PROJECT REPORT On

**“Music Player”**

***Submitted By***

**Aman Kundal**

## (161)

***Guided By:-***

Mr. Ratnesh K. Choudhary



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## S. B. JAIN INSTITUTE OF TECHNOLOGY

**MANAGEMENT AND RESEARCH, NAGPUR.**

(An Autonomous Institute, Affiliated to RTMNU, Nagpur)

### 2021-2022

#### © S.B.J.I.T.M.R Nagpur 2022

i

**S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT AND RESEARCH, NAGPUR**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

SESSION 2021-2022

# CERTIFICATE

This is to certify that the Project titled **“Music Player”** is a bonafide work of **Aman Kundal** carried out for the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering in **Computer Science & Engineering.**

|  |  |  |
| --- | --- | --- |
| **Mr. Ratnesh K. Choudhary**  Assistant Professor |  | **Mr. Animesh Tayal**  Head of Department |

# INDEX

[CERTIFICATE i](#_Toc21790)

[INDEX ii](#_Toc21791)

LIST OF FIGURES iii

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 METHODOLOGY 2

CHAPTER 3 TOOLS/PLATFORMS 3-4

CHAPTER 4 DESIGN & IMPLEMENTATION 5-11

4.1 ALGORITHM

4.2 FLOWCHART

4.3 SOURCE CODE

CHAPTER 5 RESULT & DISCUSSION 12-

14

5.1 OUTPUT

5.2 DISCUSSION

5.3 APPLICATION

CHAPTER 6 CONCLUSION 15

REFERENCES 16

**LIST OF FIGURE**

|  |  |  |
| --- | --- | --- |
| **FIG. NO.** | **TITLE OF FIGURE** | **PAGE NO.** |
| 4.2 | Flowchart for music player | 6 |
| 5.1.1 | Initial player interface | 11 |
| 5.1.2 | Importing interface | 12 |
| 5.1.3 | After selecting song | 13 |
| 5.1.4 | After Deleting the song | 13 |

### CHAPTER 1

### INTRODUCTION

Audio is an important source of communication and is as important as text in today’s time. We know that the audio files are digital files. Therefore, there is a need of a tool to run the digital files or in other words, play the files. Without this tool or player, we’ll never be able to listen to music, movies or the contents of any audio file.

Thus, we need MP3 players. It is a device using to play MP3s and other digital audio files. We can build this by ourselves without have to download and install premium music players. The Mp3 player GUI project idea attempts to emulate the physical MP3 Player.

This program will allow you to play songs, music, and all MP3 files on your desktop or laptops. MP3 player using Python is a basic programming application built using the programming language

Python. It is a GUI program built by the means of Python libraries Tkinter, Pygame and Mutagen.

We need an application that will allow us to play or listen to digital audio files. MP3 player is the device to play MP3s and other digital audio files. The MP3 GUI program application attempts to emulate the physical MP3 Player. This program will allow you to play songs, music, and all MP3 files on your desktop or laptops.

The GUI aspect of the application is built using the Tkinter library of Python. The interactive part of the application that handles the MP3 files uses the Pygame and Mutagen libraries.

Python has libraries that can play audio files, such as Pygame, which allows you to work with multimedia files in few lines of code. Similar libraries are Pymedia. These libraries can handle a lot of digital audio files. They can handle other file types, not just the MP3 files. You can also

implement a feature that allows users to create a playlist.

**CHAPTER 2**

### METHODOLOGY

#### 1. REQUIREMENT ANALYSIS: -

At this stage we will review what users need basic features in media player like start, stop, forward, backward, mute etc.

#### 2. PLANNING: -

In the planning stage, we should first try to explore out the features that the media player can have. Next, we will eliminate the features that users feel no really useful or low cost-effective.

Finally, each feature’s is prioritized and assigned to an iteration.

#### 3. DESIGN: -

The design stage is prepared according to the requirements of users. Since there are many details and problems encountered during development to be considered for each feature. Therefore, we will discuss and formulate solutions and test strategies to verify the product at this stage.

#### 4. IMPLEMENTATION: -

During the development phase, we will iteratively implement each of the features listed during the planning phase. At this stage, there will be many setbacks and obstacle, we need to constantly overcome these obstacles. Moreover, we will prioritize the most important features and need to make intelligent trade-offs between the depth of completeness of a single feature and the breadth of implementation of multiple features.

#### 5. TESTING: -

In this stage, we will test the performance of each feature in order to check whether it meets the requirements of users. For example, we will test whether the application can be properly run, and check whether any errors occur in the running process and each feature is up to standard.

**CHAPTER 3**

## TOOLS/PLATFORMS

**3.1 SOFTWARE REQUIREMENT:**

1. **IDE / FRAMEWORK:** Pycharm
2. **LIBRARIES:**, pygame ,tkinter, ttkthemes , mutagen , time , Os
3. **OPERATING SYSTEM:** Windows 11
4. **Language:** *Python*, Version: - 3.10.4

**1. Tkinter:** Tkinter is the standard GUI library for Python. Python when combined with

Tkinterprovides a fast and easy way to create GUI applications. Tkinter provides a powerful objectoriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task.

All you need to do is perform the following steps –

* Import theTkintermodule.
* Create the GUI application main window

**2.Pygame:** Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.

**3.OS :** The OS module in python provides functions for interacting with the operating system. OS,

comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. The \*os\* and \*os.path\* modules include many functions to interact withthe file system.

**4.Threading :** In simple words, athreadis a sequence of such instructions within a program that can be executed independently of other code. For simplicity, you can assume that a thread is simply a subset of a process!A thread contains all this information in aThreadControlBlock(TCB):

ThreadIdentifier:Unique id (TID) is assigned to every new thread

Stackpointer:Points to thread’s stack in the process. Stack contains the local variables under thread’s scope.

Programcounter:a register which stores the address of the instruction currently being executed by thread.

Threadstate:can be running, ready, waiting, start or done.

Thread’sregisterset:registers assigned to thread for computations.

ParentprocessPointer:A pointer to the Process control block (PCB) of the process that the thread lives on.

**5.Time :** Time, which is measured by the system clock. System time represents a computer system's notion of the passing of time. One should remember that the system clock could be modified by the operating system, thus modifying the system time.

**6.Mutagen mp3 :** Mutagen is a Python module to handle audio metadata. It supports ASF,

FLAC, MP4, Monkey’s Audio, MP3, Musepack, Ogg Opus, Ogg FLAC, Ogg Speex, Ogg Theora, Ogg Vorbis, True Audio, WavPack, OptimFROG, and AIFF audio files. All versions of ID3v2 are supported, and all standard ID3v2.4 frames are parsed. It can read Xing headers to accurately calculate the bitrate and length of MP3s. ID3 and APEv2 tags can be edited regardless of audioformat. It can also manipulate Ogg streams on an individual packet/page level.

**7.Ttkthemes :** The "themed" aspect of the new Ttk widgets is one of the most powerful and exciting aspects of the new widget set. Yet because it does things quite differently from how Tk has traditionally worked, and because in trying to be flexible it does alotof things, it's certainly the most confusing for many people.

**3.2 HARDWARE REQUIREMENT :**

1. **PROCESSOR :** P IV or above.
2. **RAM :** 1 GB RAM.
3. **HARDISK :**120 GB

**CHAPTER 4**

### DESIGN & IMPLEMENTATION

#### 4.1 ALGORITHM

##### Step1. Start

**Step2.** Import the necessary module that are require for the music player e.g pygame ,tkinter, ttkthemes , mutagen , time , Os.

**Step 3.** Then Define the GUI .

**Step 4.** Define a function for Browsing a file .

**Step 5.** Define a function to add song playlist which want to Add .

**Step 6.** Define a function to Remove song playlist which want to Remove .

**Step** **7**. Define a function to show all the details about the Music player.

**Step** **8**. Define a function for count .

**Step** **9**. Define a function to play a song which want to play by user.

**Step** **10**. After that define for stop .

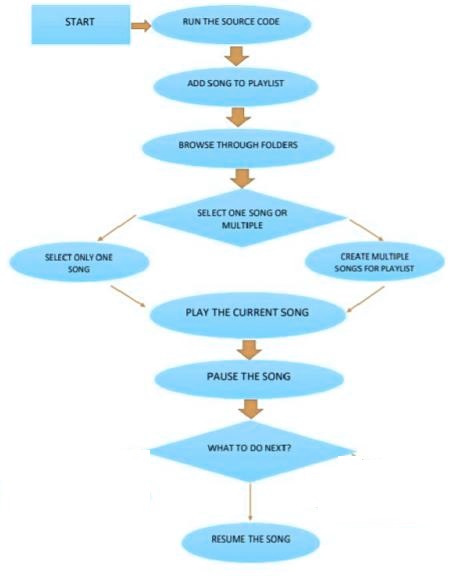
**Step** **11**. Then define for pause.

**Step** **12.** Define a function to rewind The music which want to rewind .

**Step** **13.** At last Define function to close .

**Step** **14**. STOP.

#### 4.2 FLOWCHART



**Fig 4.2.Flowchart for music player**

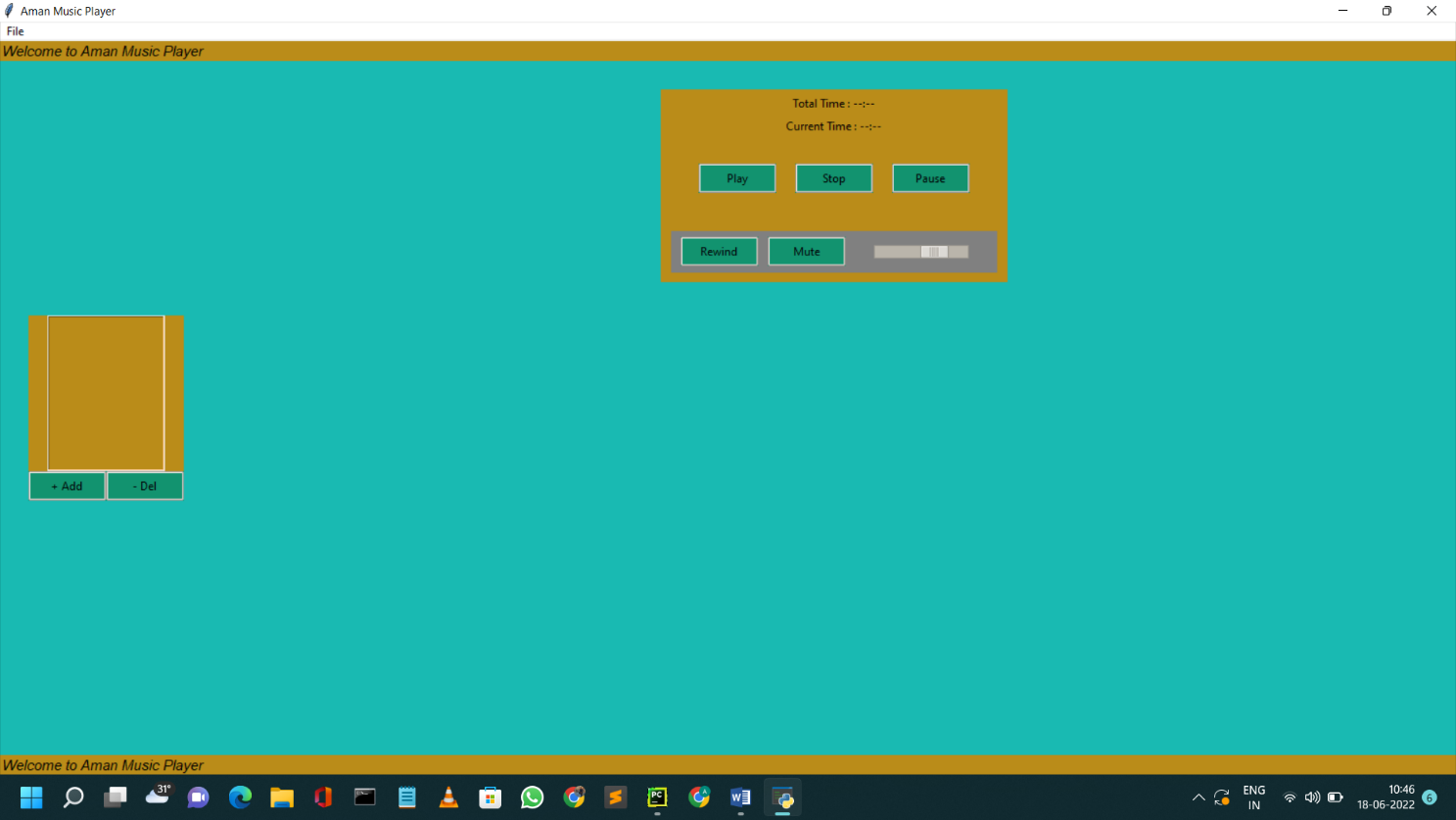
#### 4.3 SOURCE CODE

*"""  
Aman Presents Music Player  
"""*from tkinter import filedialog  
from tkinter import ttk  
import tkinter.messagebox  
from tkinter import \*  
import os  
import threading  
from ttkthemes import themed\_tk as tk  
from mutagen.mp3 import MP3  
import time  
from pygame import mixer  
  
root = tk.ThemedTk()  
root["bg"]="#19BAB2"  
root.set\_theme("elegance")  
statusbar = ttk.Label(root, text="Welcome to Aman Music Player", anchor=W, font='Arial 11 italic')  
statusbar.pack(side=BOTTOM, fill=X)  
statusbar1 = ttk.Label(root, text="Welcome to Aman Music Player", anchor=W, font='Arial 11 italic')  
statusbar1.pack(side=TOP, fill=X)  
menubar = Menu(root)  
root.config(menu=menubar)  
#// Create the submenu  
subMenu = Menu(menubar, tearoff=0)  
#list for storing play  
songplaylist = []  
def browse\_file():  
 global filename\_path  
 filename\_path = filedialog.askopenfilename()  
 add\_to\_songplaylist(filename\_path)  
 mixer.music.queue(filename\_path)  
def add\_to\_songplaylist(filename):  
 filename = os.path.basename(filename)  
 index = 0  
 songplaylistcontainer.insert(index, filename)  
 songplaylist.insert(index, filename\_path)  
 index += 1  
menubar.add\_cascade(label="File", menu=subMenu)  
subMenu.add\_command(label="Open", command=browse\_file)  
subMenu.add\_command(label="Exit", command=root.destroy)  
mixer.init() #// initializing the mixer  
root.title("Aman Music Player")  
ltframe = Frame(root,bg="#BA8C19")  
ltframe.pack(side=LEFT, padx=30, pady=30)  
songplaylistcontainer = Listbox(ltframe,fg='white')  
songplaylistcontainer["bg"]="#BA8C19"  
songplaylistcontainer.pack()  
addBtn = ttk.Button(ltframe, text="+ Add",command=browse\_file)  
addBtn.pack(side=LEFT)  
def remove\_song():  
 sel\_song = songplaylistcontainer.curselection()  
 sel\_song = int(sel\_song[0])  
 songplaylistcontainer.delete(sel\_song)  
 songplaylist.pop(sel\_song)  
root.style = ttk.Style()  
root.style.theme\_use("clam")  
remBtn = ttk.Button(ltframe, text="- Del", command=remove\_song)  
remBtn.pack(side=LEFT)  
root.style.configure('TButton', background='#12936F')  
rtframe = Frame(root,bg="#BA8C19")  
rtframe.pack(pady=30,padx=20)  
topframe = Frame(rtframe,bg="#BA8C19")  
topframe.pack()  
root.style = ttk.Style()  
#root.style.theme\_use("clam")  
root.style.configure('TLabel', background='#BA8C19')  
lengthlabel = ttk.Label(topframe,text='Total Time : --:--')  
lengthlabel.pack(pady=5)  
currenttimelabel = ttk.Label(topframe, text='Current Time : --:--')  
currenttimelabel.pack()  
def show\_details(play\_song):  
 file\_data = os.path.splitext(play\_song)  
 if file\_data[1] == '.mp3':  
 audio = MP3(play\_song)  
 total\_length = audio.info.length  
 else:  
 a = mixer.Sound(play\_song)  
 total\_length = a.get\_length()  
 mins, secs = divmod(total\_length, 60)  
 mins = round(mins)  
 secs = round(secs)  
 timeformat = '{:02d}:{:02d}'.format(mins, secs)  
 lengthlabel['text'] = "Total Time" + ' - ' + timeformat  
 t1 = threading.Thread(target=start\_count, args=(total\_length,))  
 t1.start()  
def start\_count(t):  
 global paused  
 current\_time = 0  
 while current\_time <= t and mixer.music.get\_busy():  
 if paused:  
 continue  
 else:  
 mins, secs = divmod(current\_time, 60)  
 mins = round(mins)  
 secs = round(secs)  
 timeformat = '{:02d}:{:02d}'.format(mins, secs)  
 currenttimelabel['text'] = "Current Time" + ' - ' + timeformat  
 time.sleep(1)  
 current\_time += 1  
def play\_music():  
 global paused  
 if paused:  
 mixer.music.unpause()  
 statusbar['text'] = "Music Resumed"  
 paused = FALSE  
 else:  
 try:  
 stop\_music()  
 time.sleep(1)  
 sel\_song = songplaylistcontainer.curselection()  
 sel\_song = int(sel\_song[0])  
 play\_it = songplaylist[sel\_song]  
 mixer.music.load(play\_it)  
 mixer.music.play()  
 statusbar['text'] = "Playing music" + ' - ' + os.path.basename(play\_it)  
 show\_details(play\_it)  
 except:  
 tkinter.messagebox.showerror('File not found','Aman music player could not find the file. Please select again.')  
def stop\_music():  
 mixer.music.stop()  
 statusbar['text'] = "Music Stopped"  
paused = FALSE  
def pause\_music():  
 global paused  
 paused = TRUE  
 mixer.music.pause()  
 statusbar['text'] = "Music Paused"  
def rewind\_music():  
 play\_music()  
 statusbar['text'] = "Music Rewinded"  
def set\_vol(val):  
 volume = float(val) / 100  
 mixer.music.set\_volume(volume)  
muted = FALSE  
def mute\_music():  
 global muted  
 if muted:  
 mixer.music.set\_volume(0.7)  
 volumeBtn.configure(text="Mute")  
 scale.set(70)  
 muted = FALSE  
 else:  
 mixer.music.set\_volume(0)  
 volumeBtn.configure(text="Volume")  
 scale.set(0)  
 muted = TRUE  
middleframe = Frame(rtframe,bg="#BA8C19")  
middleframe.pack(pady=30, padx=30)  
playBtn = ttk.Button(middleframe, text="Play", command=play\_music)  
playBtn.grid(row=0, column=0, padx=10)  
stopBtn = ttk.Button(middleframe, text="Stop", command=stop\_music)  
stopBtn.grid(row=0, column=1, padx=10)  
pauseBtn = ttk.Button(middleframe, text="Pause", command=pause\_music)  
pauseBtn.grid(row=0, column=2, padx=10)  
bottomframe = Frame(rtframe,bg="gray")  
bottomframe.pack(pady=10,padx=5)  
rewindBtn = ttk.Button(bottomframe, text="Rewind", command=rewind\_music)  
rewindBtn.grid(row=0, column=0,padx=10)  
volumeBtn = ttk.Button(bottomframe, text="Mute", command=mute\_music)  
volumeBtn.grid(row=0, column=1)  
scale = ttk.Scale(bottomframe, from\_=0, to=100, orient=HORIZONTAL, command=set\_vol)  
scale.set(70) # implement the default value of scale when music player starts  
mixer.music.set\_volume(0.7)  
scale.grid(row=0, column=2, pady=15, padx=30)  
def on\_closing():  
 stop\_music()  
 root.destroy()  
root.protocol("WM\_DELETE\_WINDOW", on\_closing)  
if \_\_name\_\_==mainloop():  
 root.mainloop()

**CHAPTER 5**

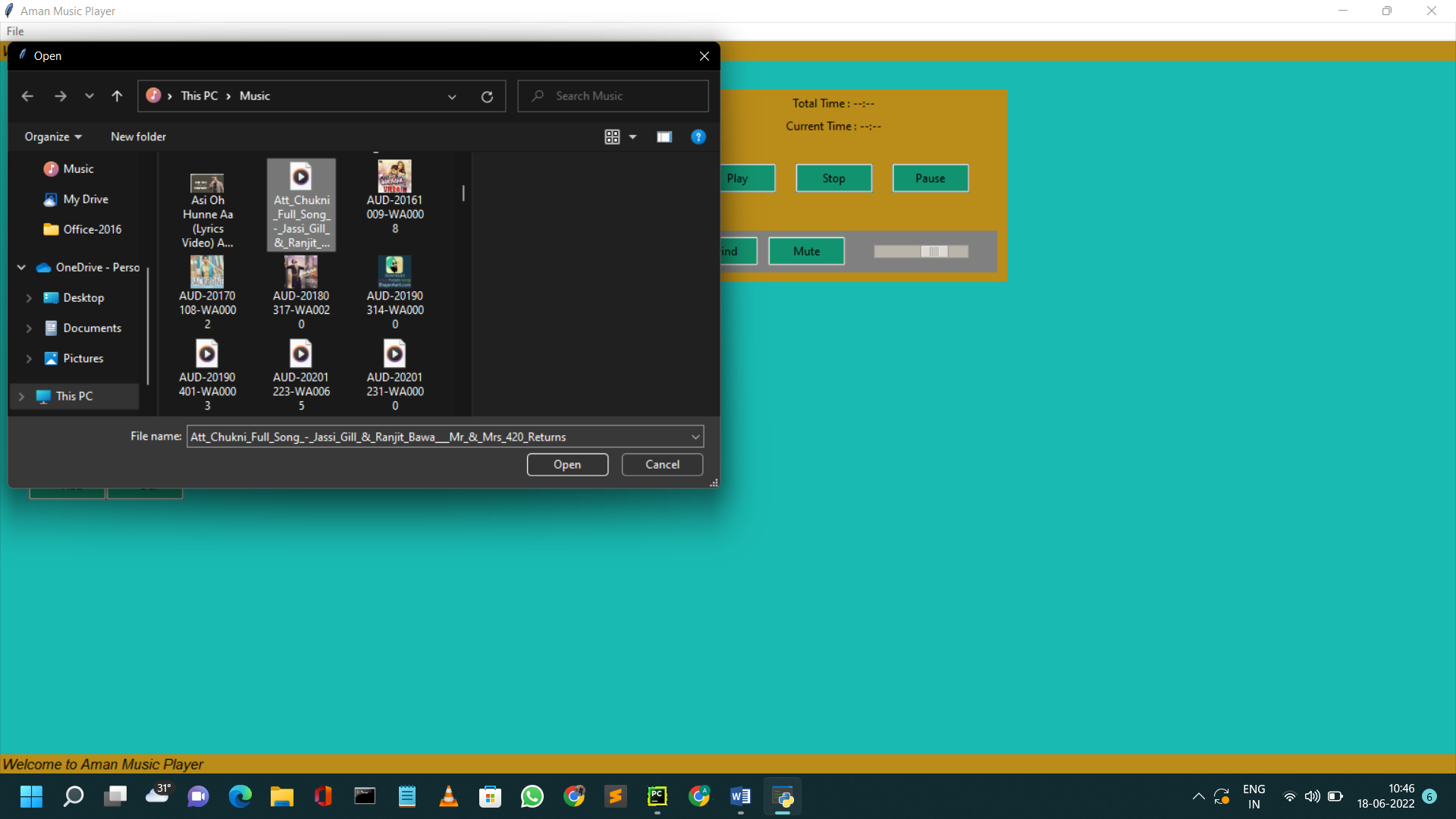
### RESULT & DISCUSSION

#### 5.1 OUTPUT



##### Fig 5.1.1 : Initial player interface

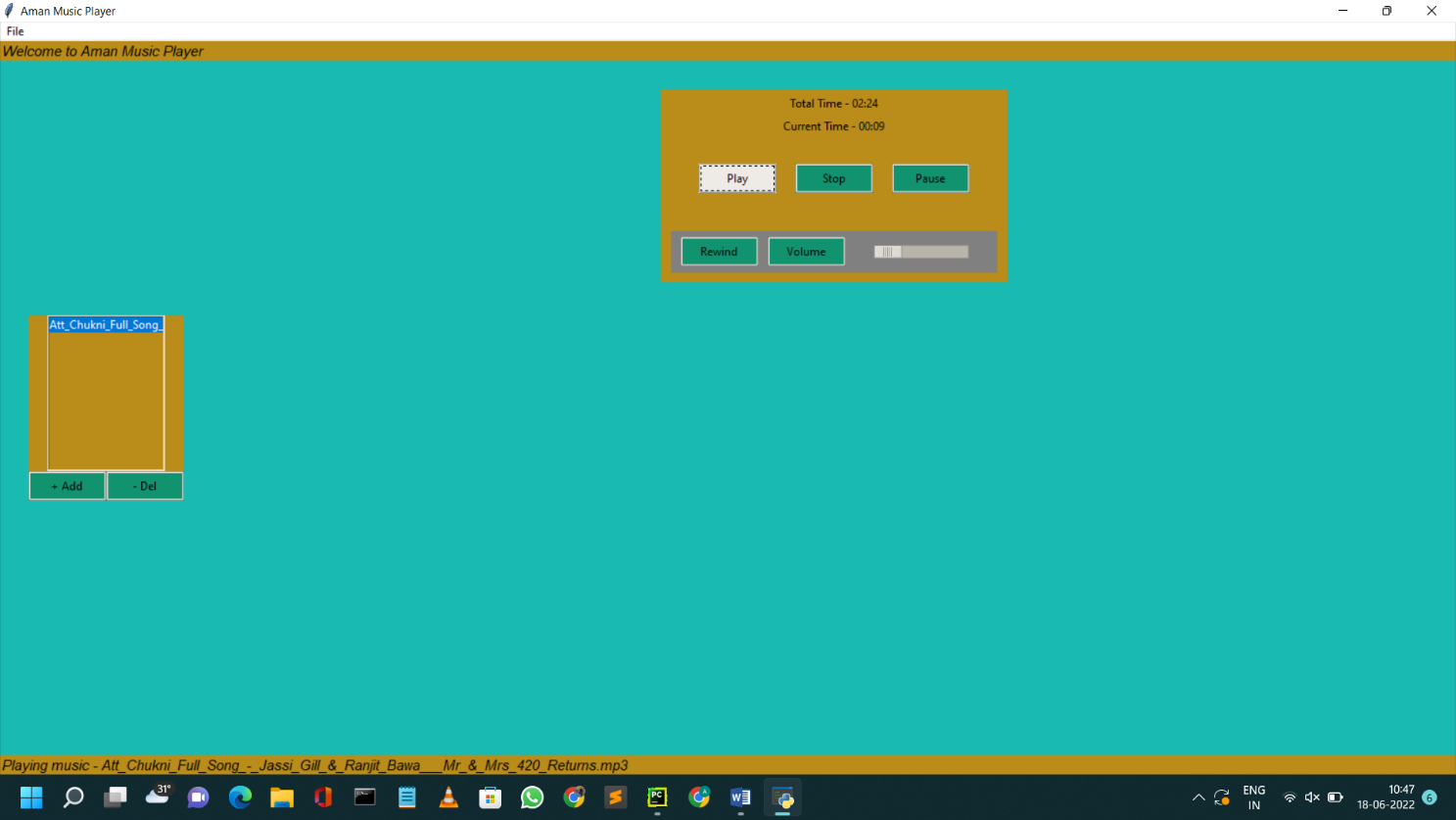
**Fig 5.1.1 :** It shows the Welcome screen to the user as soon as the program starts. And provides the user choice of Song Which user want to add in the playlist.



##### Fig 5.1.2: Importing interface

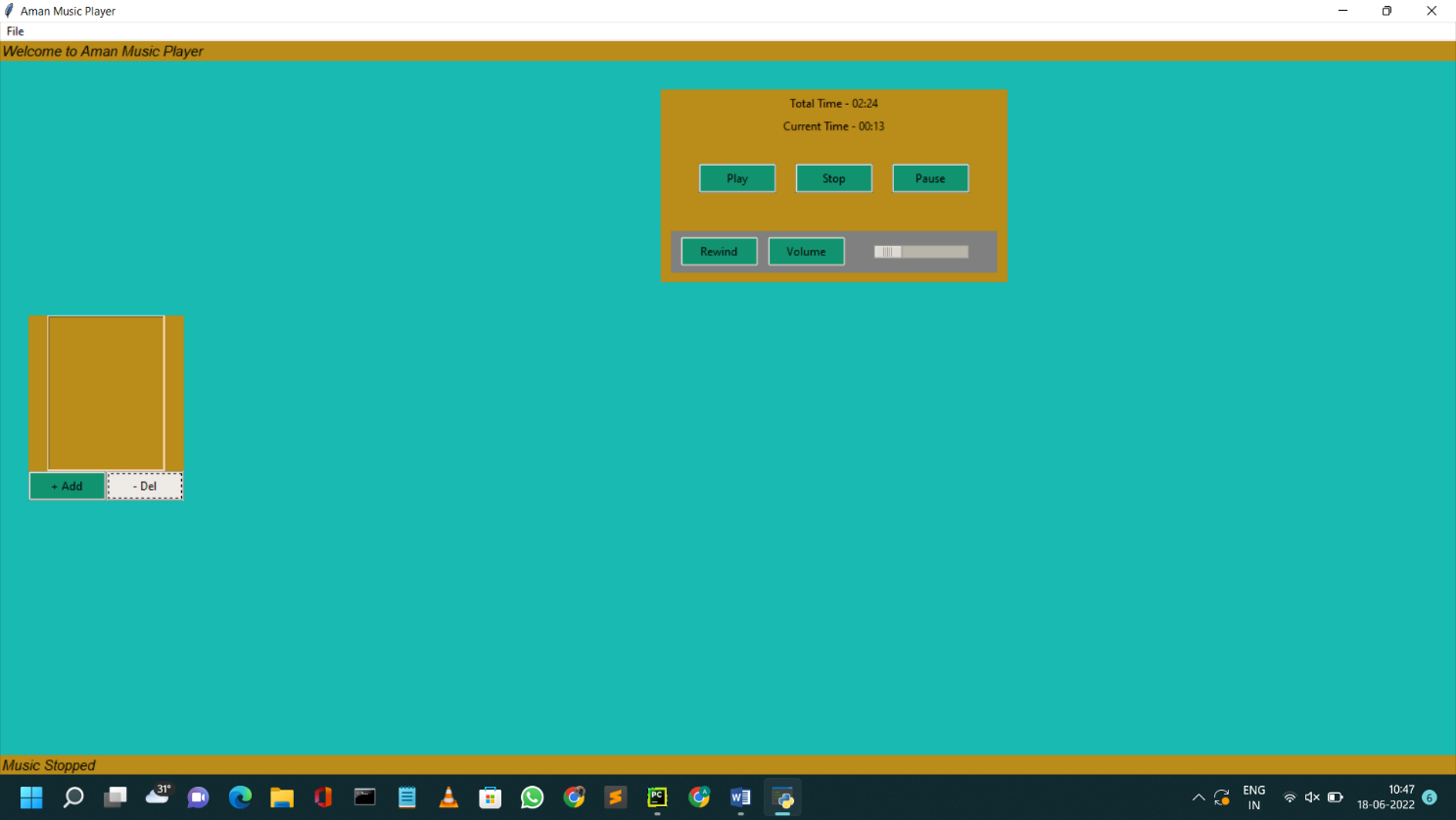
**Fig 5.1.2 :** A brief look of how the interface looks while Importing the song from file manager .

After selecting The song that user wanna play it automatically import to our music player playlist.



##### Fig 5.1.3 : After selecting song

**Fig 5.1.3 :** Now it import on the playlist , now you can select and play the song do you want .



**Fig 5.1.4 : After Deleting the song Fig 5.1.4 :** You can delete the song from playlist and many more .

**5.2 Discussion**

First To create an MP3 player using the Python programming language to play and listen to songs, MP3 files, and other digital audio files. The player must have a simple and easy-to-use interface with options for various functions and a screen around the entire playlist and buttons to turn off the player. The player should be able to play any song. It must be able to play MP3 files or other digital audio files. It should give the user the option to pause or resume the song. the user should be given basic details about the song being played. So here is the discussion about the music player that how it works.

First we take a review what users need basic features in media player like start, stop, Rewind, delete , mute , add etc.

* After the review, we are clear what we have to create
* So we decide to use vscode and pycharm ide.
* We have imported required library’s like (pygame, tkinter, mutagen.mp3, tthemed etc).
* Then we have create a class named MusicPlayer in that we have define various functions to perform functionality like (play, pause, stop, rewind, mute, add song, delete song, volume slider). • Then create a if condition in which we define the window title, geometry, and other functions in it.
* After writing the whole code, we run the program and gui of media player appear.
* Then we add songs in it and play the song.
* The whole program is properly working without any error**.**

**5.3 APPLICATION :**

They are portable digital music players that play music as audio files, such as MP3. In addition, most of these devices allow to store video, pictures, and to receive radio and TV programs

(podcasting). Earphones and external speakers are the typical output devices delivering sound to the listener.

**CHAPTER 6**

**CONCLUSION**

Here I have designed and developed **Music Player**  by applying engineering knowledge which provides an approach in learning or building, interesting and different, apps and games. Have identified and analyzed problem while building the game as it was a whole new experience of studying new modules like os , tkinter , mutagen tthemed and develop a Music player with it. I have used **modern tools** like **VS Code** to implement this project. During the development of the project we have applied **professional ethics** and we understood the importance of **time management** through the whole process of developing the project. While showcasing our project, enhanced my communication skills and displayed professional ethics which results in **lifelong learning.**

MP3 player is a device built to play and listen to digital audio files. These can be either MP3 files or some other audio files. The player was built using Python language. A GUI implementation of the application was developed that is simple and easy to use.The application provides the user with five options — to add song to a playlist, to play the song, to pause , stop the song, to play the previous song and to mute song.

The player also has the capability to add multiple songs to the playlist at the same time. It has a large display area where the playlist is visible.

Once a song is selected and played, we can hear it and can also see details about the song on top of the display. This information includes details about the song such as song time , duration of the song, and at the bottom the song name were displayed.

The Tkinter library of Python was used to create the GUI of the project. It was used to create the option buttons, the label and the display area.

The Pygame and Mutagen library was used to add songs, play the songs, provide pause .

In conclusion, a successful project was built in which songs will play Which you want to play after selecting and the entire playlist will be there for you.

**FUTURE SCOPE**

In future, I can try to apply the knowledge of **machine learning** or **artificial intelligence** in order to fully automate the project by making it like a spotify music player . Also can add **different themes** for the player etc.

**REFERENCE**

**Websites**

* **https://www.pythontutorial.net/tkinter/**
* **https://pypi.org/project/mutagen/**