A
Project Report
On

MULTIMEDIA PROCESSING

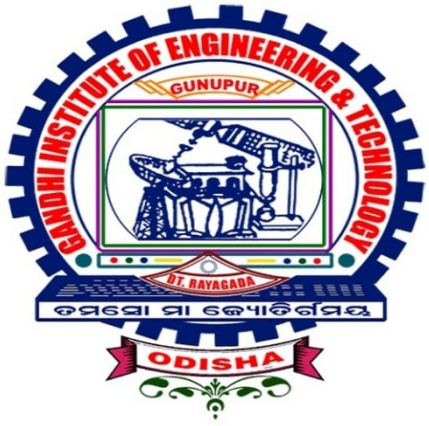
Submitted in partial fulfillment of
the requirements for the 5th Semester Sessional Examination of

BACHELOR OF TECHNOLOGY
IN

COMPUTER SCIENCE & ENGINEERING

By
VV SRIVATSHA

1801210532



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GIET MAIN CAMPUS AUTONOMOUS

GUNUPUR – 765022

2020 - 21

GIET MAIN CAMPUS AUTONOMOUS,

GUNUPUR-765022, Dist: Rayagada (Orissa), India

(Approved by AICTE and Govt, of Orissa)

Ph: 06857 – 250172(Office), 251156(Principal), 250232(Fax),

E-mail: gandhi_giet@yahoo.com visit us at WWW.giet.org

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project work entitled "MULTIMEDIA PROCESSING" is done by Vv Srivatsha-,1801210532. -in partial fulfillment of the requirements for the 5th Semester Sessional Examination of Bachelor of Technology in Computer Science and Engineering during the academic year 2020-21. This work is submitted to the department as a part of evaluation of 5th Semester Project.

Mr Ranjeet Pattanayak
Class Teacher

Prof. (Dr) .Sanjay Kumar Kuanar
HoD, CSE

ACKNOWLEDGEMENT

I would like to express my special thanks and gratitude to my class-teacher "Mr Ranjeet Pattanayak" and my teammates "Subhadip Maity" and "Satyam Kumar" for their able support and guidance in completing our Project.

I would also like to extend my gratitude to our H.O.D Sir "Prof.(Dr.).Sanjay Kumar Kuanar for providing me with all the support that was required.

Date:

21/01/2021

Vv Srivatsha

Sec: A, 5th Sem

ABSTRACT

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.

The MP3 player application should have the capabilities of playing a song, create and display a playlist, pause and resume a long and change the song, that is, play the previous or next song.

INTRODUCTION

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 main objective of this project is to allow users to play MP3 and digital audio files. To be engaging for users, the application has to have a simple but beautiful user interface.

This GUI project is developed using Python programming language. 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 library.

You can have an interface for listing the available MP3 files. The users will also expect the MP3 Player to have an interface that shows information on the file that is playing. Some of the information you can include are the name of the file, its length, the amount played, and the amount not played, in minutes and seconds.

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 and Simpleaudio. 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. To do this, you'll need a database to store information on the created playlists. Python's sqlite3 module allows you to use the SQLite database.

The SQLite database is a better option in this case, because it is file based and easier to set up than other SQL databases. While SQLite is file based, it is better for saving data than a regular file.

PROBLEM DESCRIPTION

1. To build an MP3 player using Python programming language to be able to play and listen to songs, MP3 files .
2. Determine the functionalities of the MP3 player.
3. The player should be have a simple and easy to use GUI with options for various functions, display screen to display the entire playlist and buttons to shut down the player.
4. The player should be able to play any song. It should be capable of playing MP3 files.
5. The player should allow the user to browse through the contents of the computer drive to choose song/s to be played or queued.
6. It should provide the user with option to pause or resume the song.
7. The user should be able to play the previous or the next song in the playlist.
8. Lastly, the user should get basic details about the current playing song. The details can include the song name, singer's name, the duration of the song, size of the file, etc.

ALGORITHM

1.Import the libraries.
2. Create an object of the tkinter and Pygame libraries.
3. Create a window using Tkinter object and create a Frame.
4. Add buttons that provide different functionalities.
    Add a song
    Play the song
    Pause the song
    Play previous song
    Play next song

    Turn on Shuffle
5. Add a song button when pressed should open a dialog box to browse and choose the directory.
6. Add label to display the song's information.
    Name
    Singer
    Duration
    Size of the file, etc.
7. Display screen will display the details of the entire playlist.

8. Add a search bar where user can search for the songs in the playlist.
9. Close button will automatically clear the song list and will stop playing the song

SYSTEM ANALYSIS

User Requirements :

User have to open the mp3 player and have to understand the buttons.
First of all add Songs in the mp3 player and automatically songs will start playing.
User have to use different types of buttons to activate and deactivate functionalities like shuffle, play/pause, stop.
According to the user requirement it must be able to play mp3 files.

Hardware Requirements :

(i) Processor : Intel core (i3,i5,i7,i9)
(ii) Display : LCD
(iii) Accessories : Mouse

Software Requirements :

(i) Operating System : Microsoft Windows 7 or higher
(ii) Application : Jupyter notebook / Spyder
(iii) Programming language : python
(iv) Framework and Libraries : Tkinter, pygame, PIL, OS.

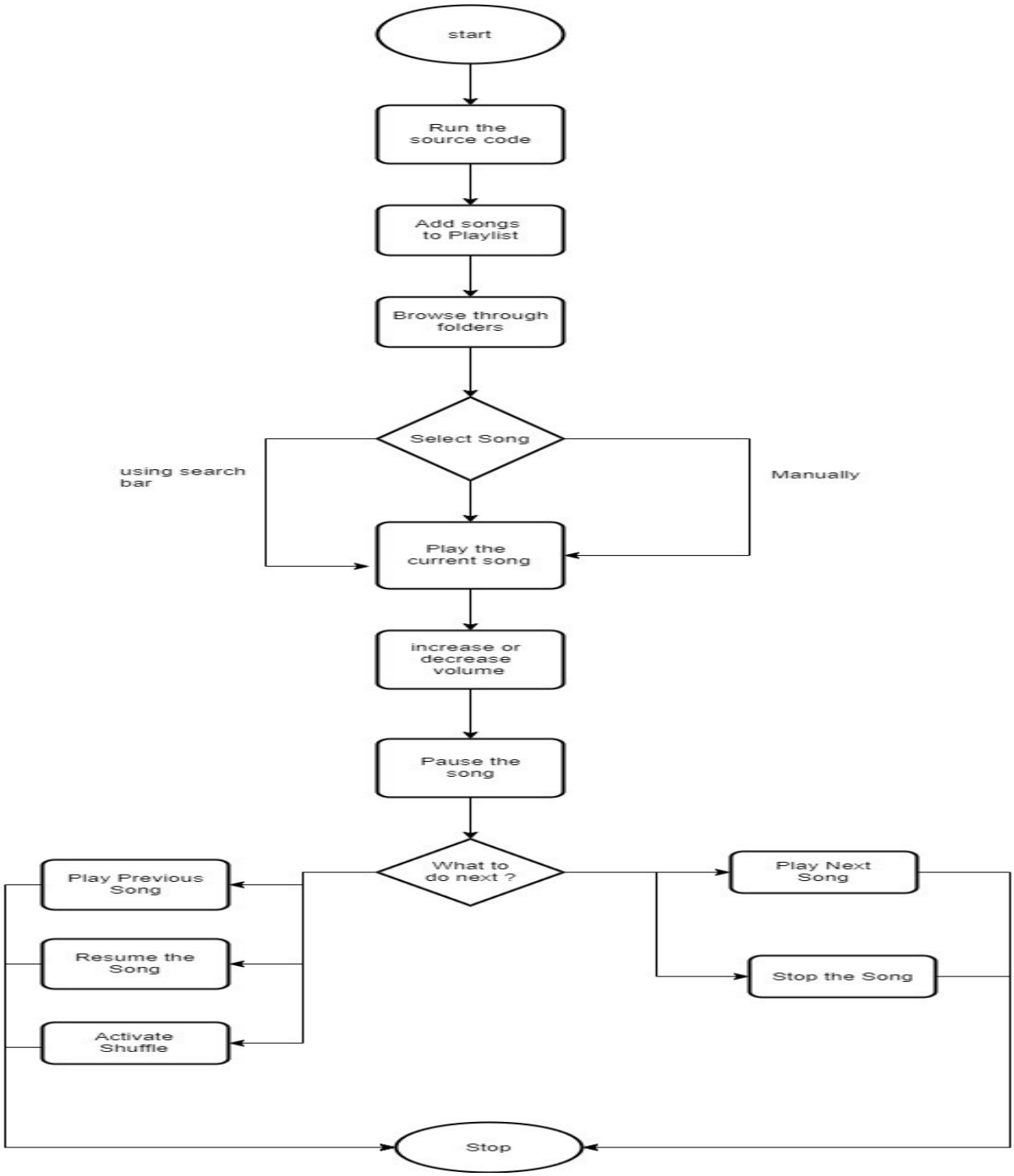
User (Questionnaires) :

> How will the player upload songs and play them ?
> What do we need to install this system?

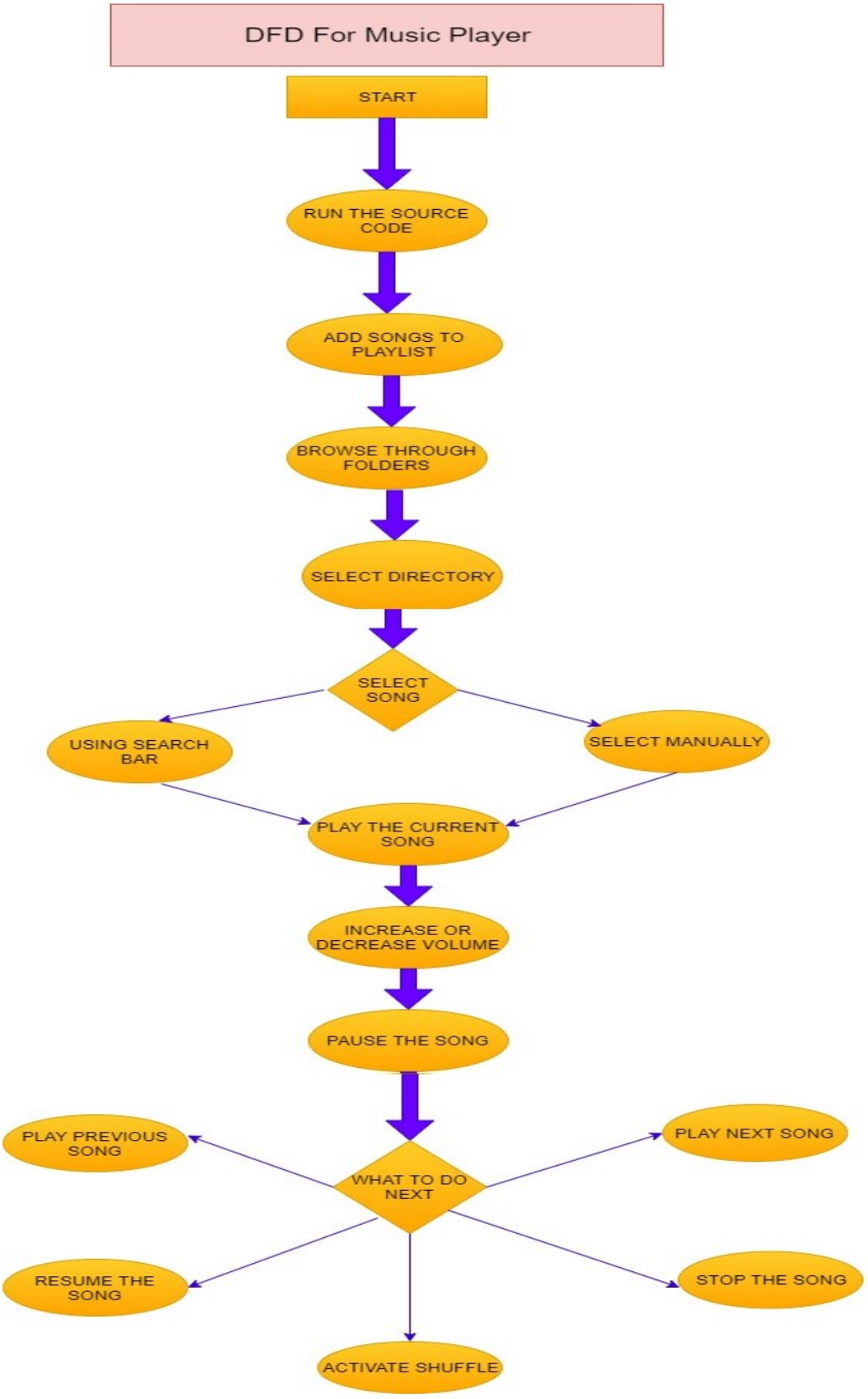
SYSTEM DESIGN AND SPECIFICATIONS

HIGH LEVEL DESIGN(HLD) :

Structure Chart:

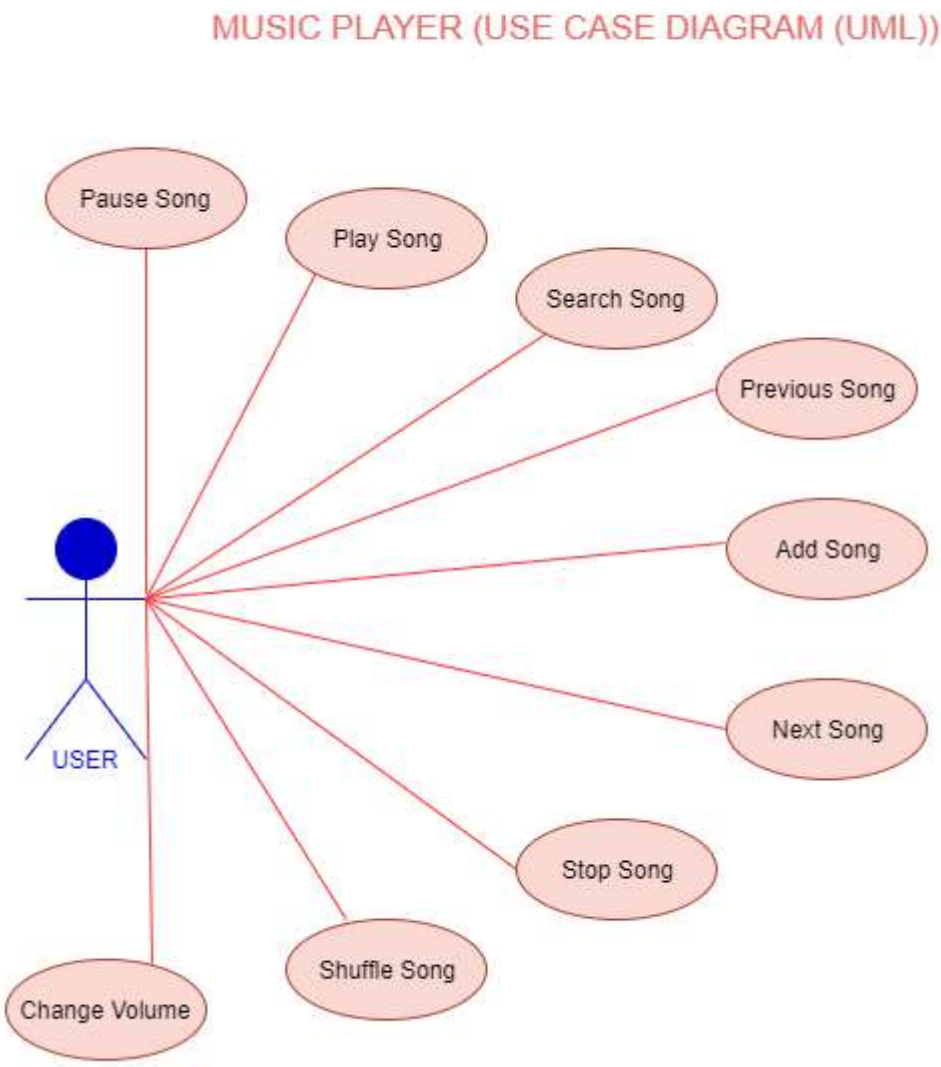


Data Flow Diagram (DFD):



.

UML (Use Case):


MUSIC PLAYER (USE CASE DIAGRAM (UML))
Pause Song
Play Song
Search Song
Previous Song
Add Song
Next Song
Stop Song
Shuffle Song
Change Volume
USER

LOW LEVEL DESIGN (LLD) :

Pseudocode:

- **Library Required:**
  - Tkinter
  - Pygame
  - PIL
  - OS
- **Install and Import the required libraries**.
- **Upload songs directory:**

  dir=askdirectory()
- **Initialize the operation:**

  pygame.mixer.init()
- **Load song :**

  pygame.mixer.music.load(songslist[index])
- **Play Song :**

  pygame.mixer.music.play(0)
- **Pause Song :**

  pygame.mixer.music.pause()
- **Unpause Song**

  pygame.mixer.music.unpause()
- **Stop Song :**

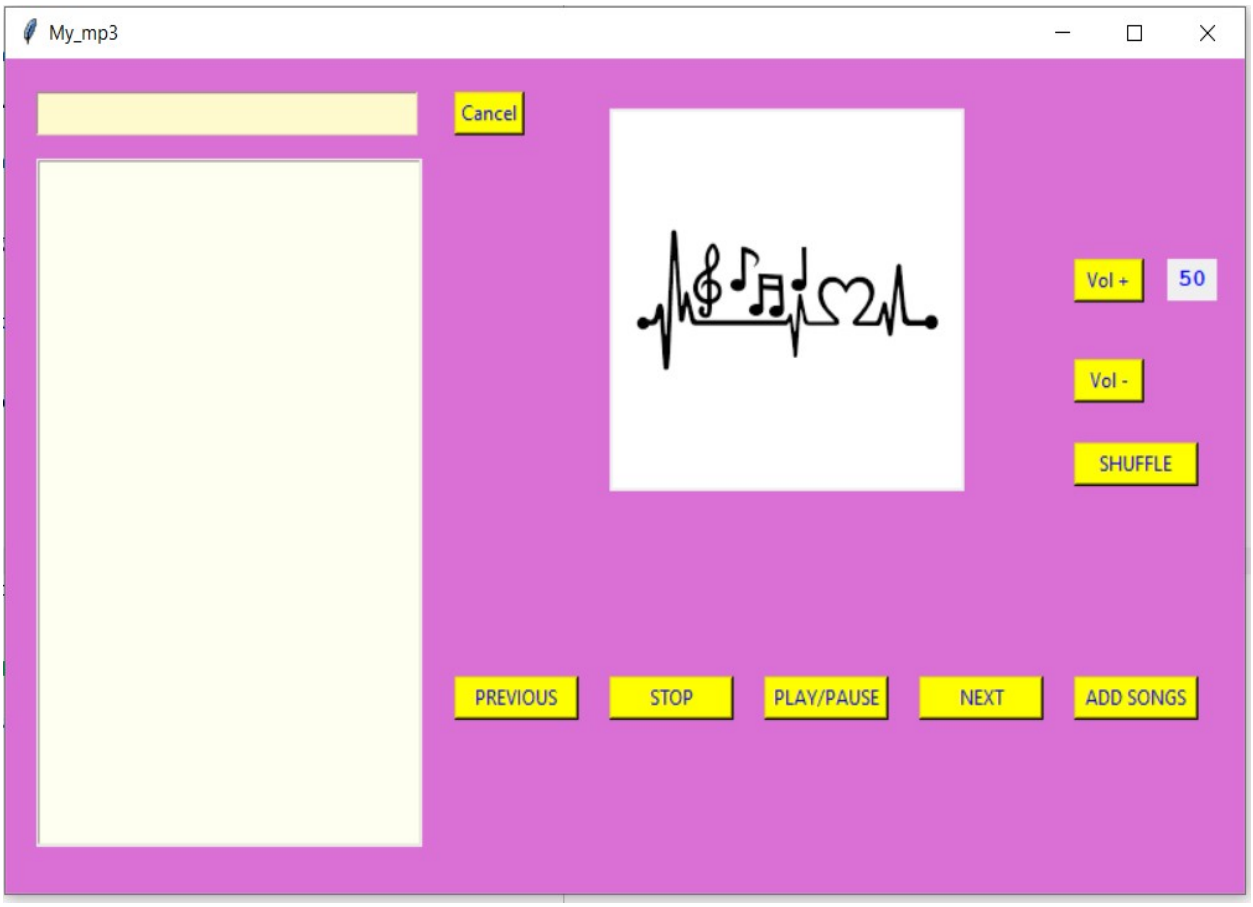
  pygame.mixer.music.stop()

- **Volume change :**

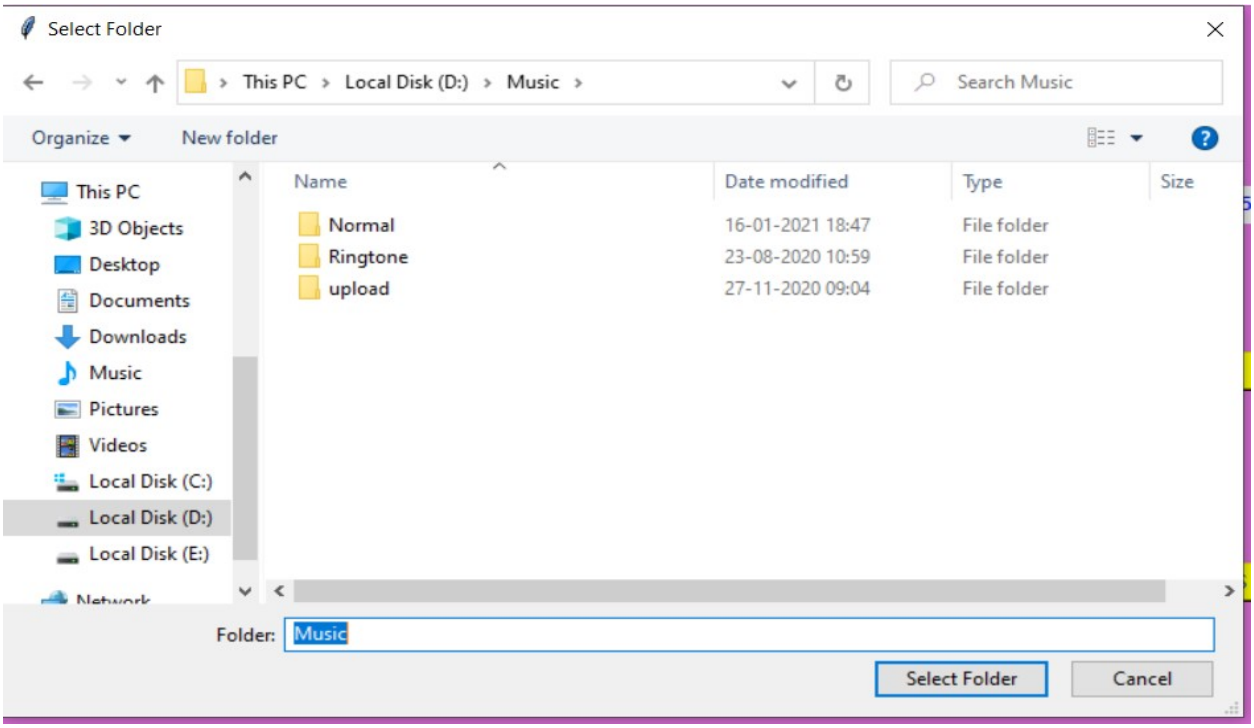
  pygame.mixer.music.set_volume(vol)

SCREENSHOTS:

Home page :

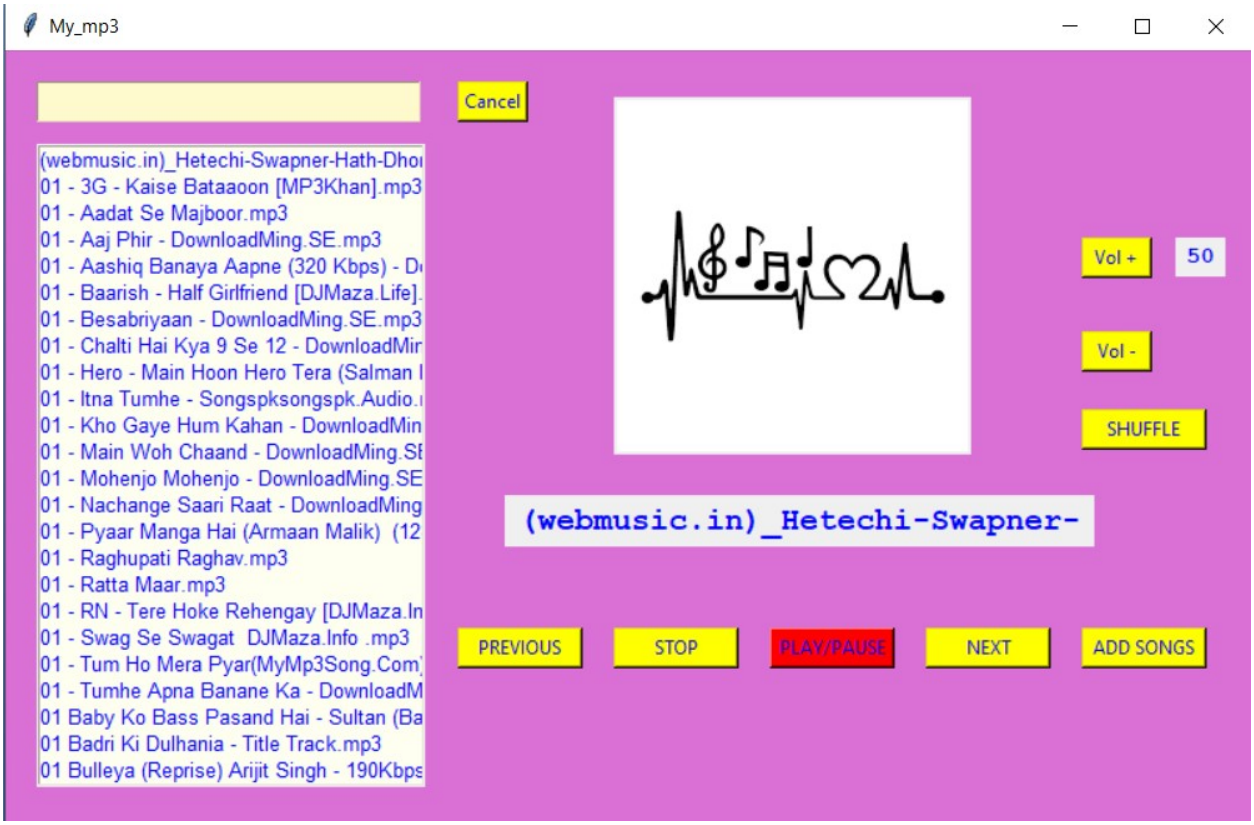


Adding Songs Directory:

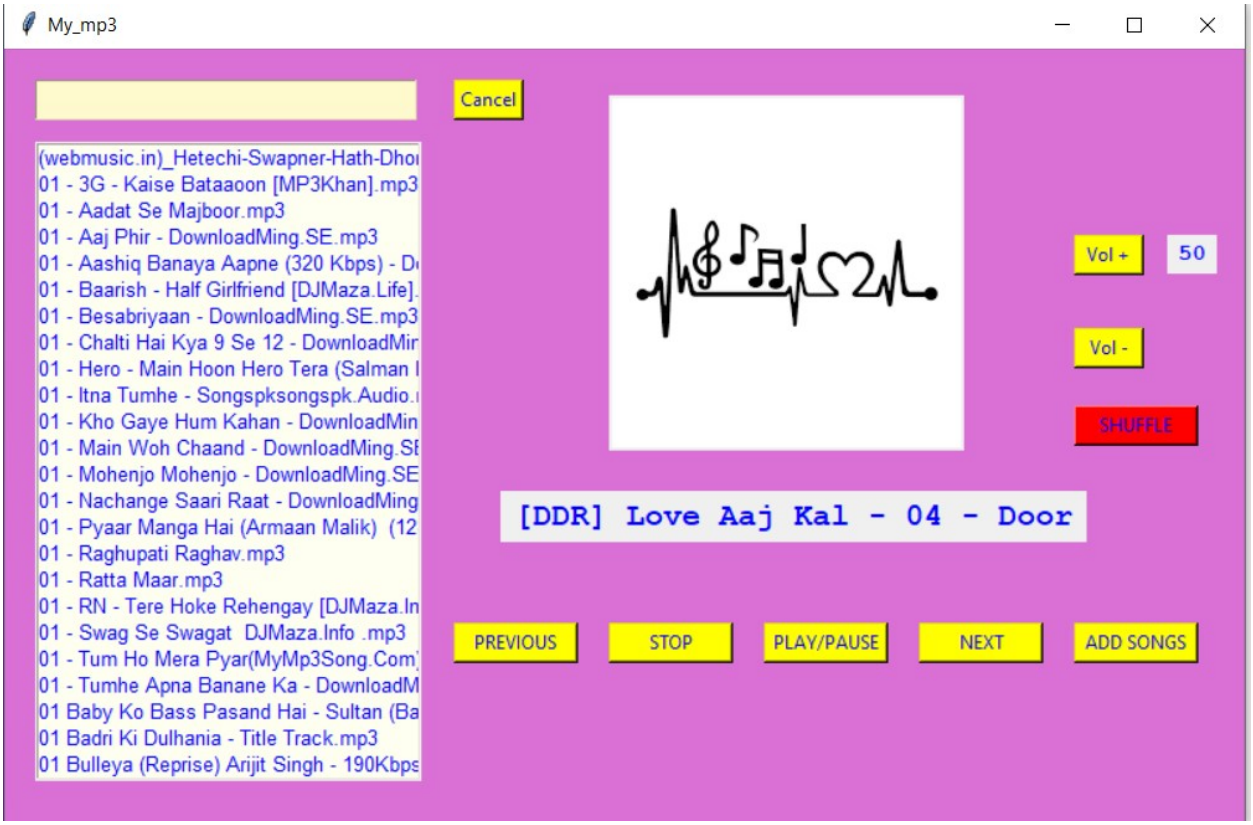


Songs Added and playing:

Paused the song:



Shuffle On:



Click on songs to play:

Searching Songs:

CODING:

Import Libraries:

```
import pygame
import os
from tkinter.filedialog import askdirectory
from tkinter import *
from PIL import Image,ImageTk
import random
```

Initializing all Objects and Variables:

```
root=Tk()
root.title("My_mp3")
f=Frame(root,height=500,width=800,bg="orchid")
f.propagate(0)
f.pack()
img=ImageTk.PhotoImage(Image.open('img3.png'))
l=Label(f,image=img)
l.pack()
l.place(x=390,y=30)
totalsongs=[]
songslist=[]
index=0
stop1=0
shuffle=0
vol=0.5
m1=Message(f,text=str(int(vol*100)),width=200,font=('Courier',-15,'bold'),fg='blue')
m1.place(x=750,y=120)
lb=Listbox(f,font=("Arial 10"), fg="blue", bg="ivory", height=24, width=35,
selectmode=SINGLE)
lb.place(x=20,y=60)
e1=Entry(f,width=22,fg='firebrick',bg='LemonChiffon',font=('Arial',14))
e1.place(x=20,y=20);
p1=0
```

Function to Add songs from Directory:

```
def add_songs():
    dir=askdirectory()
    os.chdir(dir)
    for file in os.listdir(dir):
        if file.endswith(".mp3"):
            songslist.append(file)
            totalsongs.append(file);
    for i in songslist:
        lb.insert(END,i)
    play(0)
```

Load Songs in the FILE to play:

```
def load_songs(event):
    global index
    songslist.clear()
    lb.delete(0,END);
    str1=e1.get()
    str1=str1.lower()
    n=len(str1)

    for i in range(0,len(totalsongs)):
        f=0
        s=totalsongs[i].lower()

        for j in range(0,10-n):
            if(s[j]==str1[0]):
                f=1
                for k in range(1,n):
                    if(s[k+j]!=str1[k]):
                        f=0
                        break

            if(f==1):
                songslist.append(totalsongs[i])
                break
```

```
    if(songslist):
        for i in songslist:
            lb.insert(END,i)
        index=0;
        pygame.mixer.music.stop()
        play(0)
    else:
        lb.insert(END,"NO SONGS FOUND")

e1.bind("<Return>",load_songs);
```

Select from song list:

```
def on_select(event):
    global index
    pos=lb.curselection()
    index=pos[0]
    playselect(index)
```

Playing Sequentially:

```
def queue():
    global stop1
    global shuffle
    global index
    global vol
    pos=pygame.mixer.music.get_pos()
    if stop1==0:
        if pos==-1:
            if(shuffle%2==0):
                index+=1
                if(index==len(songslist)):
                    index=0
                pygame.mixer.music.load(songslist[index])
                pygame.mixer.music.set_volume(vol)

                m=Message(f,text=songslist[index][:30],width=450,font=('Courier',-
20,'bold'),fg='blue')
```

```
            m.place(x=310,y=285)
            pygame.mixer.music.play(0)
        else:
            index=random.randint(0,len(songslist)-1)
            pygame.mixer.music.load(songslist[index])
            pygame.mixer.music.set_volume(vol)
            m=Message(f,text=songslist[index][:30],width=450,font=('Courier',-
20,'bold'),fg='blue')
            m.place(x=310,y=285)
            pygame.mixer.music.play(0)

    pygame.mixer.music.set_volume(vol)
    if stop1==0:
        root.after(1000,queue)

lb.bind('<<ListboxSelect>>',on_select)
```

Play Selected song from list:

```
def playselect(index):
    b4["bg"]="yellow"
    b3["bg"]="yellow"
    global stop1
    global p1
    global shuffle
    global vol
    stop1=0
    p1=0
    pygame.mixer.init()
    pygame.mixer.music.load(songslist[index])
    pygame.mixer.music.set_volume(vol)
    m=Message(f,text=songslist[index][:30],width=450,font=('Courier',-20,'bold'),fg='blue')
    m.place(x=310,y=285)
    pygame.mixer.music.play(0)
    queue()
```

Playing the song:

```
def play(index):
    b4["bg"]="yellow"
    b3["bg"]="yellow"
    global stop1
    global p1
    global shuffle
    global vol
    stop1=0
    p1=0
    if shuffle%2==0:
        pygame.mixer.init()

        pygame.mixer.music.load(songslist[index])
        pygame.mixer.music.set_volume(vol)
        m=Message(f,text=songslist[index][:30],width=450,font=('Courier',-
20,'bold'),fg='blue')
        m.place(x=320,y=285)
    else:
        pygame.mixer.init()
        index=random.randint(0,len(songslist)-1)
        pygame.mixer.music.load(songslist[index])
        pygame.mixer.music.set_volume(vol)
        m=Message(f,text=songslist[index][:30],width=450,font=('Courier',-
20,'bold'),fg='blue')
        m.place(x=320,y=285)
    pygame.mixer.music.play(0)
    queue()
```

Stop Song:

```
def stop():
    global stop1
    pygame.mixer.music.stop()
    b3["bg"]="red"
    stop1=1
```

Next Song:

```
def nextsong():
    global index
    if index==len(songslist)-1:
        index=0
    else:
        index+=1
    play(index)
```

Previous Song:

```
def presong():
    global index
    if index==0:
        index=len(songslist)-1
    else:
        index-=1
    play(index)
```

Play/Pause Song:

```
def playpause():
    global p1
    p1+=1
    if p1%2==1:
        pygame.mixer.music.pause()
        b4["bg"]="red"
    else:
        pygame.mixer.music.unpause()
        b4["bg"]="yellow"
```

Controlling Button Actions:

```
def click(num):
    global shuffle
    global vol
    global index
    if num==1:
        presong()
    elif num==2:
        nextsong()
    elif num==3:
        stop()
    elif num==4:
        playpause()
    elif num==5:
        lb.delete(0,END)
        add_songs()
    elif num==6:
        shuffle+=1
        if shuffle%2==1:
            b6["bg"]="red"
        else:
            b6["bg"]="yellow"
    elif num==7:
        if vol<1.0:
            vol+=0.1

        if vol>1.0:
            vol=1.0
        m1=Message(f,text=str(int(vol*100)),width=200,font=('Courier',-15,'bold'),fg='blue')
        m1.place(x=750,y=120)
    elif num==8:
        if vol>0.0:
            vol-=0.1
        if vol<0.0:
```

```
            vol=0.0
        m1=Message(f,text=str(int(vol*100)),width=200,font=('Courier',-15,'bold'),fg='blue')
        m1.place(x=750,y=120)
    elif num==9:
        lb.delete(0,END)
        songslist.clear()
        for i in totalsongs:
            songslist.append(i)
        for i in songslist:
            lb.insert(END,i)
        index=0
        pygame.mixer.music.stop()
        play(0)
```

Initialization of Buttons:

```
b1=Button(f,text='PREVIOUS',width=10,height=1,bg='yellow',fg='blue',activebackground
='red',activeforeground='black',command=lambda:click(1))
b2=Button(f,text='NEXT',width=10,height=1,bg='yellow',fg='blue',activebackground='red'
,activeforeground='black',command=lambda:click(2))
b3=Button(f,text='STOP',width=10,height=1,bg='yellow',fg='blue',activebackground='red'
,activeforeground='black',command=lambda:click(3))
b4=Button(f,text='PLAY/PAUSE',width=10,height=1,bg='yellow',fg='blue',activebackgrou
nd='red',activeforeground='black',command=lambda:click(4))
b5=Button(f,text='ADD
SONGS',width=10,height=1,bg='yellow',fg='blue',activebackground='red',activeforegrou
nd='black',command=lambda:click(5))
b6=Button(f,text='SHUFFLE',width=10,height=1,bg='yellow',fg='blue',activebackground=
'red',activeforeground='black',command=lambda:click(6))
b7=Button(f,text='Vol
+',width=5,height=1,bg='yellow',fg='blue',activebackground='red',activeforeground='blac
k',command=lambda:click(7))
b8=Button(f,text='Vol -
',width=5,height=1,bg='yellow',fg='blue',activebackground='red',activeforeground='black'
,command=lambda:click(8))
```

```
b9=Button(f,text='Cancel',width=5,height=1,bg='yellow',fg='blue',activebackground='red',
activeforeground='black',command=lambda:click(9))
```

Packing Buttons in Frame:

```
b1.pack()
b2.pack()
b3.pack()
b4.pack()
b5.pack()
b6.pack()
b7.pack()
b8.pack()
b9.pack();
```

Placing Buttons in Frame co-ordinate:

```
b3.place(x=390,y=370)
b1.place(x=290,y=370)
b4.place(x=490,y=370)
b2.place(x=590,y=370)
b5.place(x=690,y=370)
b6.place(x=690,y=230)
b8.place(x=690,y=180)
b7.place(x=690,y=120)
b9.place(x=290,y=20)

root.mainloop()
```

CONCLUSION

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 many options like — to add song to a playlist, to play the song, to pause or resume the song, to play the previous song and to play the next song, shuffle and volume up down and search songs in the search bar.

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 display. This information includes details about the song name.

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 library is used to add songs, play the songs, provide pause and resume and other options.

In conclusion, a successful project was built in which songs will play one after the other automatically and the entire playlist will play all over again once concluded.

BIBLIOGRAPHY

Following are the sources referred to complete this project successfully:

>Core Python Programming, Dr. R. Nageswar Rao
> pygame.org
> stackoverflow.com

---O---