# ***"Music Player"***

Mini Project Report
Open Source Technology Lab

Submitted by


| | |
|---|---|
| Mr. Sushil Dubey | 18CE2011 |
| Mr. Parth Gujar | 18CE2019 |
| Ms. Prithvi Dambal | 18CE2007 |
| Ms. Samiksha Varpe | 18CE2030 |


Guided by

Sumithra T.V



Department of Computer Engineering

Ramrao Adik Institute of Technology

Dr. D. Y. Patil Vidyanagar,Sector 7, Nerul, Navi Mumbai 400 706**.**

(Affiliated to University of Mumbai**)**


**April – 2020**

# *Index*

# ABSTRACT

A software program or hardware device capable of playing a media file or disc. For example, many media players today are capable of playing audio files such as playing an MP3 song file and video files such as a short video clip or movie. Below is a list of some of our favorite media players. The image is a visual example of the Microsoft Windows **Media player**.

This means that a phone includes memory storage for MP3, AAC or similar music files, and software for playing that music. Generally, music can be downloaded into the phone from a computer and played back later through a headset attached to the phone.

This is not to be confused with real-music ringtones, which are usually a distinct feature. A "music player" feature can play whole songs, independently of the ringer functionality.

Newer phones with High-Speed Data may support downloading music directly over the mobile network.

Hardware or software that plays audio files encoded in MP3 and other audio formats. On the software side, applications that reside in the user's computer, such as iTunes, Windows Media Player and RealPlayer, are used to organize a music collection, play audio files and rip music from a CD. Software players may also provide access to Internet radio stations and other streaming audio sites.

Music player, mp3 player app is a best media player for your mobile. This app find all song formats quickly. You can view music by some convenient categories: song title, artist, album. Music player gives high quality sound and better experience. If you want to change music information and optimize its size, "Music player" is the best choice. Music player app can play all formats of music files. Such as: MP3, WAV, MP4, FLAC, 3GP, OGG, etc. Because mp3 format is most popular song format in android, so we also call this as mp3 player.
Music player helps you listen to music everywhere, play any favourite songs with high quality. Music player scans all music automatically and group them by title, artist, album. Easy to find the song you want. Supports audio equalizer to improves music sound. Listen to songs on "Music player" every day is a smart choice.

# Chapter 1
# Introduction

Music Player Introduction

Music is a vital part of daily living, as Albert Einstein stated, “life without playing music would be inconceivable”. There is no one living in this earth who does not listen to any kind of music.

There are many types of music, including popular music, traditional music, art music, and music written for religious ceremonies. It ranges from the strictly organized compositions such as classical music, through the improvisational music like jazz etc., and the contemporary music from the 20th and 21st centuries.
Music can be divided into genres such as country music, rock music, pop, classical, etc.; and genres can be further divided into sub-genres. Music has many different fundamentals or elements, these can include: pitch, beat or pulse, tempo, rhythm, melody, harmony, texture, style, allocation of voices, timbre or colour, dynamics, expression, articulation, form, and structure.

Within the arts, music may be classified as a performing art, a fine art or as an auditory art. It may be played or sung and heard live at a rock concert or orchestra performance, as part of a dramatic work such as an opera, or it may be recorded and listened to on a radio, MP3 player, CD player, smartphone or as a film score or TV show. Music is composed and performed for many purposes, ranging from aesthetic pleasure, religious or ceremonial purposes, or as an entertainment product for the marketplace.

Music, without question, plays an important role in people’s way of life–in religious rituals, ceremonies like graduation or marriage, social gatherings and other cultural activities. People may make music as a hobby, or work as a professional musician or singer.

The brief of the technologies used for the system are as follows:
Language: Python 3.8.1 (64-bit)
IDE’s Involved : IDLE 3.8/Pycharm
Thus, the overview of the functioning of the system is given along with the main modules of the system.

## Introduction to Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited to many tasks. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural,) object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive
standard libraries.

## SPYDER

Spyder is an open source cross-platform integrated development environment for scientific programming in the Python language**.** It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

## IDLE

**IDLE** (short for Integrated Development Environment or Integrated Development and Learning Environment) is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1. It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and the **Tkinter** GUI toolkit (wrapper functions for Tcl/Tk).IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.

## GUI (Graphical user Interface) IN PYTHON

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications.

**The important libraries and methods used while developing the system are been briefed below:**

## 1) TKINTER

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface
to the Tk GUI toolkit, and is Python's standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The

name **Tkinter** comes from **Tk interface**. Tkinter was written by Fredrik Lundh. As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application. Python 2.7 and Python 3.1 incorporate the "themed Tk" ("ttk") functionality of Tk 8.5. This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter).
Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. Button : The Button widget is used to display buttons in your application.

- **Frame**

The Frame widget is used as a container widget to organize other widgets. Its important for grouping and organizing all the widgets of window in a effective manner.

- **Label**

The Label widget is used to provide a single-line caption for other widgets. It can also contain images. This widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want.

- **Entry**

The Entry widget is used to provide the single line text-box to the user to accept a value from the user. We can use the Entry widget to accept the text strings from the user. It can only be used for one line of text from the user.

- **Button**

These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.
All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

· The **pack()** Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.

· The **grid()** Method − This geometry manager organizes widgets in a table-like structure in the parent widget.

· The **place()** Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

## 2) SQLite

The database management system is the software that interacts with end users, applications, and the database itself to capture and analyse the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system".
SQLite in general, is a server-less database that can be used within almost all programming languages including Python. Server-less means there is no need to install a separate server to work with SQLite so you can connect directly with the database.

## 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. Pygame uses the Simple Direct Media Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.
Other features that SDL doesn't have include vector math, collision detection, 2d sprite scene graph management, MIDI support, camera, pixel-array manipulation, transformations, filtering, advanced free type font support, and drawing. Applications using Pygame can run on Android phones and tablets with the use of Pygame Subset for Android (pgs4a). Sound, vibration, keyboard, and accelerometer are supported on Android.

## 4) Mutagen

Quod Libet has more strenuous requirements in a tagging library than most programs that deal with tags. Therefore, we felt it was necessary to write our own.
Mutagen has a simple API, that is roughly the same across all tag formats and versions and integrates into Python’s built-in types and interfaces.
New frame types and file formats are easily added, and the behaviour of the current formats can be changed by extending them.
Freeform keys, multiple values, Unicode, and other advanced features were considered from the start and are fully supported.
All ID3v2 versions and all ID3v2.4 frames are covered, including rare ones like POPM or RVA2.
We take automated testing very seriously. All bug fixes are committed with a test that prevents them from recurring, and new features are committed with a full test suite.

# Chapter 2
# Problem Analysis

Python application on kernel terminals also completely broke the traditional understanding of the terminals. And appreciate music is one of the best ways to relieve pressure in stressful modern society life. Therefore, many kinds of music players are also developed. However, a lot of players devote to fancy appearance and function, while caused resources wasting to the user's resources, such as large required memory and CPU, which brings a lot of inconvenience as multiple programs running at the same time. For the most ordinary users, many functions are useless. The purpose of this project is to develop a player which can play the mainstream music file format. To browse and query the storage space as well as operation of adding, deleting, and playing can be realized. Meanwhile, this software can play, pause and select songs with latest Btn and next Btn according to users' requirement as well as set up songs' order and etc... Music player based on Python application is popular in the market at the present. The completing development of system gives developers a nice platform, which can learn the popular computer technology combining with learned knowledge, and master the latest knowledge, enrich oneself, and enjoy entertainment.

**Functional Requirements**

1. **Creating a new playlist**: - To create a new playlist the user as to enter the name of the playlist and press new playlist button to add new playlist. If there is no playlist existing with same name a new playlist is added to the database.
2. **Loading music from existing playlist**: - If the user wishes to play songs from the any of the already existing playlist the user a load the songs of the playlist by entering the name of the playlist.

3. **Adding song to the playlist**: - If the user wishes to add new music to the playlist then the user can do so by using the add button. The file path destination of the song is stored in the database and the song is loaded in the playlist.
4. **Deleting song from playlist: -** If the user wishes to remove a song from the playlist the user can easily do so by delete button. The user needs to select the song to be removed and then press the delete button to remove it from the playlist as well as the current playing sequence
5. Pause function: - The ongoing music can be paused using the pause button. When the play button is pressed after pause the song continues to play from the same point it was paused.
6. Stop function: - The user can stop the on-going playing music by using the stop button. When stop button is pressed the current playing song is stopped and the music counter is reset.
7. Volume Control: - The user can easily control the loudness at which the music is played by using the volume control slider. The maximum output of the slider is 100%.
8. Mute: - This feature allows to reduce the volume of the music to be zero.

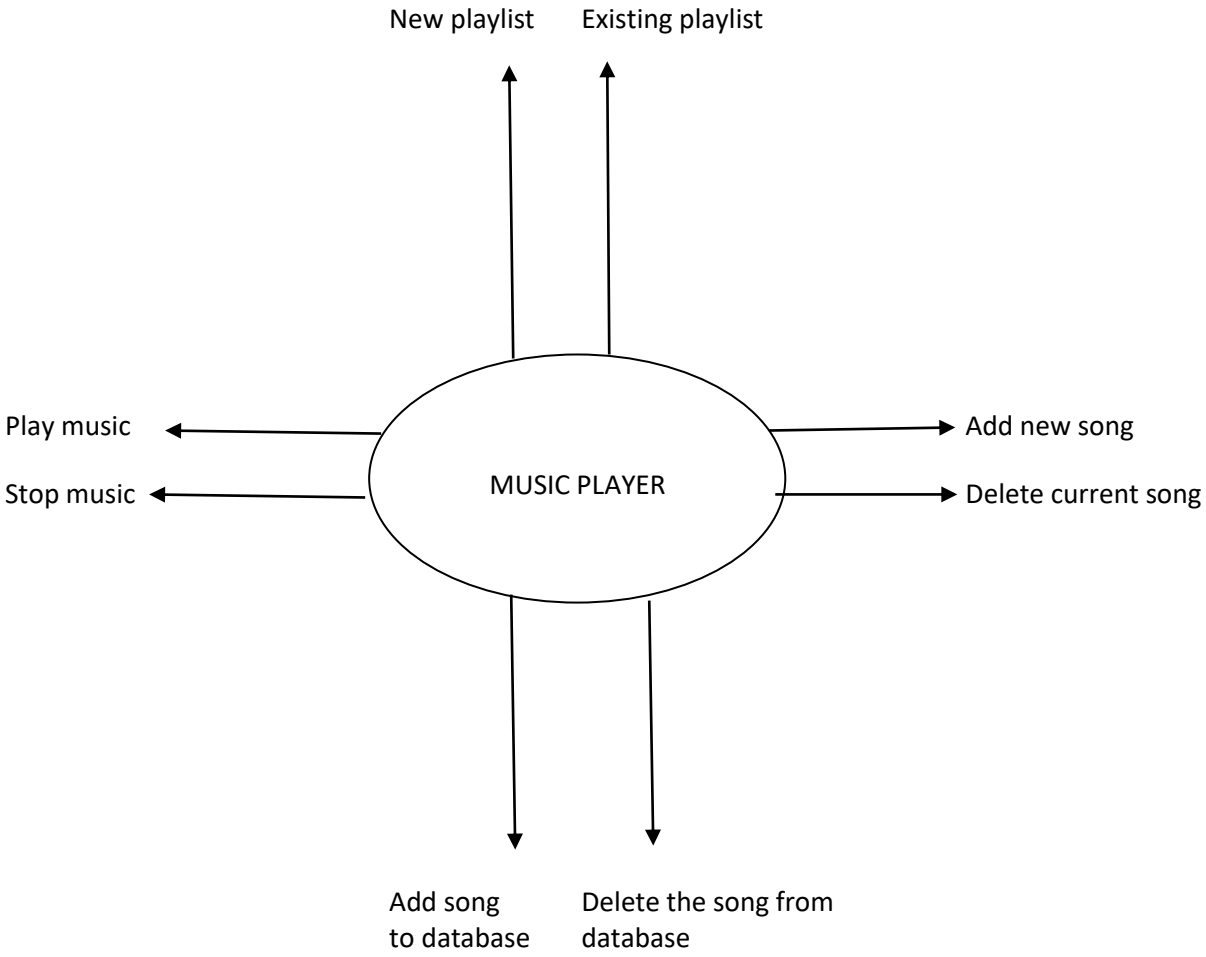
**Non- Functional Requirements**

1. The music can be played in the background without interruption and allows to use all other application simultaneously.
2. The program does not use to much resources to slow down the computer.
3. The program should be able to show the list of the already existing playlist.
4. The program should be able to show the current list of the songs.
5. The program should allow the user to switch songs according.

# Chapter 3

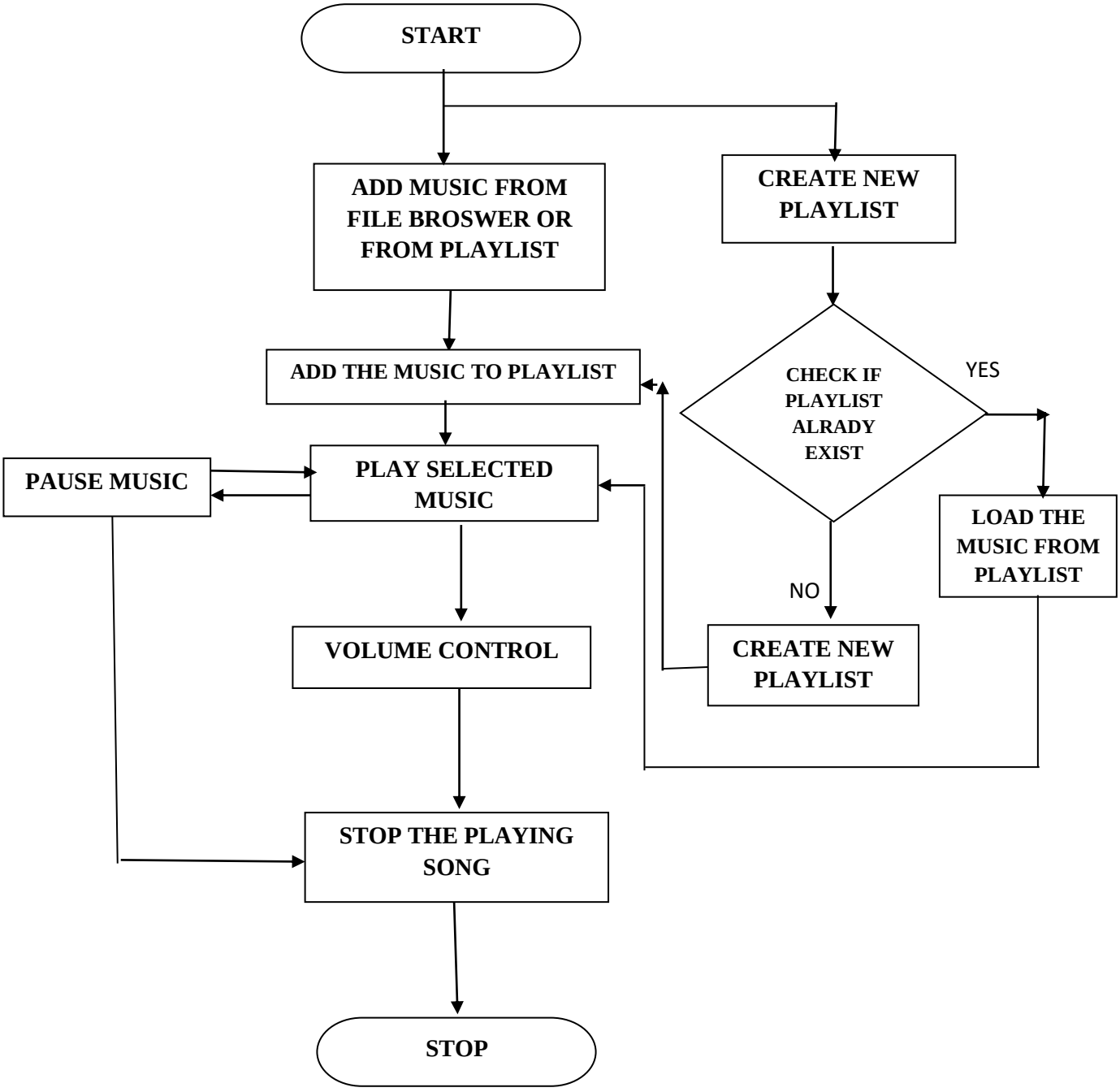
# Design

The dataflow diagram and explanations are shown as follows:




The data flow for music player system is shown in the above diagram.
All the various aspects of the system and their flow of data is shown with the help of the diagram. This data flow diagram also provides the information about outputs and inputs of each entity and the process itself. The project requires one databases one to store all the tables which are have path to the music stored.

## Flow Diagram



# Chapter 4

# Implementation

```
import os
import threading
from tkinter import *
from tkinter import filedialog
from mutagen.mp3 import MP3
import tkinter.messagebox
import time
from tkinter import ttk
from ttkthemes import ThemedTk as tk
from pygame import mixer
import sqlite3
conn=sqlite3.connect('users.db')
c=conn.cursor()
# ---- Initialization -------------
root = tk(theme='equilux')
root.get_themes()
root.set_theme("arc")
playlist = []
# --- Menubar ---
menubar = Menu(root)
root.config(menu=menubar)
# -------------------
# creating submenu
submenu = Menu(menubar, tearoff=0)

def listofplaylist():
    query1="SELECT * FROM list"
    c.execute(query1)
    playlist_list=c.fetchall()
    i=0
    length=len(playlist)
    # while length :
    #     playlist_list_box1.delete(length)
    #     #playlist_list_box1.pop(length)
    #     length=length-1

    for item in playlist_list:
        name=item[0]
        playlist_list_box1.insert(i,name)
```

```
        i=i+1

def add_to_playlist(f):
    f = os.path.basename(browseFile.fileName)
    index = 0
    playlistbox.insert(index,f)
    playlist.insert(index,browseFile.fileName)
    index = index + 1

def browseFile():
    browseFile.fileName = filedialog.askopenfilename()
    add_to_playlist(browseFile.fileName)
    insertintoplaylist()

def loadfromdb():
    user_name = userentry.get()
    sql="SELECT * FROM "+user_name
    try:
        c.execute(sql)
    except:
        tkinter.messagebox.showerror(title="ERROR", message="No Playlist
found")
    songlist=c.fetchall()
    i=0
    while i<len(songlist) :
        browseFile.fileName=songlist[i][0]
        add_to_playlist(browseFile.fileName)
        playlist=songlist[i][0]
        i+=1

def newplaylist():
    user_name=userentry.get()
    sql="CREATE TABLE "+user_name+' ("filepath"       TEXT)'
    sql2="INSERT INTO list(playlist) VALUES (?)"
    c.execute(sql)
    c.execute(sql2,(user_name,))
    conn.commit()
    listofplaylist()

def insertintoplaylist():
    user_name=userentry.get()
    sql="INSERT INTO "+user_name+"(filepath) VALUES (?)"
    c.execute(sql,(browseFile.fileName,))
```

```
    conn.commit()


def del_song():
    selected_song = playlistbox.curselection()
    selected_song = int(selected_song[0])
    playlistbox.delete(selected_song)
    playlist.pop(selected_song)
#### future work delete the song from database #####
    # user_name=userentry.get()
    # conn=sqlite3.connect('users.db')
    # c=conn.cursor()
    # sql="SELECT * FROM "+user_name
    # c.execute(sql)
    # songlist=c.fetchall()
    # for item in songlist:
    #     browseFile.fileName=item[0]
    #     f=os.path.basename(browseFile.fileName)
    #     if f == selected_song:
    #         query = " DELETE FROM " +user_name+ " WHERE filepath =
'"+item[0]+"' "
    #         c.execute(query)
    #         conn.commit()
    #         print("commmit")

# --- Left Frame ---
leftframe = ttk.Frame(root)
leftframe.pack(side=LEFT, padx = 20, pady = 10)

addPhoto = PhotoImage(file="Images/add.png")
delPhoto = PhotoImage(file="Images/minus.png")
addBtn = Button(leftframe, image=addPhoto, command = browseFile)
delBtn = Button(leftframe, image=delPhoto, command = del_song)

label3=ttk.Label(leftframe,text="Current Songs")
label3.pack()

playlistbox = Listbox(leftframe)
playlistbox.pack()



addBtn.pack()
```

```
delBtn.pack(pady=10)

# --- Right Frame ---
rightframe = ttk.Frame(root)
rightframe.pack()
#--------------------

# --- Top Frame -----
topframe = ttk.Frame(root)
topframe.pack(pady = 20)
#---------------



# ----------------

# --- Submenu ---

menubar.add_cascade(label="File", menu=submenu)
submenu.add_command(label="Open", command=browseFile)
submenu.add_command(label="Load Playlist", command=loadfromdb)
submenu.add_command(label="Exit", command=root.destroy)
# ----------------
#--- Info about us ---

def aboutUs():
   tkinter.messagebox.showinfo('About Us', 'This Project is made by : \n1) Parth
(18CE2019) \n2) Sushil (18CE2011) \n3) Prithvi (18CE2007)\n4)
Samiksha(18C32030)')

# -------------------

submenu = Menu(menubar,tearoff=0)
menubar.add_cascade(label="Help", menu=submenu)
submenu.add_command(label="About Us", command=aboutUs)

# ---------------



# MAIN
mixer.init()
root.title("MP3 Player")
root.iconbitmap(r'icon.ico')
```

```
# -- Images -------
playPhoto = PhotoImage(file="Images/play.png")
pausePhoto = PhotoImage(file="Images/pause.png")
stopPhoto = PhotoImage(file="Images/Stop.png")
unmutePhoto = PhotoImage(file="Images/lmute.png")
mutePhoto = PhotoImage(file="Images/mute.png")

# --------------
usernamelabel = ttk.Label(topframe, text='Play List:')
usernamelabel.pack()

userentry = ttk.Entry(topframe)
userentry.pack()

usernamelabel1 = ttk.Label(topframe)
usernamelabel1.pack()

loadbtn=ttk.Button(topframe,command=loadfromdb,text="'Load'")
loadbtn.pack()

usernamelabel2 = ttk.Label(topframe)
usernamelabel2.pack()

newplaylist=ttk.Button(topframe,command=newplaylist,text="'New Playlist'")
newplaylist.pack()

usernamelabel2 = ttk.Label(topframe,text="")
usernamelabel2.pack()

lengthlabel = ttk.Label(topframe, text='Total Length : --:--')
lengthlabel.pack()

currentlabel = ttk.Label(topframe, text="Current Time : --:-- ")
currentlabel.pack()


# --- Middle Frame ---

middleFrame = ttk.Frame(root, relief=RAISED, borderwidth=1)
middleFrame.pack(padx = 30, pady = 20)

#----Playlist List Box ----#
```

```
label4=ttk.Label(leftframe,text="List of playlist")
label4.pack()
playlist_list_box1 = Listbox(leftframe)
playlist_list_box1.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:
        tkinter.messagebox.showerror('Error 000002x', 'Use a mp3 file only')

    mins, secs = divmod(total_length, 60)
    mins = round(mins)
    secs = round(secs)
    timeformat = '{:02d}:{:02d}'.format(mins, secs)
    lengthlabel['text'] = "Total Length" + ' - ' + 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)
            currentlabel['text'] = "Current Time" + ' - ' + timeformat
            time.sleep(1)
            current_time = current_time+1



# --- Play music ---
```

```
def playFunction():
    global paused
    if paused:
        mixer.music.unpause()
#       playBtn.configure(image=pausePhoto)
        paused = FALSE

    else:
        try:
            stopFunction()
            time.sleep(1)
            selected_song = playlistbox.curselection()
            selected_song = int(selected_song[0])
            play_it = playlist[selected_song]
            mixer.music.load(play_it)
            mixer.music.play()
            show_details(play_it)

            paused = FALSE


        except FileNotFoundError:
            tkinter.messagebox.showerror('Error 000001x ', 'The file could not be
found or is corrupted.')

        else:
            print("No error")


playBtn = Button(middleFrame, image=playPhoto, command=playFunction,
highlightthickness = 0, bd = 0)
playBtn.grid(row=2, column=0, padx = 10)
# --------------------
paused = FALSE
# --- Pause music ---
def pauseFunction():
    global paused
    paused = TRUE
    mixer.music.pause()

pauseBtn = Button(middleFrame, image = pausePhoto, command = lambda :
pauseFunction(), highlightthickness = 0, bd = 0)
pauseBtn.grid(row = 2,column=1, padx = 10)
```

```
# --------------------
# --- Stop music -----
def stopFunction():
    global stopped
    stopped = TRUE
    mixer.music.stop()
stopBtn = Button(middleFrame, image=stopPhoto,
command=lambda:stopFunction(), highlightthickness=0, bd=0)
stopBtn.grid(row = 2,column=2, padx = 10)

# --- Bottom Frame -----
bottomframe = ttk.Frame(root, relief=RAISED, borderwidth=1)
bottomframe.pack(padx = 30, pady=20)
# --- mute and unmute ----
muted = FALSE
def muteFunction():
    global muted
    if muted :
        unmuteBtn.configure(image=unmutePhoto)
        mixer.music.set_volume(0.5)
        muted = FALSE
        scale.set(50)
    else:
        unmuteBtn.configure(image=mutePhoto)
        mixer.music.set_volume(0)
        scale.set(0)
        muted = TRUE
unmuteBtn = Button(bottomframe, image=unmutePhoto,
command=muteFunction, highlightthickness=0, bd=0)
unmuteBtn.grid(row=0, column=1)
# --------------------------
# --- Volume Slider ---
def setVolume(val):
    vol = float(val)/100
    mixer.music.set_volume(vol)
    if vol > 0:
        unmuteBtn.configure(image=unmutePhoto)
        muted = FALSE
    if vol == 0:
        unmuteBtn.configure(image=mutePhoto)
        muted = TRUE
```

```
scale = ttk.Scale(bottomframe, from_ = 0, to = 100, orient=HORIZONTAL,
command =setVolume)
scale.set(50)
mixer.music.set_volume(0.5)
scale.grid(row = 0, column = 2,pady = 15)

listofplaylist()
def on_closing():
    stopFunction()
    root.destroy()
root.protocol("WM_DELETE_WINDOW", on_closing)
root.mainloop()
# def start():
#     root.protocol("WM_DELETE_WINDOW", on_closing)
#     root.mainloop()
```

# Chapter 5

# Output/ Results

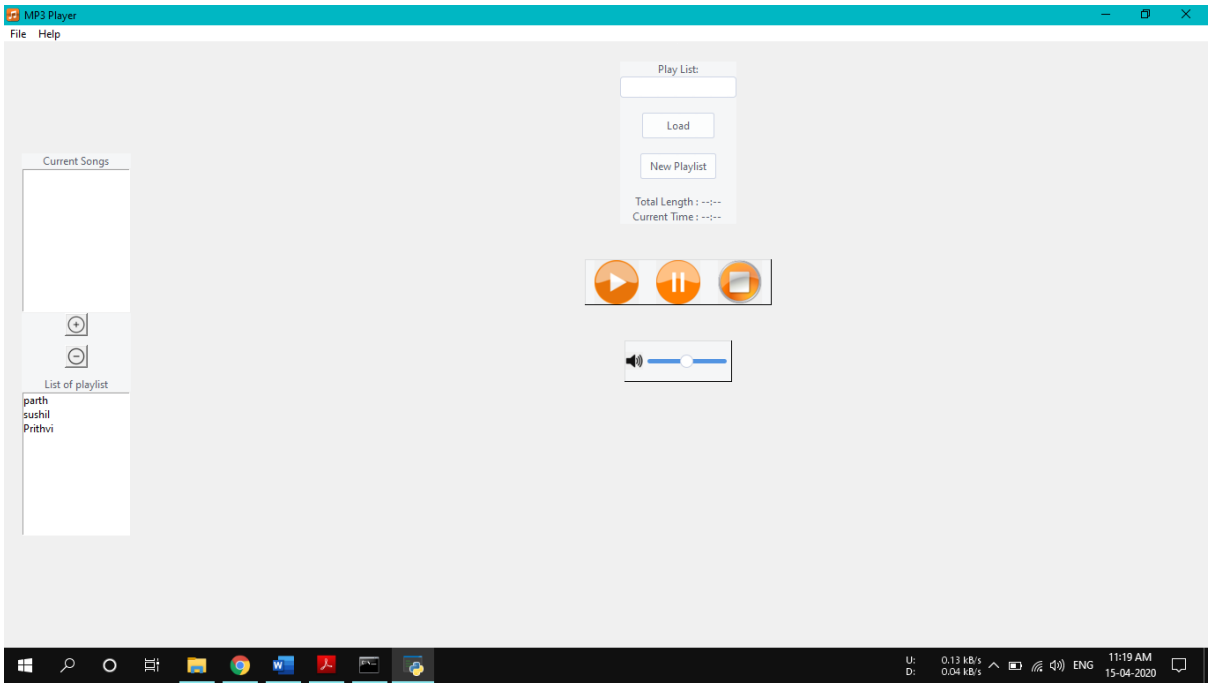



**Figure 1: - Home screen**

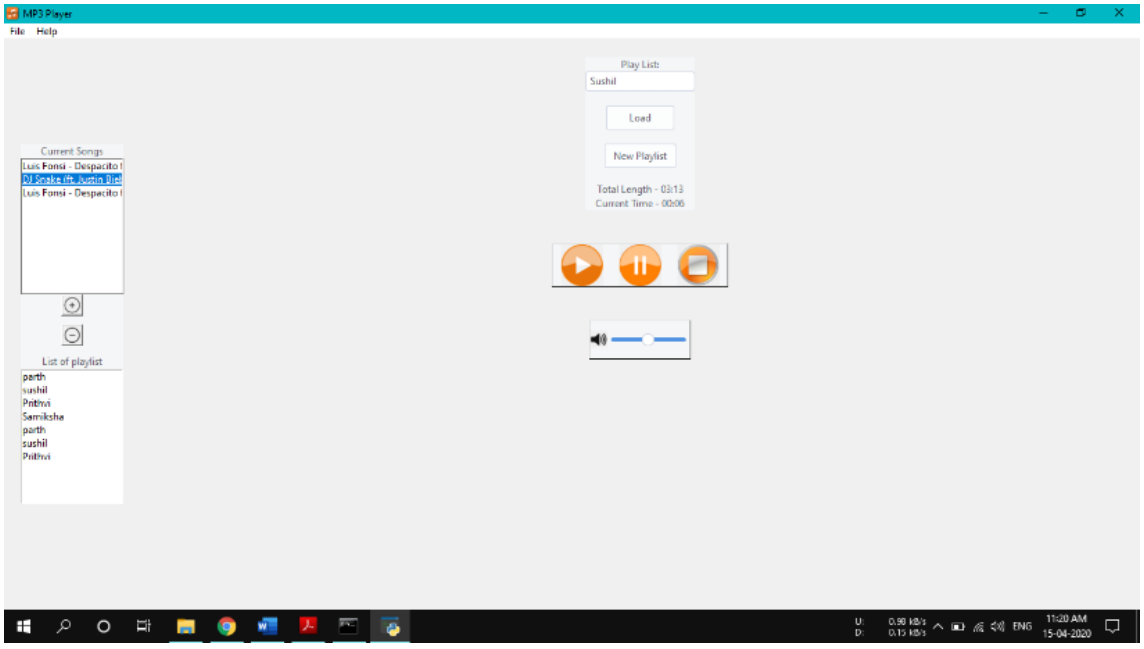



**Figure 2: - Pausing the current song**

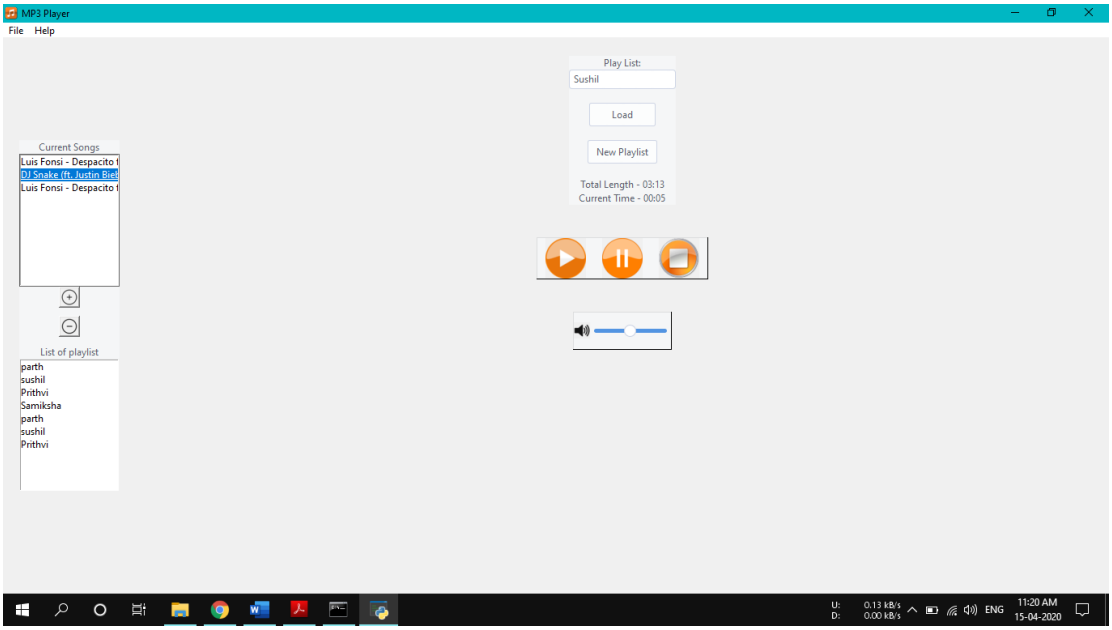



**Figure 3: - Selecting a song to play**

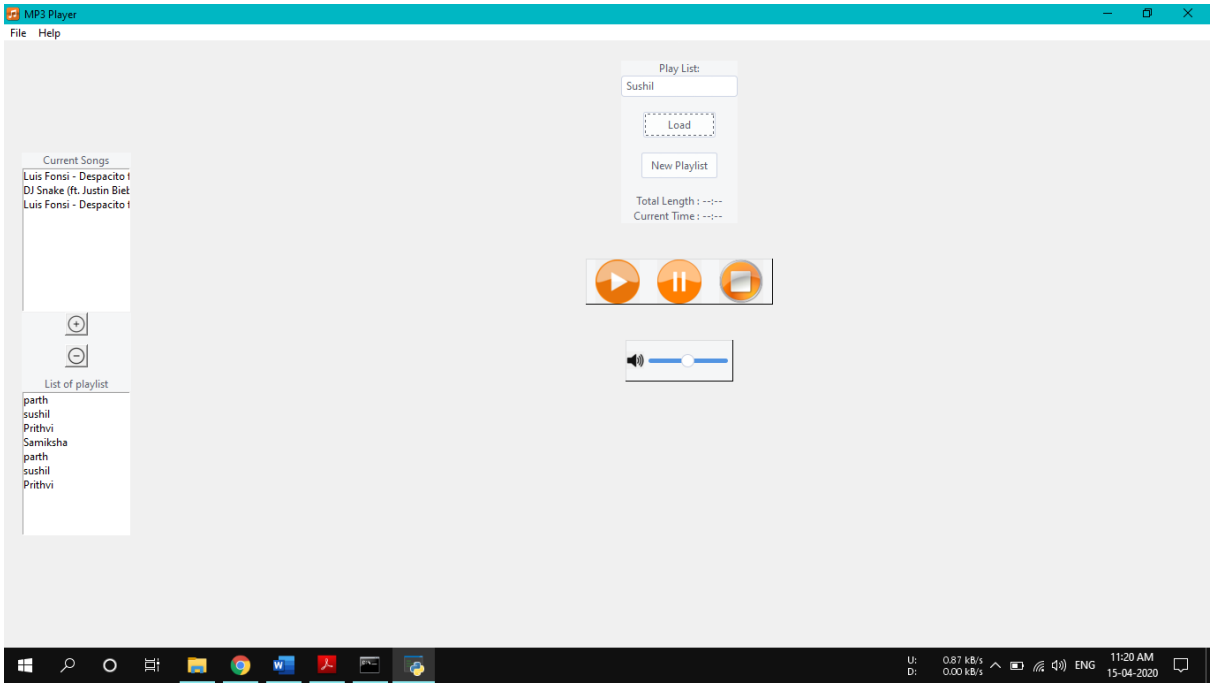



**Figure 4: - Adding song from existing playlist**

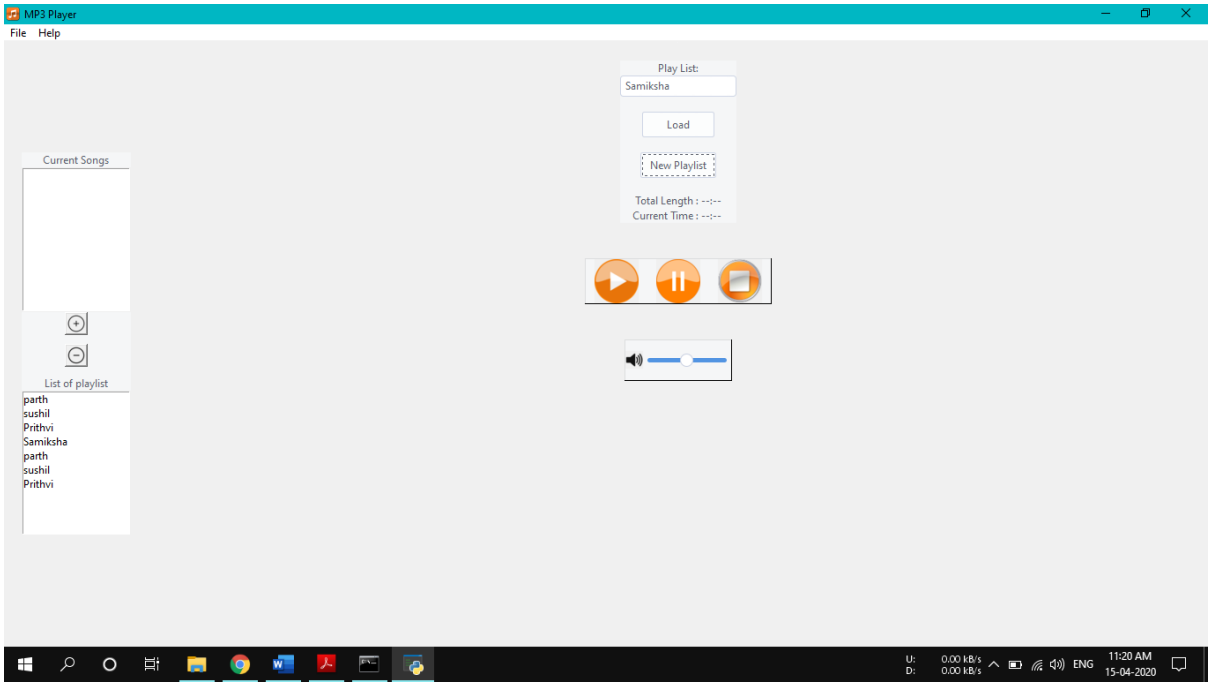



**Figure 5: - Adding a new playlist**

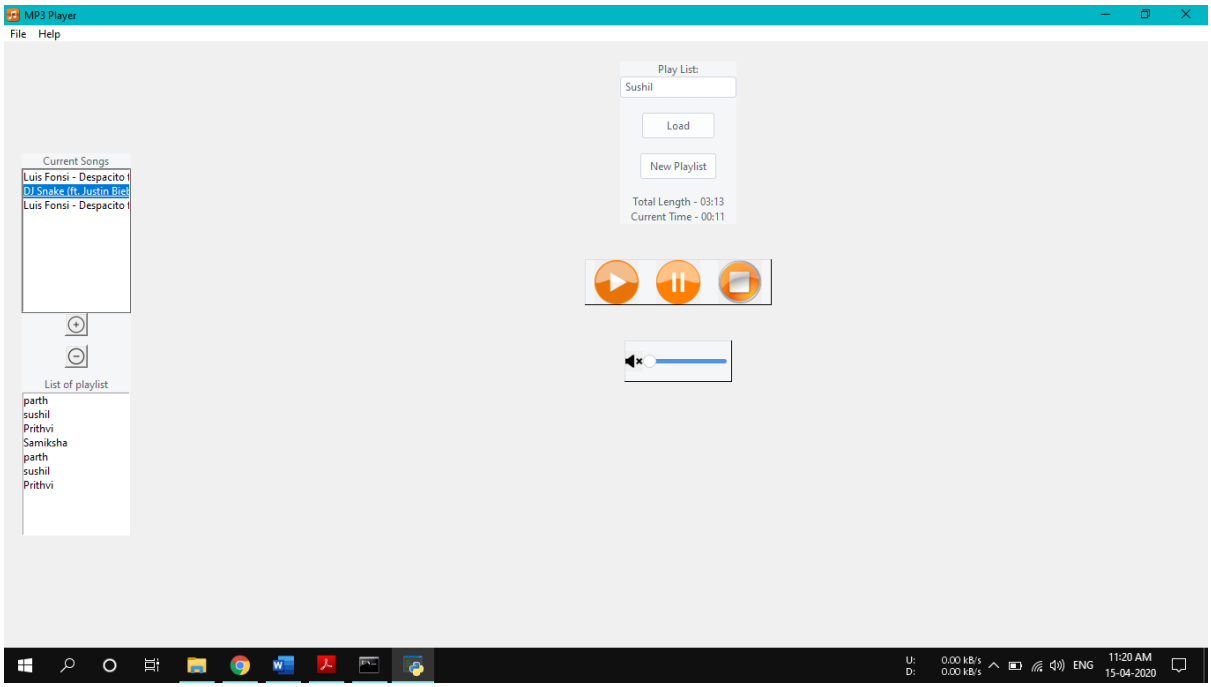



**Figure 6: - Mute Volume**