**Demo video link:**

[**https://northampton.mediaspace.kaltura.com/media/17425087/1\_cmwbiu8k**](https://northampton.mediaspace.kaltura.com/media/17425087/1_cmwbiu8k)

Table of Contents

[Introduction 3](#_Toc535154400)

[Analysis and Implementation of Project Requirements 4](#_Toc535154401)

[Minimum Specification 4](#_Toc535154402)

[Basic Design Requirements 4](#_Toc535154403)

[Implementation of Basic Design Requirements 4](#_Toc535154404)

[Basic Functional Requirements 5](#_Toc535154405)

[Implementation of Basic Functional Requirements 5](#_Toc535154406)

[Advance Specification 10](#_Toc535154407)

[Forwarding and Backwarding features 10](#_Toc535154408)

[Volume Slider 12](#_Toc535154409)

[Music Slider 15](#_Toc535154410)

[Plot scanner 16](#_Toc535154411)

[Reverb 17](#_Toc535154412)

[Final Design of User Interface 19](#_Toc535154413)

[Conclusion 20](#_Toc535154414)

[References 21](#_Toc535154415)

# Introduction

MATLAB merges a desktop environment in order to built a interative design and analysis processes using programming language which directly expresses array and matrix mathematics. This technical report includes the explanation of the assigned project of designing and implementation of a music player created using MATLAB which covers the minimum specification of importing more than one audio file (.wav) and displaying them in a playlist. The playlist can be played, paused and resumed.

The asssignment also expects the programmer to add high quality useful additional features which could improve the usability of the player. The rationale principles behind those added features has been explained in analysis and design section of the report in details. User experience and UI designs for functionality has also been included.

The emphasis of the key elements for the implementation of putting the low level functions together nad technical guide of the working of the created software has been reflected in this report. The following screenshot displays the final design of the music player where how the functions and designs are implemented into the player in order to get the last following result is explained in the report.



# Analysis and Implementation of Project Requirements

## Minimum Specification

### Basic Design Requirements

First of all, the design of the project must focus in simplicity in order to improve the user experience, the user should be able to interact with the player easily understanding the working of it. If the design is easy to understand than the programmer can easily find the bug, hindering its proper working. The design should include GUI for user interaction which must be built using Matlab GUI toolkit, “GUIDE”.

For the design of the minimum specification of the music player, there require a sound files adding button which when clicked should search for the available wav sound files and add one or multiple of them in a listbox to be displayed as a playlist. ‘Play’, ‘Pause’ and ‘Stop’ buttons must be added in order to handle the files accordingly.

### Implementation of Basic Design Requirements

In order to meet the basic design requirements of the project, the following design has been done:

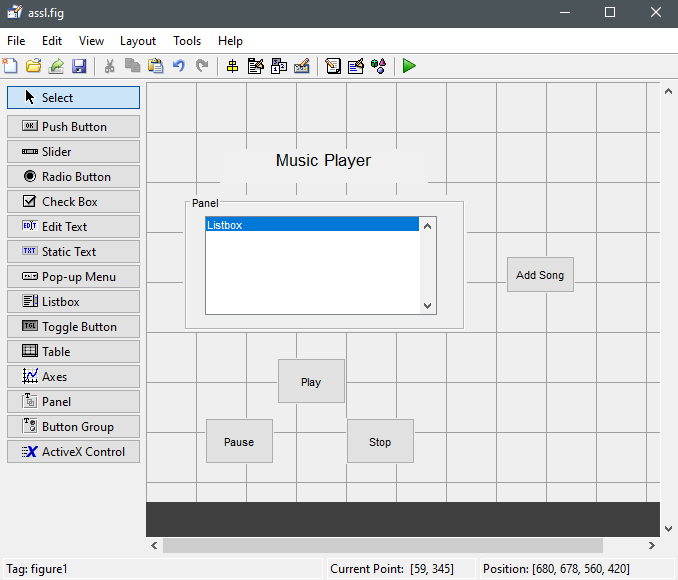
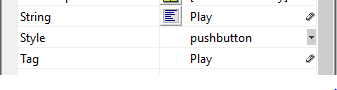


Fig I

A new guide was created by typing “guide” in the command window which allowed the creation of a new blank GUI. In order to add the required components to the blank GUI, the components are allowed to be selected from the component palette which is located at the left side. The desired panel or button groups could be cursored upon and dropped into the layout area .

For the basic design of the project, I dragged and dropped the panel upon which I dropped a ‘Listbox’ and a Push Button to add the song files into it .The three Push buttons to play,pause and stop sound files were then added. All of the components added were given a meaningful and unique identifier. In order to do so, the components were doubled clicked and after which the property inspector which is matlab UI control appeared on the screen. The String and the Tag of the components were selected and replaced with the necessary identifiers as shown above in fig I for the buttons. For the listbox ‘address’ has been given as identifier. The component Static Text has been used to display the text “Music Player” at the top of the layout screen.



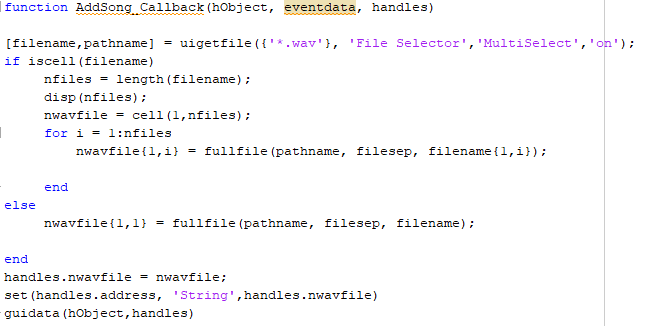
Then the GUI is saved and runned to check the final basic design.

### Basic Functional Requirements

In association to the design, the added components of the GUI must be fully functional. After clicking on the “AddSong’, push button, one or multiple sound files with .wav extention should be able to be searched and added to the playlist in the listbox. The selected song must then be played by clicking on the “Play” button. The song then should be paused and stopped while playing after clicking on “Pause” and “Stop” buttons repectively.

### Implementation of Basic Functional Requirements

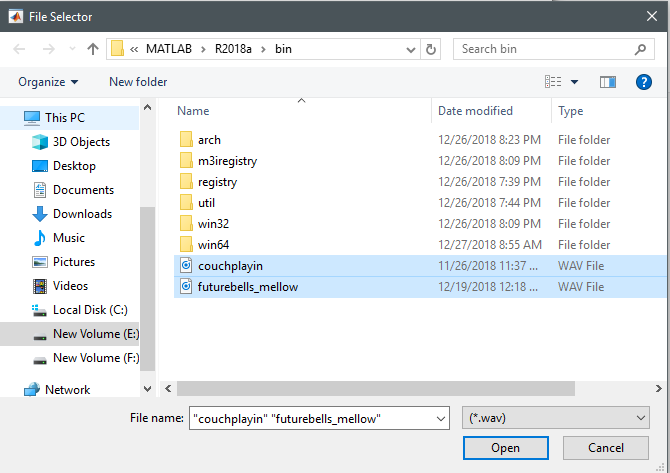
After saving the GUIDE, it added an empty callback function definition to the .m extention function file “ass1.m” with the control’s Callback property to each functions referencing to the created UI tag name of each component. The functions created could accept the inputs and return outputs as required.

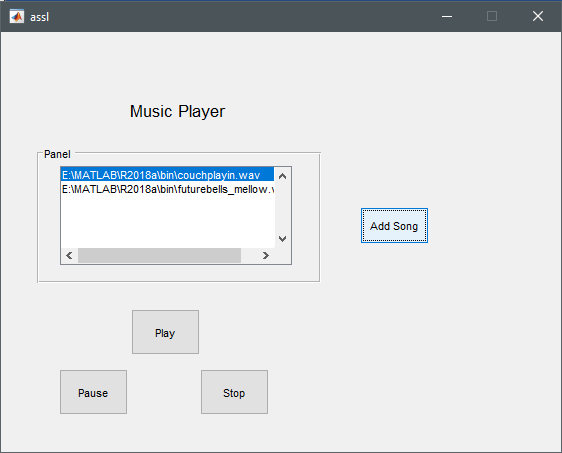


The function Add\_Callback has been created with the parameters hObject, eventdata,handles where hObject triggers the callback of the UI component named “AddSong” whereas eventdata is a variable containing detailed information of the specific action(in the case of AddSong pushbutton, when clicked should allow the song files to be added to the playlist) and handles is a struct containing all the objects in GUIDE. GUIdata function is used by the GUIDE to maintain and store structure handles.

Here, uigetfile returns filename and pathname to the files having .wav extension when the user clicked on “Open” else the uigerfile returns 0 as the output argument. ‘File Selector ’ was used ofcourse for the selection of the available files and ‘Multiselect ’ specifed the users to get the selection of multiple files. The mode of which is set ‘on’ to enable the multi selection. If condition is used where ‘iscell’ is used to determine whether the input is cell array checking whether a single file or multiple files had been selected. The selected file or files were added to handles and set to to the ‘address’ which is the listbox for the playlist.

Guidata(hObject,handles) stores ‘handles’ with the object specified by hObject. After the coding, a single or multiples files can be added to the playlist as required in the project.





Now, in order to play the selected sound file, the player must first of all be able to read the selected sound file. For which the listbox holding the sound files must have a callback function. Thus, function “address\_Callback” was created where coding has been done for users to select audio files.



The selected sound index was added to the handles structure and assiged it as a value of input argument hObject. Similarly addresses was added as a ‘string’ and returned as array. After which file selected\_items held the select\_index i.e one sound file from the addresses i.e. list of sound files.



The above code then read data from the file selected\_items and sampled data “y” and sampled rate for data “ f ” and made instance of the audioplayer with the above y and f.



Where pl was then added to the handles.

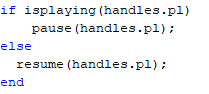


Finally, the handles structure got updated. After all these processes, the selected sound file was ready to get played.

Play button need to respond to the user’s click. The callback function should be made responsive to which function play\_Callback has been created which used the play method to play the selected sound file.



Similarly, the Pause button also had a callback function “pause\_Callback” which checked whether the sound file was playing or not using ‘isplaying’ method and if was playing, pause method was used to pause the sound file whereas the sound file could resume the sound file using resume method.



For the last basic specification of letting the user to stop the playing song, callback function stop\_Callback was created which simple used the stop method to stop the playing sound file.



With the implementation of all the above basic functional requirements, the minimum specification for the project is completed.

|  |  |
| --- | --- |
| Requirements | Remarks |
| Importing more than one audio files in ‘wav’ format and displaying them within the created playlist. | Done |
| Playing any file selected from the playlist | Done |
| Controlling the playback like pause and resume. | Done |

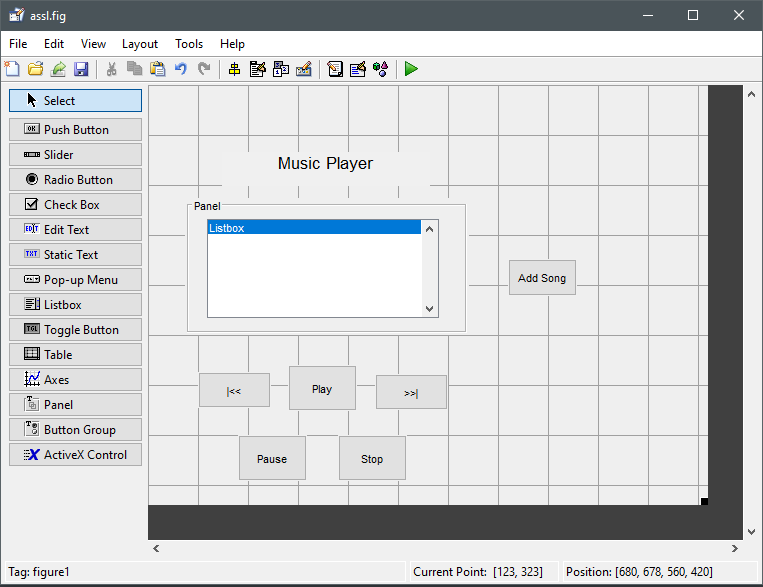
## Advance Specification

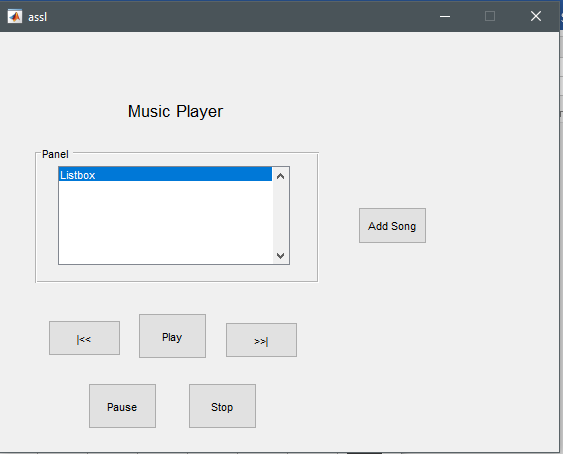
Since, the programmer is expected to add high quality additional useful features for improving the usability of the created music player, the following features have been added in the audio player.

### Forwarding and Backwarding features

The rational principle behind adding forwarding and backwarding features to the audio player is that the users do not have to search for the playing audio in the playlist in order to play the next or the previous audio from the list. Rather they can play the previous or next song in just one click.

The backward button with a backward symbol ( |<< ) and forward button with a forward symbol (>>|) have been placed to the left and right of the play button respectively as a traditional music player would have. For the functional requirement of these, the created buttons should be able to read the playing audio file so that the previous or the next file to the playing audio could be played, paused, resumed and stopped.

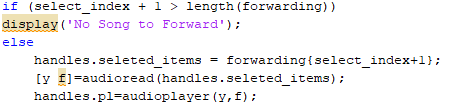




The forward button and backward button have the callback functions, ‘forward\_Callback’ and ‘backward\_Callback’ respectively where the necessary coding have been done to implement the functional requirements. For forwarding, the reading of the playing sound file has been done by taking variable ‘select\_index’ from the handles of the function ‘address\_Callback’ where adding of 1 to this variable automatically selects next audio in the playlist.

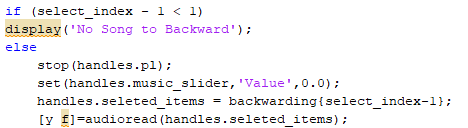


The variable ‘forwarding’ takes the list of the audio files from ‘handles.addresses’ which consists of the list of all sound files of the playlist. The if condition has been applied to check if there is a sound file after the playing sound file. If yes, after clicking the forward button by the user, the next sound would not be played rather ‘No song to forward’ message appears in the command window. Only if the condition is no, the next song in the palylist gets played.



The above sniffet shows the coding for forwarding the audio in the playlist after clicking the forward button.

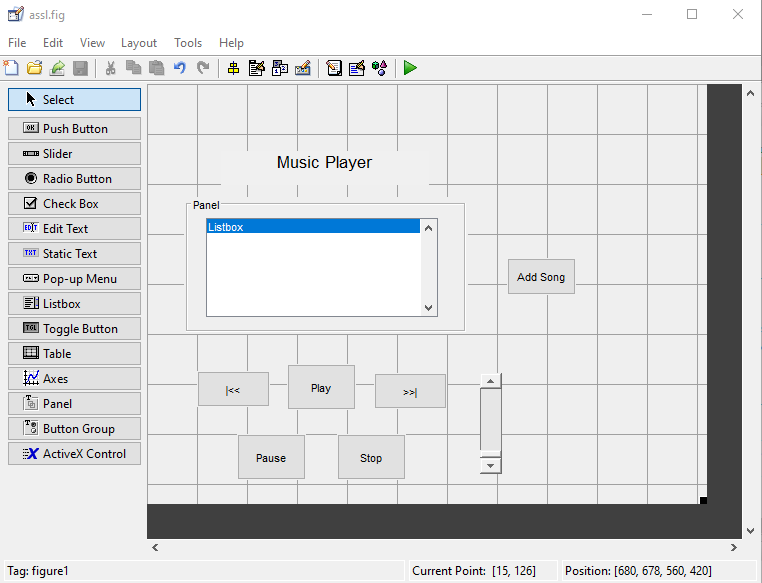
For backwarding, the reading of the currently playing sound file is done similarly as done while forwarding the sound file. The only difference is in the implementation of if condition where this time, it checks if there is any sound file before the playing file. If yes, the previous audio would not be played displaying ‘No song to Backward’ in the command window as with the forward button. If no, the previous song in the playlist gets played.



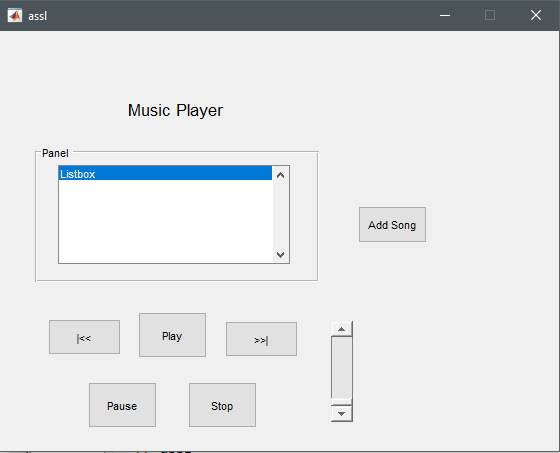
The above sniffet shows the coding for the working of the backwarding of the audio in the playlist.

### Volume Slider

The rational principle behind adding the volume slider feature to the audio player is that the users can be able to adjust the volume of the audio being played as their will. They do not have to go to the system’s volume adjuster/speaker just to adjust the volume of the audio which eventually increases the user usability of the created audio player.



The volume slider is kept at the right next to the buttons in the vertical position as a traditional music player would have. So that, the user can be able to slide the volume up or slide the volume down as their will.

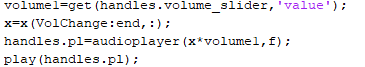


The function of the volume slider is that the user should be able to increase or decrease the volume where the audio player should play with the increased or decreased volume without restarting the sample play. For which the sample playing must be read by the player.



Here, get is the query property values for the object audioplayer. The ‘VolChange’ value is returning the ‘CurrentSample’ being played for object ‘handles.pl’.

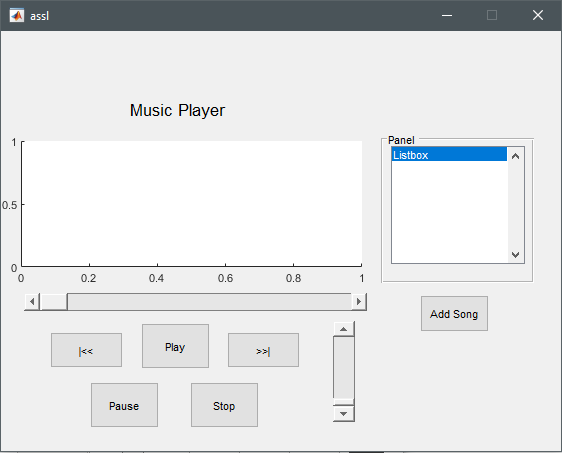
After returning the ‘CurrentSample’ being played, the player must read, add the new value to the handles and finally play the changed volume value of the sample so that the user can be able to listen to the audio with the changed volume.



The above snippet displays how the ‘volume1’ is returning the value for volume slider . we can see that ‘x’ is extracting the returned ‘VolChange’ through the last element. The audioplayer then plays the multiplied value of ‘x’ with the playing sample changing the volume according to the slider.

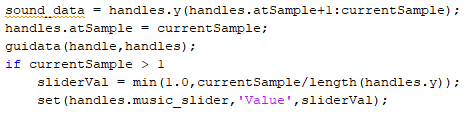
### Music Slider

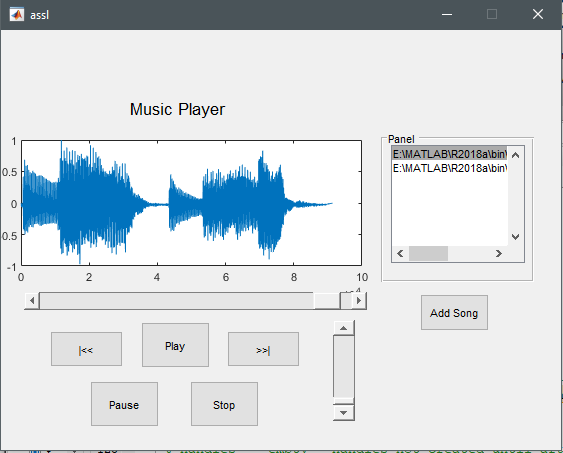
The rational principle behind adding the feature music slider to the audio player is that users can be able to view the playing of the audio as it proceeds along with listening to it.



As we can see, the music slider is placed at the top buttons and volume slider. The axes is also placed at the top of the music slider so that the plotting of the audio could be done along with the moving of the slider from left to right. The playlist listbox is shifted to the top right of the player.

In order to update the slider position in cordination to the playing audio, the programmer assigned a periodic timer to the audio player as in every one tenth of the second, the position of the slider gets updated. Firstly, the player gets the structure of the handles and then ‘currentSample ’ is used to determine what the current value is. After getting all the data of the sound by ‘sound\_data’ . the object handles is updated by the currentSample and than the slider gets updated according to the playing audio.

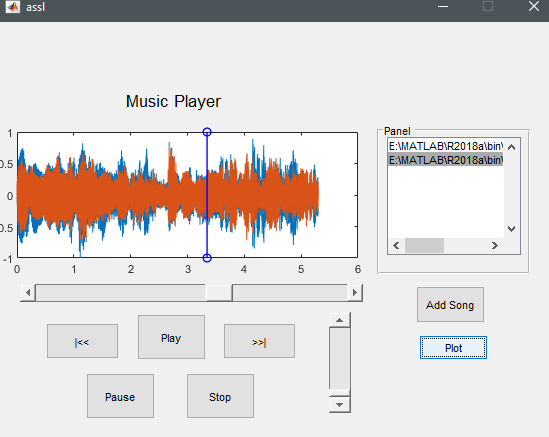




The plotting of the audio has been done in the ‘axes1’ of the player.

### Plot scanner

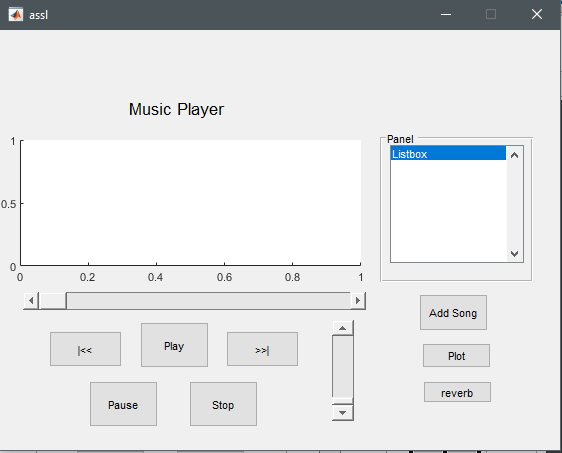
Since simply plotting of the audio data has been done, the user cannot be able to know where exactly the audio has reached during the proceeding of the audio like with the music slider. Thus, a different pushbutton ‘Plot’ was created which can show exactly where the audio has reached.



Thus, by selecting a ausio and pressing plot button, the selected audio plays where that music slider moves in coordination with the blue line in the axes.

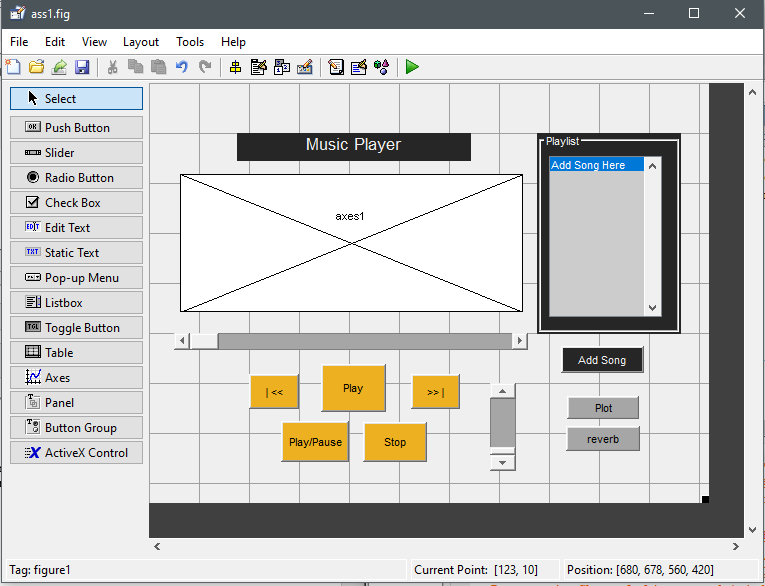
## Reverb

Lastly, since the programmer is familiar with the manipulation of the audio, a ‘reverb’ button which an effect in the sound which is produced by the amplifier making a reverable/echo sound was added to the player.



# Final Design of User Interface

Since, the design of the user interface must be simple and attractive. The design of the player is retouched were the background picture can been given to the player. The color and font size of the text along with the sizes of the button was modified.



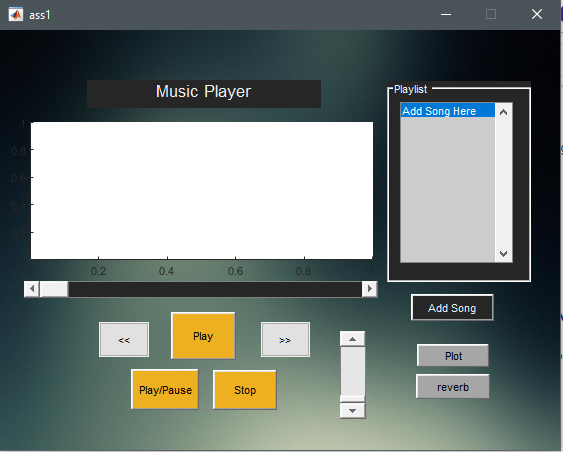


Fig: Final design

# Conclusion

In conclusion all the basic design and functional requirements have been implemented in the developed music player along with the mentioned additional features which has increased the user usability. The additional UI designs has been included for functionality. However, the developer wished to plot the moving axes functionality to the player but only the scanning of the plot which playing the audio as a additional features could be implemented. Overall, all the functions works properly so the developer is happy with the outcome of the project.

The project helped the programmer to understand different interactive MATLAB toolboxes with which the development of the music player had to be done. Moreover, using the MATLAB GUI and its controls helped with the practicing of the designing skill of the developer. This porject helped the developer to get familiar with the programming scripts and functions used in the MATLAB using GUIDE.

# References

Mathworks.com. (1994-2019). Read audio file - MATLAB audioread. [online] Available at: <https://www.mathworks.com/help/matlab/ref/audioread.html>.

Mathworks.com. (1994-2019). *Query property values for audioplayer object - MATLAB*. [online] Available at: <https://www.mathworks.com/help/matlab/ref/audioplayer.get.html>

GUI, V. (2015). *Volume Slider in Matlab GUI*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/32982549/volume-slider-in-matlab-gui>

Mathworks.com. (2015). *Make the Slider Move as the Music is Playing - MATLAB Answers - MATLAB Central*. [online] Available at: <https://www.mathworks.com/matlabcentral/answers/222105-make-the-slider-move-as-the-music-is-playing>.

Mathworks.com. (2016). *Reading audio file using GUI pushbutton and saving them to a folder - MATLAB Answers - MATLAB Central*. [online] Available at: <https://www.mathworks.com/matlabcentral/answers/281895-reading-audio-file-using-gui-pushbutton-and-saving-them-to-a-folder>.

Mathworks.com. (2011). *audioplayer - MATLAB Answers - MATLAB Central*. [online] Available at: <https://www.mathworks.com/matlabcentral/answers/18530-audioplayer>