MINI-PROJECT
(2020-2021)

MUSIC PLAYER
(PYTHON)

PROJECT REPORT

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA
UNIVERSITY
MATHURA
Established vide U.P. Act 21 of 2010.

SUBMITTED TO:

Mrs. Priya Agarwal
(Technical Trainer)

SUBMITTED BY:

Shubham Singh
(181500698)

Department of Computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Declaration

I hereby declare that the work which is being presented in the Mini Project **"Music Player",** in partial fulfillment of the requirements for Mini Project Lab is an authentic record of my own work carried under the supervision of **Mrs. Priya Agarwal, Technical Trainer.**

Shubham Singh

181500698

Department of Computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Certificate

This is to certify that the project entitled "Music Player" carried out in Mini Project – I Lab is a bonafide work done by **Shubham Singh(181500698)** and is submitted in partial fulfillment of the requirements for the award of the degree **Bachelor of Technology (Computer Science & Engineering).**

Signature of Supervisor:

Name of Supervisor: Mrs. Priya Agarwal

Training Certificate

Acknowledgement

It gives me a great sense of pleasure to present the report of the B. Tech Mini Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that I have received.

I owe a special debt of gratitude to Mrs. Priya Agarwal, Technical Trainer, for her constant support and guidance throughout the course of the work. Her sincerity, thoroughness and perseverance have been a constant source of inspiration. She has showered with all her extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught about the latest industry-oriented technologies.

I would not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of the project. Last but not the least, I acknowledge my friends for their contribution in the completion of the project.

SHUBHAM SINGH

(181500698)

Abstract

Music is important part of our life. We know that the music 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 Music 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 Music player GUI project idea attempts to emulate the physical Music Player.

This program will allow you to play songs, music, and all MP3 files on your desktop or laptops. Music 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 OS.

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

Table of Contents

Introduction

1.1 Motivation and Overview

We need an application that will allow us to play or listen to music. Music player is the device to play Muisc and other digital audio files. The Muisc Player GUI program application attempts to emulate the physical 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, Os and Pickle libraries.

1.2 Objective

There are many electronic gadgets that have a very important role in our life.
One of the most important gadget nowadays, is the music player. Listening to music is a hobby of almost every person we meet around daily, for playing this music we need to have installed a music player in our device. It will be very exciting to spend our free time with our favourite music.

Since python provides usability to write cross-platform codes. Therefore I have designed a cross-platform music player that will help user play, pause and stop a track as per their needs.

Software Requirement Analysis

Problem Definition:

1. To build a Music player using Python programming language to be able to play and listen to songs, MP3 files and other digital audio files.
2. The player should be having 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.
3. The player should be able to play any song. It should be capable of playing MP3 files or any other digital audio files.
4. The player should allow the user to browse through the contents of the computer drive to choose song/s to be played or queued.
5. It should provide the user with option to pause or resume the song.
6. The user should be able to play the previous or the next song in the playlist.

Tools Used:

Python Programming:

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Features of Python:

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Python Tkinter GUI:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented 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 −

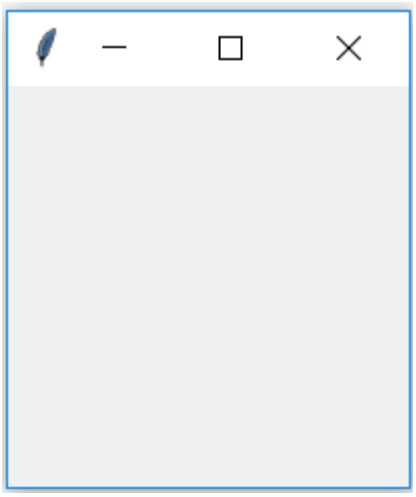
- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example

```
#!/usr/bin/python

import tkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

This would create the following window:



Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

The following are the widgets of Tkinter:

Sr.No.	Operator & Description
1	**Button** The Button widget is used to display buttons in your application.
2	**Canvas** The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	**Checkbutton** The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	**Entry** The Entry widget is used to display a single-line text field for accepting values from a user.
5	**Frame** The Frame widget is used as a container widget to organize other widgets.
6	**Label** The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

7	**Listbox** The Listbox widget is used to provide a list of options to a user.
8	**Menubutton** The Menubutton widget is used to display menus in your application.
9	**Menu** The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	**Message** The Message widget is used to display multiline text fields for accepting values from a user.
11	**Radiobutton** The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
12	**Scale** The Scale widget is used to provide a slider widget.
13	**Scrollbar** The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	**Text** The Text widget is used to display text in multiple lines.
15	**Toplevel** The Toplevel widget is used to provide a separate window container.
16	**Spinbox**

	The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	**PanedWindow** A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	**LabelFrame** A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	**tkMessageBox** This module is used to display message boxes in your applications.

Pygame module:

Pygame is a cross-platform set of Python modules which is used to create video games. It consists of computer graphics and sound libraries designed to be used with the Python programming language. Pygame is suitable to create client-side applications that can be potentially wrapped in a standalone executable

Built on top of the highly portable SDL (Simple DirectMedia Layer) development library, pygame can run across many platforms and operating systems. By using the pygame module, we can control the logic and graphics of your games without worrying about the backend complexities required for working with video and audio.

Installation:

The best way to install Pygame is with the pip tool (which is what python uses to install packages). The command is the following:

py -m pip install -U pygame --user

OS Module:

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 with the file system.

Following are some functions in OS module:

1. os.name: This function gives the name of the operating system dependent module imported. The following names have currently been registered: 'posix', 'nt', 'os2', 'ce', 'java' and 'riscos'.

2. os.getcwd(): Function os.getcwd(), returns the Current Working Directory(CWD) of the file used to execute the code, can vary from system to system.

3. os.error: All functions in this module raise OSError in the case of invalid or inaccessible file names and paths, or other arguments that have the correct type, but are not accepted by the operating system. os.error is an alias for built-in OSError exception.

4. os.popen(): This method opens a pipe to or from command. The return value can be read or written depending on whether mode is 'r' or 'w'.

5. os.close(): Close file descriptor fd. A file opened using open(), can be closed by close()only. But file opened through os.popen(), can be closed with close() or os.close(). If we try closing a file opened with open(), using os.close(), Python would throw TypeError.

6. os.rename(): A file old.txt can be renamed to new.txt, using the function os.rename(). The name of the file changes only if, the file exists and user has sufficient privilege permission to change the file.

Pickle Module:

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it "serializes" the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

Pycharm:

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

Features of Pycharm:

- Coding assistance and analysis, with code completion , syntax and error highlighting, linter integration, and quick fixes
- Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages
- Python refactoring: includes rename, extract method, introduce variable, introduce constant, pull up, push down and others
- Support for web frameworks: Django and Flask
- Integrated Python debugger
- Integrated unit testing, with line-by-line code coverage.
- Version control integration: unified user interface
- Support for scientific tools like matplotlib, numpy and scipy

Requirements:

Following are the hardware and the software requirements for our project:

a) Hardware:

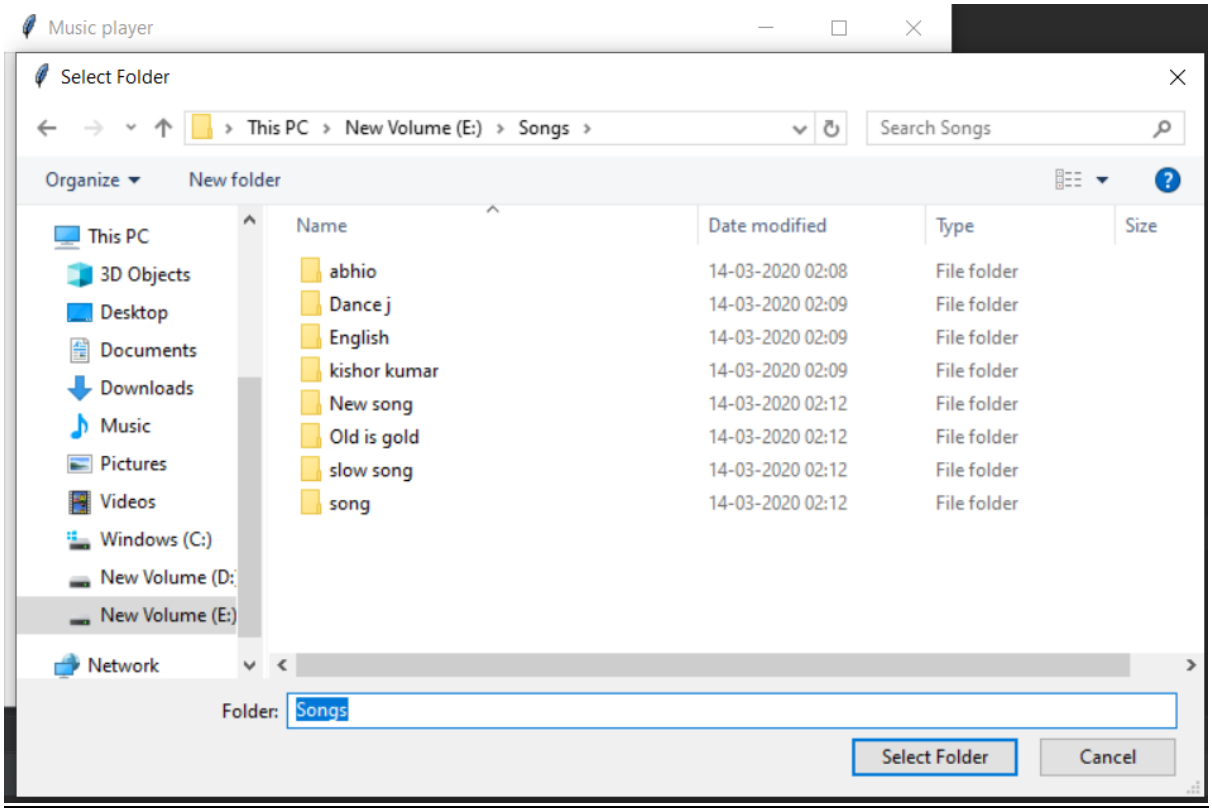
- 120 GB HDD
- Minimum 4 GB RAM
- Laptop/Desktop
- i5 processor

b) Software:

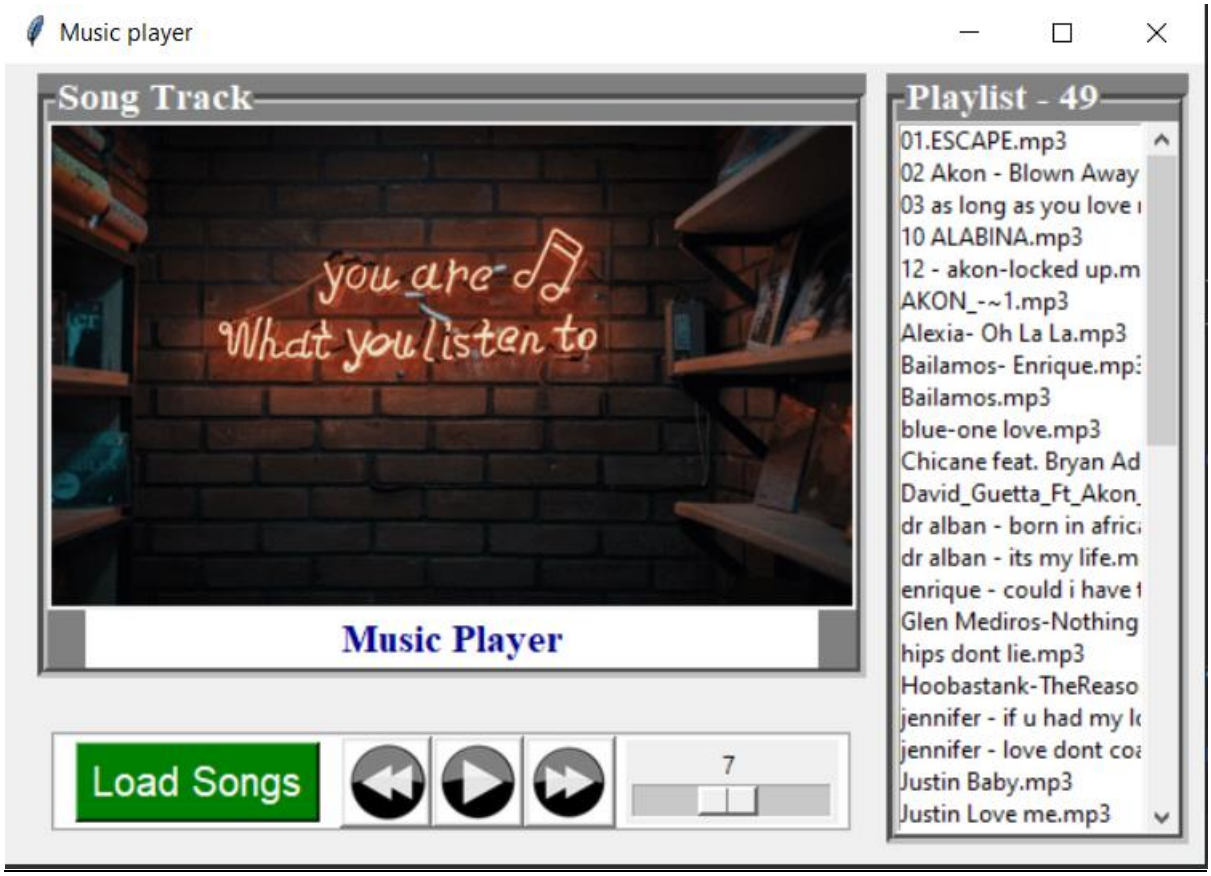
- Pycharm
- Operating System(Windows, Linux)
- **Programming Language**
  - ❖ Python 3.8
    - Tkinter (Python GUI library)
  - Pygame (Python Library)
    - OS module
    - Pickle module

Software Design

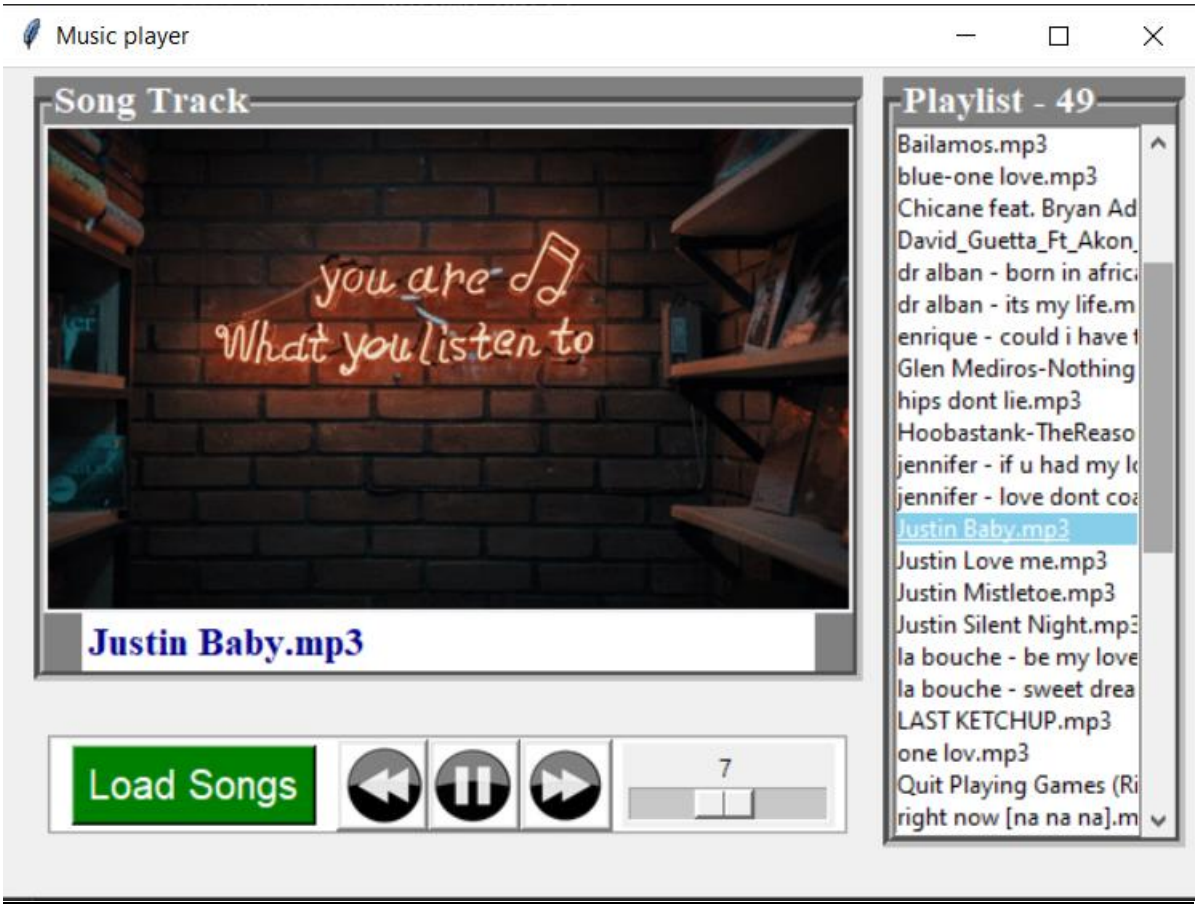
Opening Screen of the player



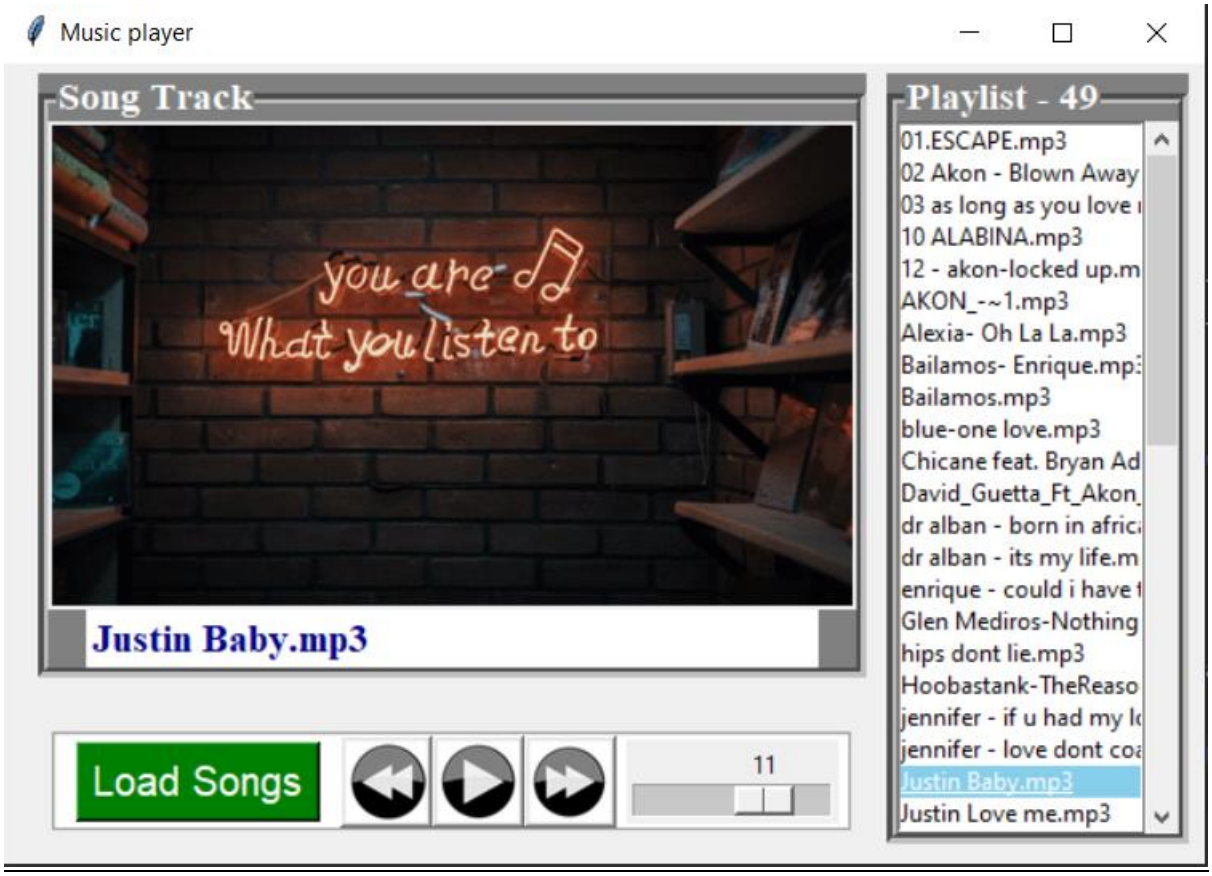
When clicking on Load Songs button



When the folder is selected



When song is played


Music player
Song Track
you are
What you listen to
Justin Baby.mp3
Load Songs
11
Playlist - 49
01.ESCAPE.mp3
02 Akon - Blown Away
03 as long as you love
10 ALABINA.mp3
12 - akon-locked up.m
AKON_-~1.mp3
Alexia- Oh La La.mp3
Bailamos- Enrique.mp
Bailamos.mp3
blue-one love.mp3
Chicane feat. Bryan Ad
David_Guetta_Ft_Akon_
dr alban - born in afric
dr alban - its my life.m
enrique - could i have
Glen Mediros-Nothing
hips dont lie.mp3
Hoobastank-TheReaso
jennifer - if u had my l
jennifer - love dont co
Justin Baby.mp3
Justin Love me.mp3

Implementation:

Code:

```
import os
import pickle
import tkinter as tk
from pygame import mixer
from tkinter import filedialog
from tkinter import PhotoImage

class Player(tk.Frame):      #player class
   def __init__(self,master):
      super().__init__(master)
      self.master=master
      self.pack()

      mixer.init()

      if os.path.exists('songs.pickle'):
         with open('songs.pickle','rb') as f:
            self.playlist=pickle.load(f)
      else:
         self.playlist=[]

      self.current=0
      self.paused= True
      self.played= False

      self.create_frames()
      self.track_widgets()
      self.control_widgets()
      self.tracklist_widget()
```

```
 def create_frames(self):
        self.track=tk.LabelFrame(self,text='Song Track',
                    font=("Times New Roman",14,"bold"),
                        bg="grey",fg="white",bd=5,relief=tk.GROOVE)
        self.track.configure(width=410,height=300)
        self.track.grid(row=0,column=0,padx=10)

        self.tracklist = tk.LabelFrame(self, text=f'Playlist - {str(len(self.playlist))}',
                            font=("Times New Roman", 14, "bold"),
                            bg="grey", fg="white", bd=5, relief=tk.GROOVE)
        self.tracklist.configure(width=190, height=400)
        self.tracklist.grid(row=0, column=1, rowspan=3,pady=5)

        self.controls = tk.LabelFrame(self,
                            font=("Times New Roman", 15, "bold"),
                            bg="white", fg="white", bd=2, relief=tk.GROOVE)
        self.controls.configure(width=410, height=80)
        self.controls.grid(row=2, column=0,pady=5,padx=10)

    def track_widgets(self):

        self.canvas= tk.Label(self.track, image=img)
        self.canvas.configure(width=400, height=240)
        self.canvas.grid(row=0, column=0)

        self.songtrack= tk.Label(self.track, font=("Times New Roman", 15,
"bold"),
                            bg='white',fg='dark blue')
        self.songtrack['text']='Music Player'
        self.songtrack.configure(width=30, height=1)
        self.songtrack.grid(row=1, column=0)


    def control_widgets(self):
        self.loadSongs=tk.Button(self.controls, bg='green',fg='white',font=10)
        self.loadSongs['text']= 'Load Songs'
        self.loadSongs['command']= self.retrieve_songs
        self.loadSongs.grid(row=0,column=0, padx=10)
```

```
  self.prev = tk.Button(self.controls,image=prev )
        self.prev['command']= self.prev_song
        self.prev.grid(row=0, column=1)

        self.pause = tk.Button(self.controls, image=pause )
        self.pause['command']=self.pause_song
        self.pause.grid(row=0, column=2)

        self.next = tk.Button(self.controls, image=next_ )
        self.next['command']=self.next_song
        self.next.grid(row=0, column=3)

        self.volume= tk.DoubleVar()
        self.slider=tk.Scale(self.controls, from_=0, to=15,
orient=tk.HORIZONTAL)
        self.slider['variable']= self.volume
        self.slider.set(7)
        mixer.music.set_volume(0.7)
        self.slider['command']=self.change_volume
        self.slider.grid(row=0,column=4, padx=5)


    def tracklist_widget(self):
        self.scrollbar=tk.Scrollbar(self.tracklist,orient=tk.VERTICAL)
        self.scrollbar.grid(row=0,column=1, rowspan=5, sticky='ns')

        self.list= tk.Listbox(self.tracklist, selectmode=tk.SINGLE,
                    yscrollcommand= self.scrollbar.set, selectbackground= 'sky blue')
        self.enumerate_songs()
        self.list.config(height=22)
        self.list.bind('<Double-1>', self.play_song)

        self.scrollbar.config(command=self.list.yview)
        self.list.grid(row=0,column=0,rowspan=5)

    def enumerate_songs(self):
        for index, song in enumerate(self.playlist):
            self.list.insert(index, os.path.basename(song))
```

```
def retrieve_songs(self):
        self.songlist=[]
        directory= filedialog.askdirectory()
        for root_,dirs,files in os.walk(directory):
            for file in files:
                if os.path.splitext(file)[1]== '.mp3':
                    path=(root_ + '/'+ file).replace('\\','/')
                    self.songlist.append(path)
        with open('songs.pickle', 'wb') as f:
            pickle.dump(self.songlist,f)

        self.playlist=self.songlist
        self.tracklist['text']=f'Playlist - {str(len(self.playlist))}'
        self.list.delete(0, tk.END)
        self.enumerate_songs()

    def play_song(self, event= None):
        if event is not None:
            self.current=self.list.curselection()[0]
            for i in range(len(self.playlist)):
                self.list.itemconfigure(i, bg='white')

        mixer.music.load(self.playlist[self.current])
        self.pause['image']= play
        self.paused= False
        self.played= True
        self.songtrack['anchor']='w'
        self.songtrack['text']=os.path.basename(self.playlist[self.current])
        self.list.activate(self.current)
        self.list.itemconfigure(self.current, bg='sky blue')
        mixer.music.play()
```

```
def pause_song(self):
    if not self.paused:
        self.paused= True
        mixer.music.pause()
        self.pause['image']= pause
    else:
        if self.played == False:
            self.play_song()
        self.paused= False
        mixer.music.unpause()
        self.pause['image']= play


def prev_song(self):
    if self.current >0:
        self.current -= 1
    else:
        self.current=0
    self.list.itemconfigure(self.current+1, bg='white')
    self.play_song()

def next_song(self):
    if self.current <len(self.playlist) - 1:
        self.current += 1
    else:
        self.current=0
        self.play_song()
    self.list.itemconfigure(self.current-1, bg='white')
    self.play_song()


def change_volume(self,event=None):
    self.v= self.volume.get()
    mixer.music.set_volume(self.v/10)
```

```
root=tk.Tk()
root.geometry('600x400')
root.wm_title('Music player')

img= PhotoImage(file='images/music.gif')
next_= PhotoImage(file= 'images/next.gif')
prev= PhotoImage(file='images/previous.gif')
play= PhotoImage(file='images/play.gif')
pause= PhotoImage(file='images/pause.gif')

app=Player(master=root)
app.mainloop()
```

Conclusion

This project has really been faithful and informative. It has made me learn and understand the many trivial concepts of Python Language. As I have used python Tkinter as a GUI it provides various controls, such as buttons, labels and text boxes to build a user friendly application. The fast growing use of internet confirms the good future and scope of the proposed project. Also it provides all necessary features a user needs from a Music Player.

Bibliography

- www.tkdocs.com
- www.youtube.com
- www.udemy.com
- www.python.org

- **Faculty Guidance:**
    - Mrs. Priya Agarwal
    - Mr. Sharad Gupta