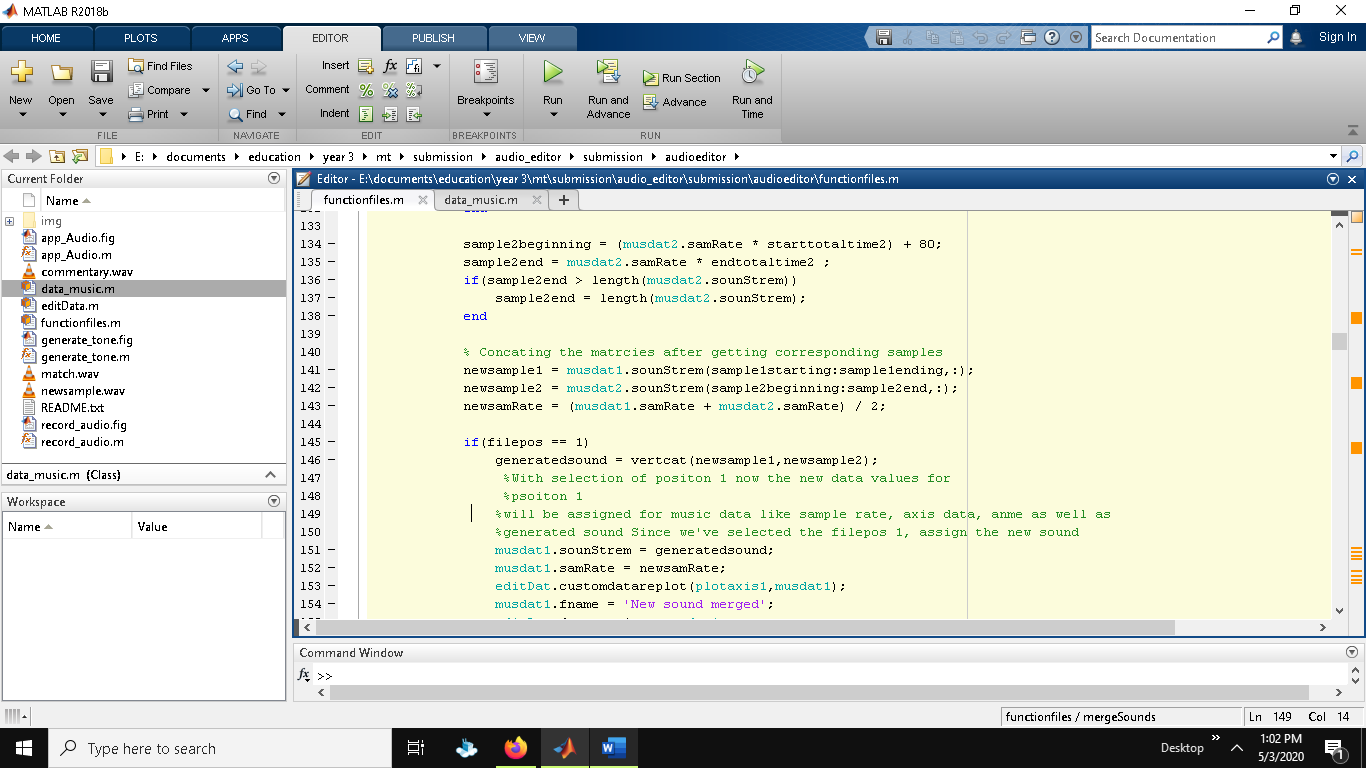
**Reversing audios**

As same as trimming data all the process is carried out till the new sound sample is to be updated where use of flipud() function is used to reverse audio.

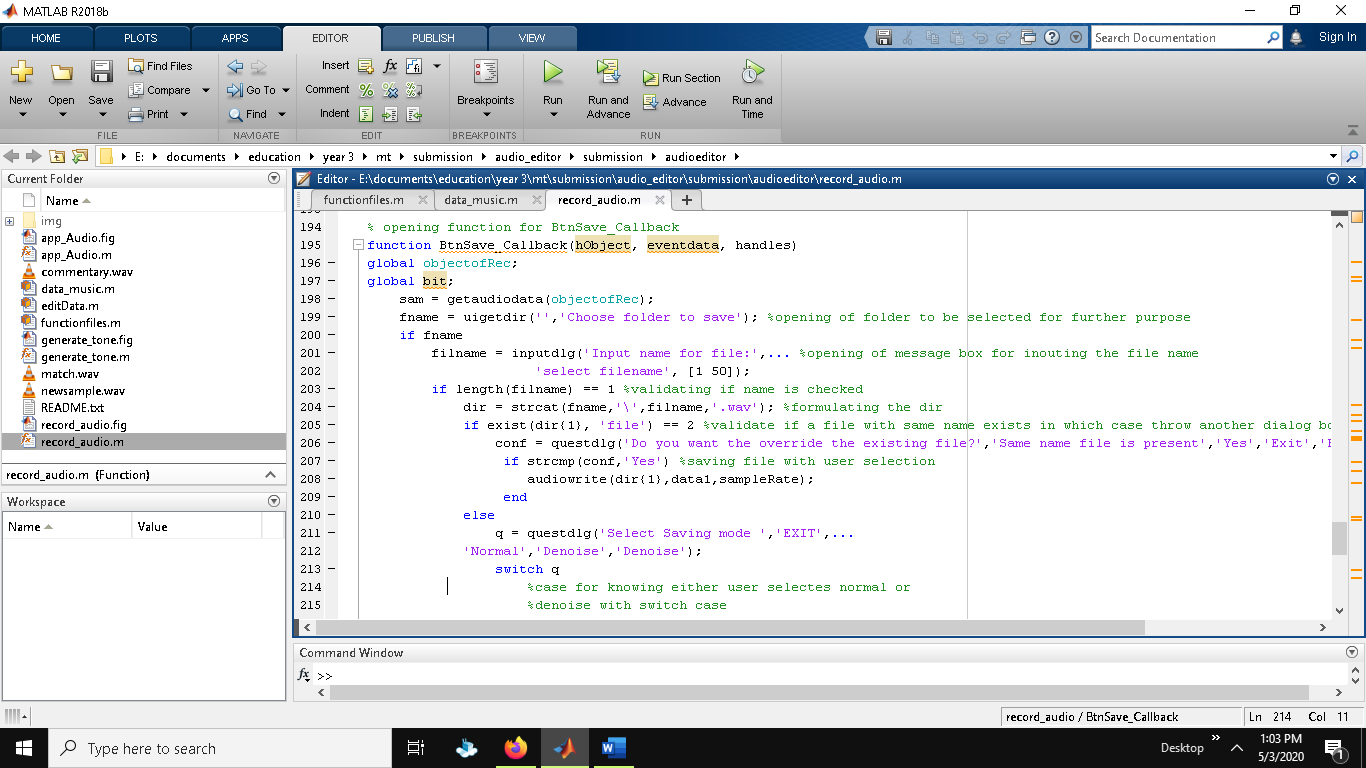


**Joining or Merging Audios**

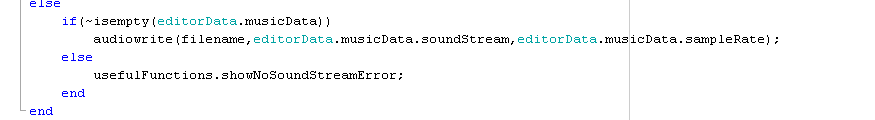
The process for merging sound is also same till the calculation of new sound samples according to given input duration which is then for updating the new sound piece1 and 2 are merged with the help of vertcat() function and then all the objects in GUI is updated as like in above processes. Vertcat concatenates piece 1 vertically to the end of piece 2 when both have compatible sizes which is done with the help of similar code used for mixing audio before adding.



**Saving audio file**

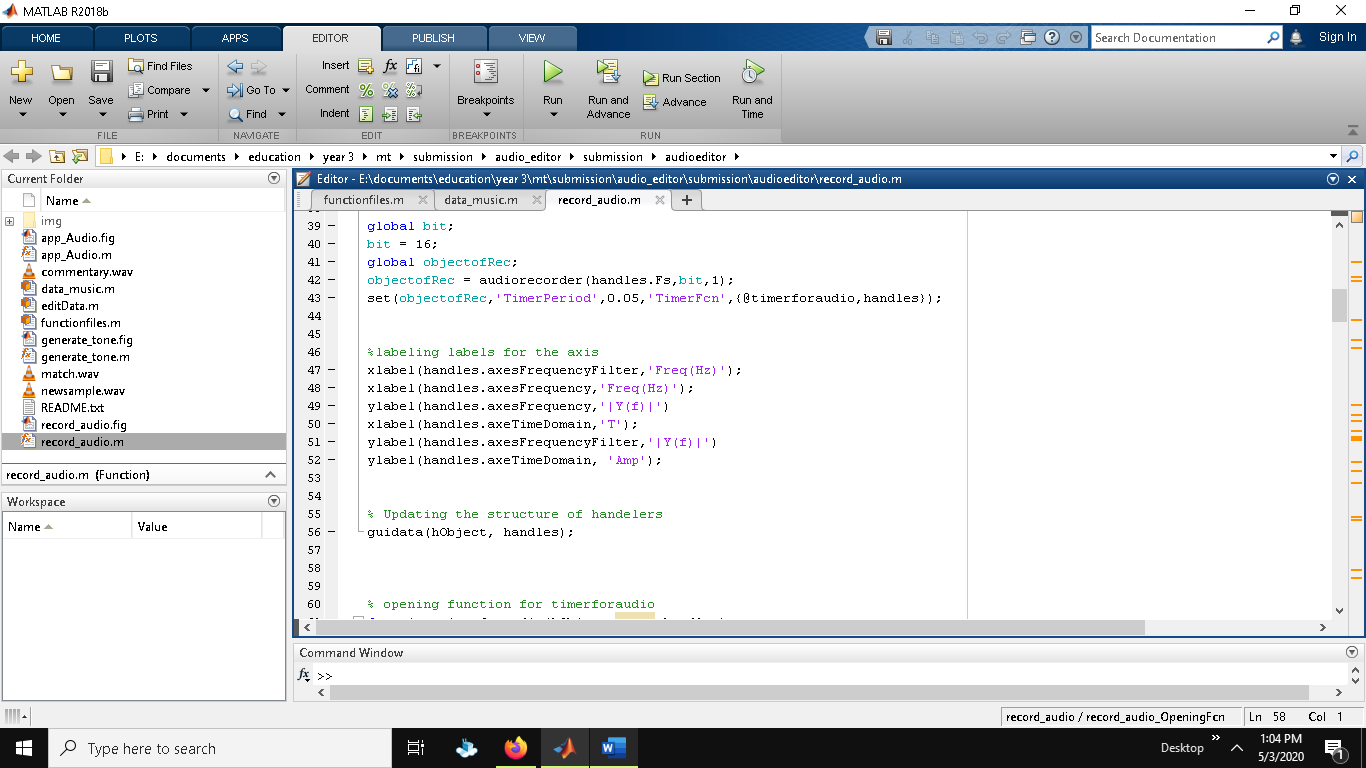
The function for saving audio file is not more complex it is just to get the object of sound file which writes with the help of audiowrite function providing the detail of audio like sample, frequency and path if required. For this system only for audio recording path is selected and then file name is given by user in system GUI where for other GUI the save function is carried directly at once. Below is the saving function audio recording.

Below is the saving function code for except audio recording so here it is simply done with single step by defining destination path and filename where audiwrite function is provided with the data of filename, sound sample and frequency.

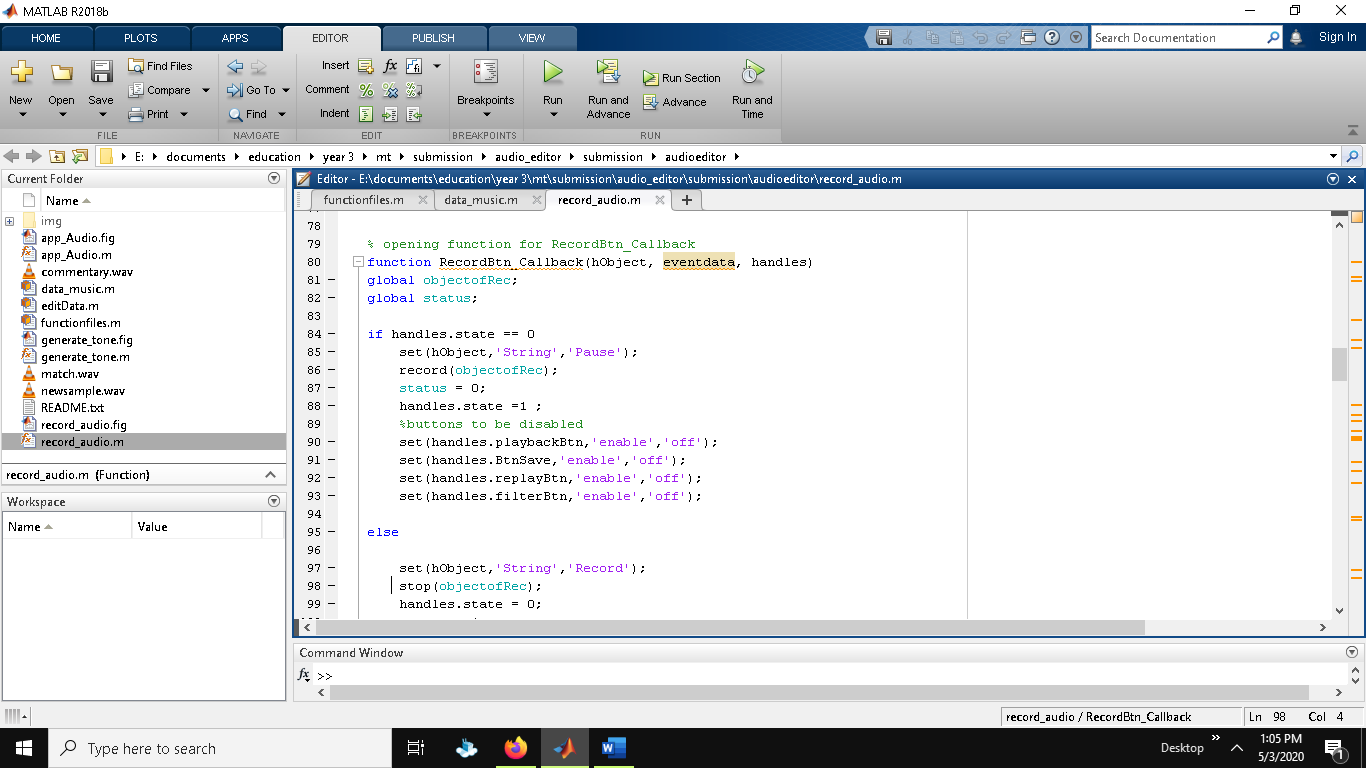


**Recording Audio**

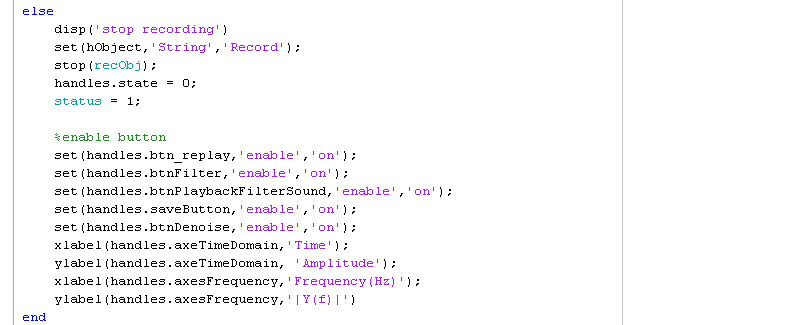
For recording audio, new GUI is used where in the opening function, recObj is created globally so that audiorecorder function can be applied with the input of given frequency and bits which is defined 8192 and 16, respectively. Beside that the x label and y labels are for the axis of recording, filtered sound and denoise sound.



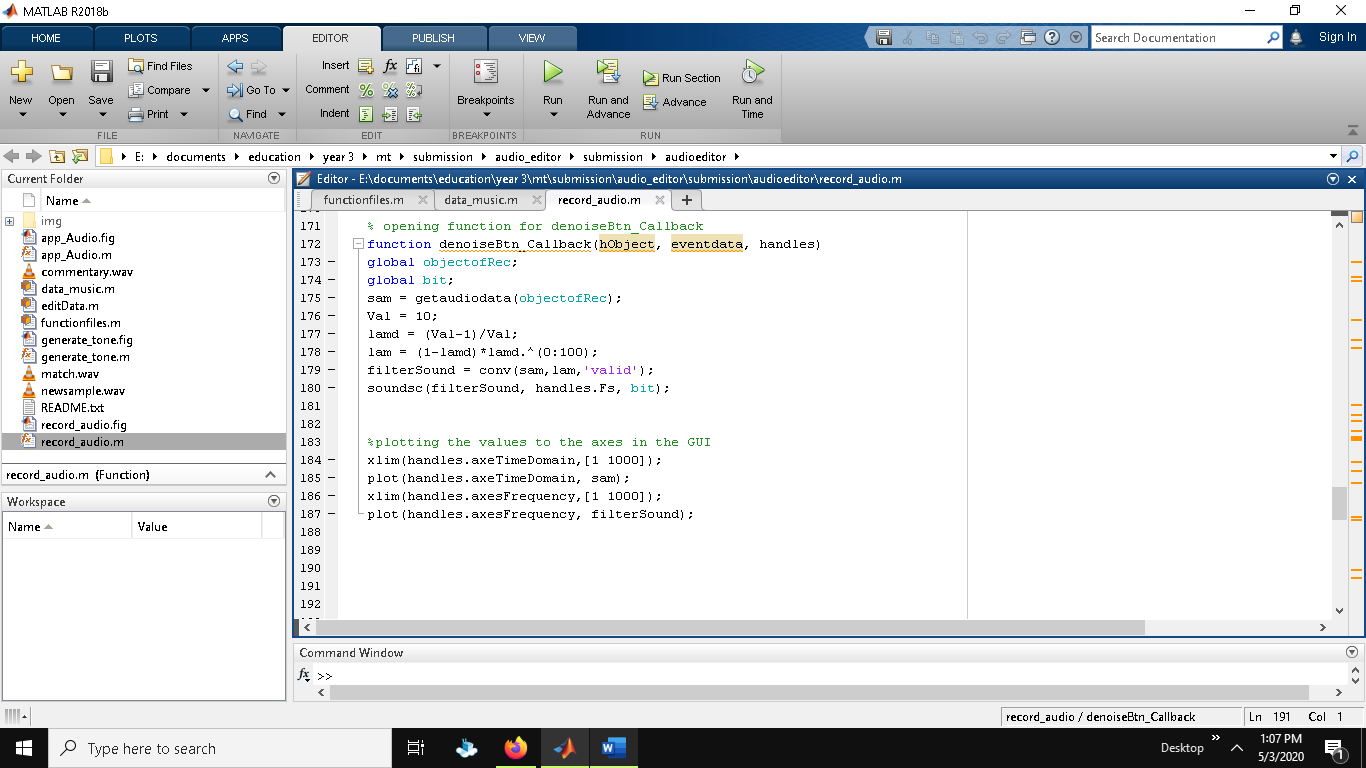
After defining the recording object and arises, now on click of record button it triggers to this function which checks either the state is 0 or not and if 0 then it sets the string to pause and then record the object which states the handle to 1 and until the recording all other buttons are set disabled.



When state is 1 then recording is stopped with the help of stop function to recording object and then handle is set to 0 after which all the buttons are set enabled.

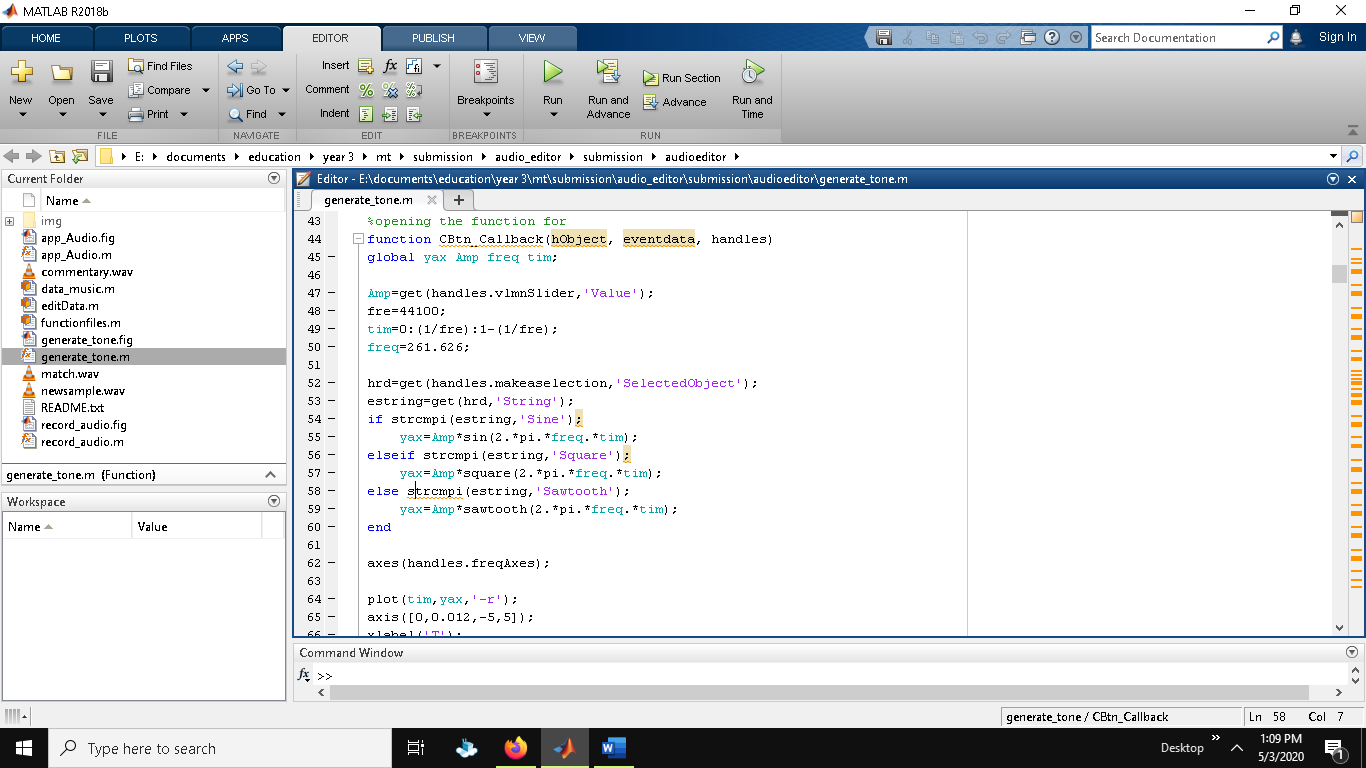


**Denoise audio**

To denoise audio first recording object is called and read with getaudiodata which is then filtered with conv function passing audio data and the lambda value and then with the help of soundsc function it takes sound sample and frequency with bits which sends signal to vector y to the pc speaker where before played signal y is scaled resulting sample sound to be played as loud without clipping as possible.

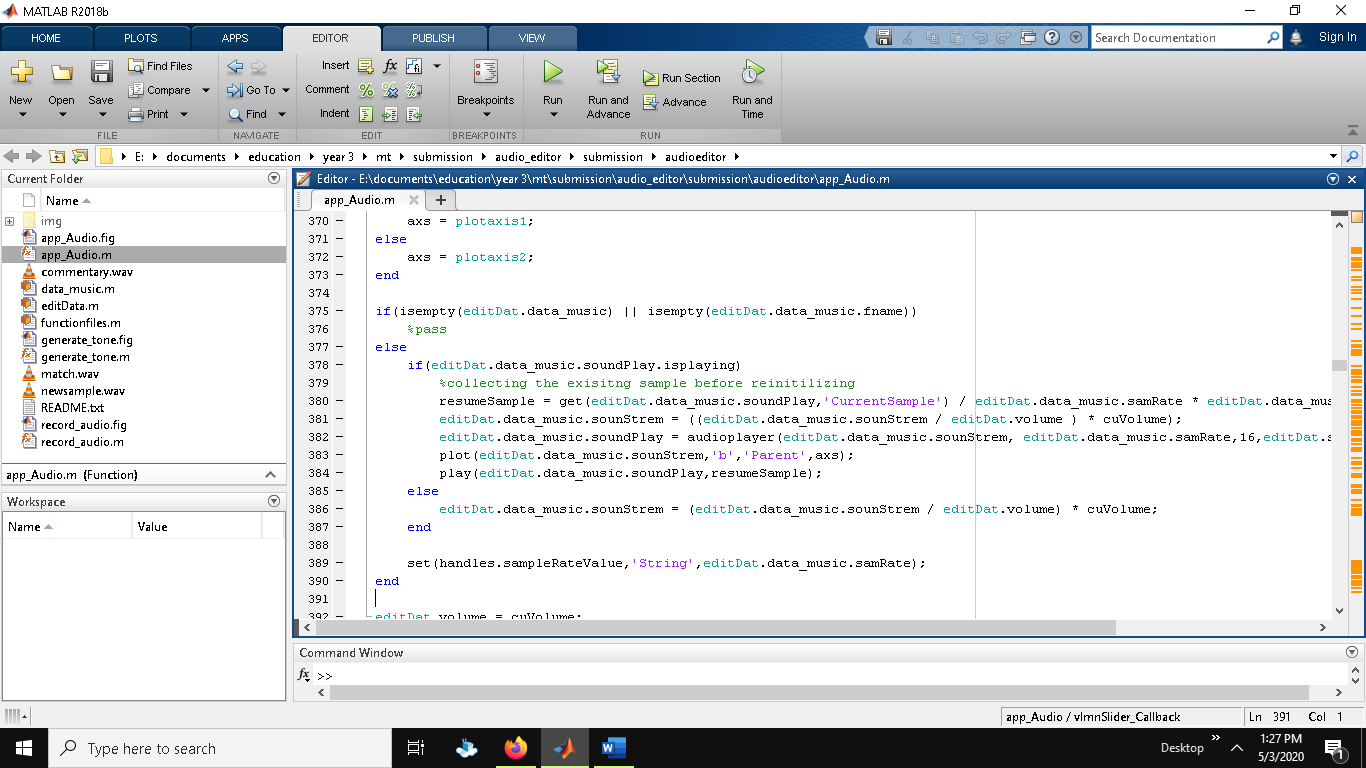
**Generate Tone**

For this section, the generation of new tone is explained where nearly every function has same creation and call back returns. In this class, the tone generation and recording of object with audio recorder is same for all the sharp or labels like a, b, g, f , a# etc. with their unique frequency to be played on pressed. The global variable included here are frequency, sample rate, channel , amplitude, and local frequency where each of these variable plays a role to update the data to produce sound at given frequencies. With selection of filter, the conditional statement starts where it decides to give which filter value to the sound sample. And then with the completion of sound sample at desired frequency, the axes are then plotted with new data which is then displayed during playing sound.



**Volume and playback sider**

The implementation for both cases are same as they act upon the slider where at first the system checks for axis which is justified with position of the button clicked in the track file section upon which the axis is adjusted or plotted later on. The code passes through nested conditional statement where at first it validated either it is empty or not and pass if it is validated and then with the help of get method the new resume sample is collected through which the new volume is adjusted according to the slider values and the sample rate of audio is decided with the data\_music class which contains information about the music that are used through editDat class and then with the success of getting volume it is read with audioplayer and then plotted with new data. This case is same for playback slider to which changes the frequency to play.



## 7.2 System technical notes recap

The overall revision to this system implementation is formatted according to the code standard where all the referenced code are given proper citation and reference where many of the function have same properties with same body content beside some unique constraints used The creation for the 3 GUI are same but the main GUI handles the two other GUI by calling them and the saving option for 3 GUI is implemented in same order as well wit the help of audio write function where in main GUI the sound sample is read with audioplayer but in the two GUI the sound sample is read with audiorecorder as they are current recording object where first GUI consist of exiting audio sound sample. The closing function for each GUI is same which is referenced from one of the online exiting code for which the citation is mentioned on the starting of technical notes. In the main GUI the function for audio merge, reverse, trimming is implemented in same order beside the section for formation of new sample which were decided as request by the event behaviour. The mixing of two files is same for ether cases like when track 1 act like man track or track 2 act like main track so that there wont be any problem and user get flexibility , and for this case each the user requests then the system check for the channels f matched or not then act upon that either to duplicated the sample or not and add the required blank audio with the help of zero for CONCAT matrices accordingly with the calculation of desired section of sound duration which is adjusted with mixer slider and same case is followed for play back speed and volume change slider.

# 8 Test Strategy

Testing strategy is the important role in system development where it decides whether system build contains bugs or not. It helps to overcome future failures or system crashes that may happen due to some short comes. There are different approaches for system testing which are as below:

## 8.1 Unit Testing:

For the case of unit testing, all possible small units of system are tested individually with different test cases based on system implementation. Overall, unit testing covers the total coverage each unit independently.

## 8.2 Black Box Testing

To know either the system functionalities or workflow are handled correctly, black box testing is carried out. The basic concept of black box testing is to see the system through the behavior. With the positive results of expected result with actual result, the system is hence verified for handed over to client.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Scenario** | **Test Data** | **Expected Result** | **Actual Result** |
| Mixing audios | 1 Two valid files  2 One invalid file | 1. Mix audios 2. Error message | 1 Audio mixed  2 Error message “Please import valid file “ |
| Merge audios | 1 Two valid files and valid timeframe  2 One invalid file and valid timeframe  3 Valid files and invalid timeframe | 1. File merged 2. Error message 3. Error message | 1 Audio merged  2 Error message “Please import valid file”  3 Error message “Duration of audio didn’t match” |
| Reverse Audio | 1 Valid file  2 Invalid files | 1. File reversed 2. Error message | 1 Audio reversed  2 Error messages ”Please import valid file” |
| Trim Audio | 1 Valid file and valid duration  2 Invalid file and valid duration  3 Valid file and in valid duration | 1. File trimmed 2. Error message 3. Error message | 1 Audio trimmed  2 Error message “Please import valid file”  3 Error message “Please input valid duration” |
| Save Audio | 1 Select file  2 Empty files | 1. File exported 2. Error | 1 Audio saved  2 Error message “ Please select file first” |
| Generate tone and save | 1 Keyboard played  2 Keyboard not played | 1. File saved 2. Error message | 1 Tone saved  2 Error message “Please generate tone with keyboard first” |

# 9 System Evaluation

## 9.1 Heuristic Evaluation

It is the evaluation method which determines either all the requirements were meet or not so that if system lacks performance or functionalities then the development should do to solve that issues. Topics that will determine the system usability are described below with the system evaluate result after completion.

**System status visibility**

All the visible solution features are usable, and the system design is designed in such a way that it never feels new if an individual have not used such system yet. The motive of making system friendly is successful.

**System with real world**

With the real time inspected research, system is made very realist where features which should be given more priority belonging to real world are given more priority and completed as well. With the respect of creativity and innovation, all are implemented beside the bad practices.

**Users freedom and control**

There is no restriction provided to users as it is freely access to every user and all the features can be used by users so that users can give their best response and system development will be succeeded. Hence users are given full freedom and they can feel comfortable with the recognition of error if occurs.

**Consistency**

Since system is fully depended in single server where it runs so there is not any problem regarding speed and data storage with the problems like errors causing data, so backup is not any more problem.

**Error prevention**

For every possible error that may cause due to invalid data inputs, recognizing error messages are given which can suggest users and no more error will occurs beside that as testing is already done and tested positive.

**Recognition and recall**

Since system UI is designed so simple that any use can get all the meaning of buttons or navigations used as well as users do not have to pressure themselves to recall all the functionalities or event triggers.

**Flexibility and efficiency**

Since the system is single server based so there will not be any problem regarding multi-tasking and users can use as wisely as they can with the same speed depending upon the server which will be their pc .

Since, all the above mention requirement for system development and performance are satisfied after the completion of system development it makes sure that system will be very useful to users and all the designs and goals that are limited to system which were found during problem domain are justified.

## 9.2 System Pilot Trials

For evaluating the system performance, survey have been done with the engagement of 2 users who are new for this kind of system. Beside this survey developer have uploaded video which link is available at the report heading about this system in full detail. Below is the response given by the users about the performances and system functionalities.

|  |  |  |
| --- | --- | --- |
| **Users** | **Events or Activity** | **Response** |
| User 1 | 1 System design  2 Mixing audios  3 Merging audios  4 Recording audios  5 Trimming audios | 1 It is simple and attractive  2 To mix two audios it is very easy and very fast  3 Very fast and easy and I can do multitasking  4 Recording audios is very attractive and removing noise from background is interesting  5 It is very helpful to trim audios so that we can use it according to our need |
| User 2 | 1 System design  2 Mixing audios  3 Merging audios  4 Recording audios  5 Trimming audios | 1 Easy to navigate and looks good  2 It is very helpful to mix audios at required section of time  3 It is interesting that we can trim and merge audios at same time  4 All the time we may not have the commentary, but recording is very helpful for that situation  5 I already mentioned it is very helpful for users like us to trim audios we need |

# 10 Project Conclusions

Overall, the project went smooth with the completion of requirement given by client as well as extra functionalities related to same topic, although the system development with MATLAB was new to use but with the knowledge and idea of using octave it gave us idea to control MATLAB. Few of the open source code were referenced in order to complete the development by knowing how the system runs because of which I can carry out functionalities and code according to need by myself for implementation .

The system started with the initial and main phase of system development which is background research and literature review. After gathering information about the development of sound annotator and its condition now to connect with the real world, I was able to evaluate the problem domain upon which solution was carried out and designed with the motive of making UI simple accordingly with the wise use of requirement engineering which decides either the system meets the requirements and can survive or not. The system is evaluated after the completion of development with the justification of problem domain encountered during background research so real time testing and several test cases which are carried out justifies that the system is ready to go and be very useful to users with the simplex UI design which is discussed at system pilot trials and system video demo.

The main problems which were not letting to implement system during development process was due to un combinational dimensional array for the case of mixing and merging audio sounds which was successfully developed with few researches on the topic for Concatenating matrices in MATLAB and proper citation with references are provided as well in the report. In time was given more then the system can be implemented for mixing audio files with video files which is not far away with the system development.

# 11 References

1. effects, c., 2020. *US8331585B2 - Audio Mixing - Google Patents*. [online] Patents.google.com. Available at: <https://patents.google.com/patent/US8331585B2/en> [Accessed 20 April 2020].
2. Ieeexplore.ieee.org. 2020. *Semantic Annotation And Retrieval Of Music And Sound Effects - IEEE Journals & Magazine*. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/4432652> [Accessed 20 April 2020].
3. Mathworks.com. 2020. *Mixing .Wav Sound Files*. [online] Available at: <https://www.mathworks.com/matlabcentral/fileexchange/72414-mixing-wav-sound-files> [Accessed 20 April 2020].
4. (How can i mix two .wav files ..iam using this code as show in figure but it gives an error? - MATLAB Answers - MATLAB Central, 2020)
5. Mathworks.com. 2020. *How Can I Mix Two .Wav Files ..Iam Using This Code As Show In Figure But It Gives An Error? - MATLAB Answers - MATLAB Central*. [online] Available at: <https://www.mathworks.com/matlabcentral/answers/214946-how-can-i-mix-two-wav-files-iam-using-this-code-as-show-in-figure-but-it-gives-an-error> [Accessed 20 April 2020].
6. Mathworks.com. 2020. *How To Position A GUI Centered On The Desktop - MATLAB Answers - MATLAB Central*. [online] Available at: <https://www.mathworks.com/matlabcentral/answers/236707-how-to-position-a-gui-centered-on-the-desktop> [Accessed 21 April 2020].
7. Mathworks.com. 2020. *Noise Reduction By Wiener Filter*. [online] Available at: <https://www.mathworks.com/matlabcentral/fileexchange/68468-noise-reduction-by-wiener-filter> [Accessed 21 April 2020].
8. GitHub. 2020. *Ittus/Matlab-Voice-Record*. [online] Available at: <https://github.com/ittus/Matlab-Voice-Record> [Accessed 21 April 2020].
9. GitHub. 2020. *Priichan/Audio-File-Editor*. [online] Available at: <https://github.com/priichan/Audio-File-Editor> [Accessed 21 April 2020].
10. Tutorialspoint.com. 2020. *MATLAB Tutorial - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/matlab/index.htm> [Accessed 20 April 2020].
11. Mathworks.com. 2020. *Create And Run A Simple App Using App Designer- MATLAB & Simulink*. [online] Available at: <https://www.mathworks.com/help/matlab/creating\_guis/create-a-simple-app-or-gui-using-app-designer.html> [Accessed 20 April 2020].
12. Mathworks.com. 2020. *Concatenate Variable Value And Variable Name Into New Variable Name. - MATLAB Answers - MATLAB Central*. [online] Available at: <https://www.mathworks.com/matlabcentral/answers/277275-concatenate-variable-value-and-variable-name-into-new-variable-name> [Accessed 20 April 2020].
13. Andor, R. (2016). *andorremus/audio\_editor*. [online] GitHub. Available at: https://github.com/andorremus/audio\_editor [Accessed 20 April 2020].
14. DIVA. (n.d.). *http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1334188*. [online] Available at: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1334188&dswid=9835 [Accessed 20 April 2020].
15. Kamen, E.W. and Heck, B.S. (2007). *Fundamentals of Signals and Systems Using the Web and MATLAB*. [online] *Google Books*. Pearson Prentice Hall. Available at: https://books.google.com.np/books?hl=en&lr=&id=RxMfAQAAIAAJ&oi=fnd&pg=PR9&dq=audio+app+with+matlab&ots=Alg-Z\_Usme&sig=9twIRF6r09e-K\_yzvrrGj58WJmU&redir\_esc=y#v=onepage&q&f=false [Accessed 20 April 2020].

# 12 Project Gantt Chart

In this section, the work done in order to complete this project is described with the given weeks where the cells inside cells represents the work load to completed as expected and the green colour shows the workload actually completed time for both system development and report writing.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Description Week** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| System introduction | |  | | --- | |  | |  |  |  |  |  |  |  |  |  |
| Project aims and objectives | |  | | --- | |  | |  |  |  |  |  |  |  |  |  |
| Problem domain |  | |  | | --- | |  | |  |  |  |  |  |  |  |  |
| Project Development Methodology |  | |  | | --- | |  | |  |  |  |  |  |  |  |  |
| Requirement Engineering |  |  | |  | | --- | |  | |  |  |  |  |  |  |  |
| Elicitation Activities |  |  | |  | | --- | |  | |  |  |  |  |  |  |  |
| Comparable system research |  |  | |  | | --- | |  | | |  | | --- | |  | |  |  |  |  |  |  |
| Development Relevant Legislation |  |  |  | |  | | --- | |  | |  |  |  |  |  |  |
| Academic Literature Review |  |  |  |  | |  | | --- | |  | |  |  |  |  |  |
| Any other relevant problem domain investigation data |  |  |  |  | |  | | --- | |  | |  |  |  |  |  |
| Requirement Specification |  |  |  |  | |  | | --- | |  | |  |  |  |  |  |
| System development |  |  |  |  | |  | | --- | |  | | |  | | --- | |  | | |  | | --- | |  | | |  | | --- | |  | | |  | | --- | |  | |  |
| System testing |  |  |  |  |  |  |  |  |  | |  | | --- | |  | |
| System evaluation |  |  |  |  |  |  |  |  |  | |  | | --- | |  | |
| Conclusion |  |  |  |  |  |  |  |  |  | |  | | --- | |  | |