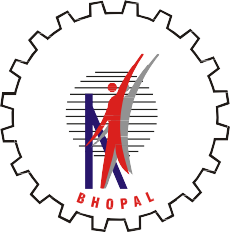
# RHYTHM MP3 PLAYER



**A Major Project Report**

**Submitted in Partial Fulfillment of requirements for the Award of Degree of Bachelor of Engineering in Computer Science & Engineering Submitted to**

# RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA,

**BHOPAL (M.P)**

**Submitted By:** PRIYANKA THULKAR En.No:-0158CS213D13

DEEPIKA NAGPURE

EN.NO. -0158CS213D07

HIMANI CHAURASIYA

EN NO. 0158CS213D08

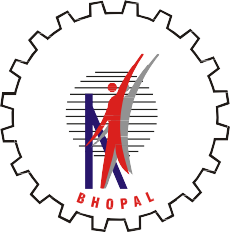
**Under the guidance**

**of Prof.Aakriti Shrivastava**

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING RADHARAMAN ENGINEERING COLLEGE, BHOPAL (M.P.)

**Session: 2023-24**

## RADHARAMAN ENGINEERING COLLEGE BHOPAL (M.P.)



**Department of Computer Science & Engineering**

## APPROVAL CERTIFICATE

This Major project work entitled **“RHYTHM MP3 PLAYER”** being submitted by **“Priyanka Thulkar” (Roll No.: 0158CS213D13) 2.** “**DEEPIKA NAGPURE” EN.NO.0158CS213D07 3. “HIMANI CHAURASIYA “** is/are approved for the award of degree of **Bachelor of Engineering** in **Computer Science and Engineering**.

**Internal Examiner External Examiner**

**Date: Date:**

## RADHARAMAN ENGINEERING COLLEGE BHOPAL (M.P.)

**Department of Computer Science & Engineering**

## CANDIDATE DECLARATION

We 01. **“PRIYANKA THULKAR ” (Roll No.: 0158CS213D13) 2.** “**DEEPIKA NAGPURE”**

**EN.NO.0158CS213D07 3. “HIMANI CHAURASIYA “ EN NO 0158CS213D08** students of **Bachelor of Engineering** in **Computer Science & Engineering, Radharaman Engineering College, Bhopal (M.P.), hereby** declare that the work presented in this Major project **RHYTHM MP3 PLAYER** is the outcome of our own work, is bonafide and correct to the best of our knowledge and this work has been carried out taking care of Engineering Ethics. Thework presented does not infringe any patented work and has not been submitted to any otheruniversity or anywhere else for the award of any degree or any professional diploma.

**Date:**

**PRIYANKA THULKAR EN. NO. - 0158CS213D13**

**DEEPIKA NAGPURE**

**EN.NO. -0158CS213D07**

**HIMANI CHAURASIYA**

**EN NO. 0158CS213D08**

# ACKNOWLEDGEMENT

This project work is the result of guidance and support of various people at REC without whom all our effort would have been directionless and fruitless. We sincerely thank all of them, for assisting us in completing the dissertation.

We express our ardent and earnest gratitude to our guide, **Prof. AAKRITI SHRIVASTAVA**, Department of Computer Science & Engineering, REC Bhopal and **Prof. RAKESH SHIVHARE**, HOD, Department of Computer Science & Engineering, REC Bhopal for their help and encouragement at all the stages of our Work. Their guidanceand motivation helped us to be fruitful in our effort.

We also express my heartfelt and profound gratitude to our Director **Dr. ANURAGJAIN** for his valuable suggestion and ample resources at all stages of the research work. Finally, we would like to say that we are indebted to my parents for everything that they have done for us. All of this would have been impossible without their constant support. And we also thanks to God for being kind to me and driving me through this journey.

**PRIYANKA THULKAR EN. NO. - 0158CS213D13**

**DEEPIKA NAGPURE**

**EN.NO. -0158CS213D07**

**HIMANI CHAURASIYA**

**EN NO. 0158CS213D08**

## INDEX

Chapter 1 **Introduction**

* Overview
* Objective
* Applications

Chapter 2 **Requirement Analysis**

* Gantt Charts

Chapter 3 **Designing & Methodology**

* Project module
* Diagrams (Flow Charts: Use Case, Class Diagram, Sequence Diagram, DFD with explanation
* Database Designing (if Any)
* E-R Diagrams

Chapter 4 **Implementation and Results**

* External Interface Requirements
  + User Interfaces
  + Hardware Interfaces
  + Software Interfaces
* Other Nonfunctional Requirements
  + Performance Requirements
  + Safety Requirements
  + Security Requirements
  + Software Quality Attributes
* Screen Shots with Description Chapter 5 **Testing**
* Testing

Chapter 6 **Conclusion**

* Conclusion

## LIST OF FIGURE

|  |  |  |
| --- | --- | --- |
| 1 | Figure 1.1 | Sequence diagram |
| 2 | Figure 1.2 | Class diagram |
| 3 | Figure 1.3 | Flow chart |
| 4 | Figure 1.4 | E-R diagram |

### Chapter 1 Introduction

Overview

The “music player” is a software project supporting all known media files and has the ability to play them with ease. User may attach Folder to Play add various media files within it. User may see track lists and play desired ones accordingly.

* 1. **Objectives**

The back-end (database) contains 1 table:

My favourites But we have to create this tables in a new user in Oracle database. So first we will create a new user in Oracle called mouzikka with the password music and then in this user we have to create the above mentioned table.

HOW TO CREATE A NEW USER IN ORACLE

* + 1. Login to System user with the password you have set. We are assuming the password is abc: connect system/abc;
    2. Now type the following command

Create user mouzikka identified by music;

* + 1. Now type the next command as shown below to give the user privileges Grant resource, connect to mouzikka;

Now login to the user mouzikka and create the table Myfavourites with the structure shown below:

* 1. **Application**

In our Music Player App we have divided the code into 3 modules or **.**py files called Model.py, View.py and Player.py

This project used with windows operating system also useful in linux platform, because it’s a open source platfor

### Chapter 2 Requirements Analysis

Example of Gantt chart

**Weeks 1 2 3 4 5 6 7 8**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

**Planning**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

**Design**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

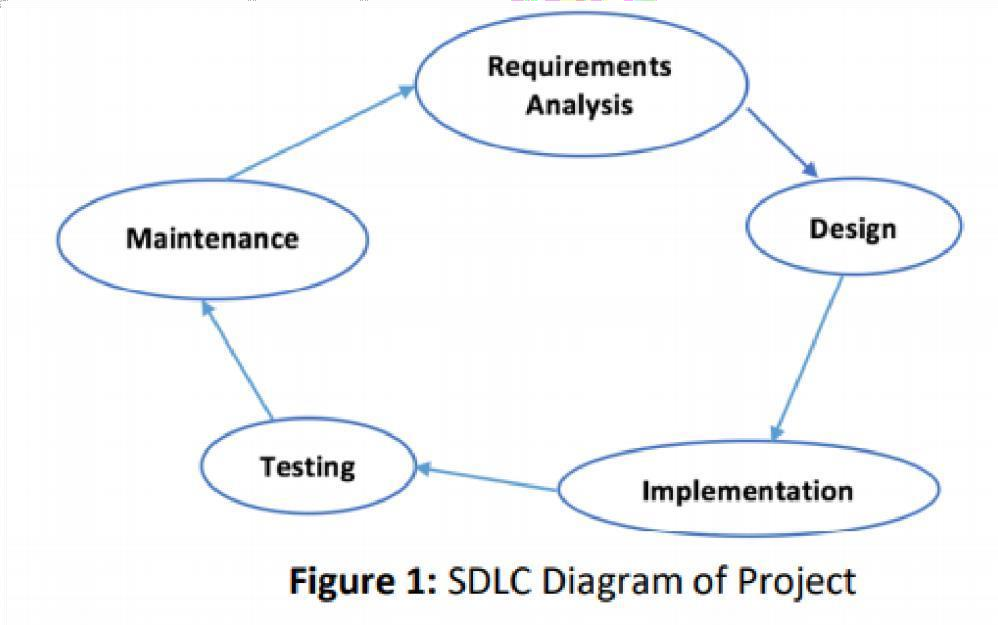
**Coding**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

**Testing**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

**Delivery**



In this project music player app is required the three modules (view.py, model.py, player.py)

And also required one database (any database like oracle,sql lite etc) on this project the database used is oracle

Many pythons libraries are required like tkinter (to used gui created), Pygame(for playing song), mutagen.mp3 (for getting song details), traceback (module) for error handling, threading module(for multithreading), Database: Oracle (for storing favourites) module used (cx Oracle) .

MVC design pattern has been implemented in the project

**Chapter : 3 Design and Methodolog**

# Project Module

In our Music Player App we have divided the code into 3 modules or **.**py files called Model.py, View.py and Player.py

1. View.py **:** All the front end code is written here. It contains the root window and UI elements our application will require and the execution will begin from here.

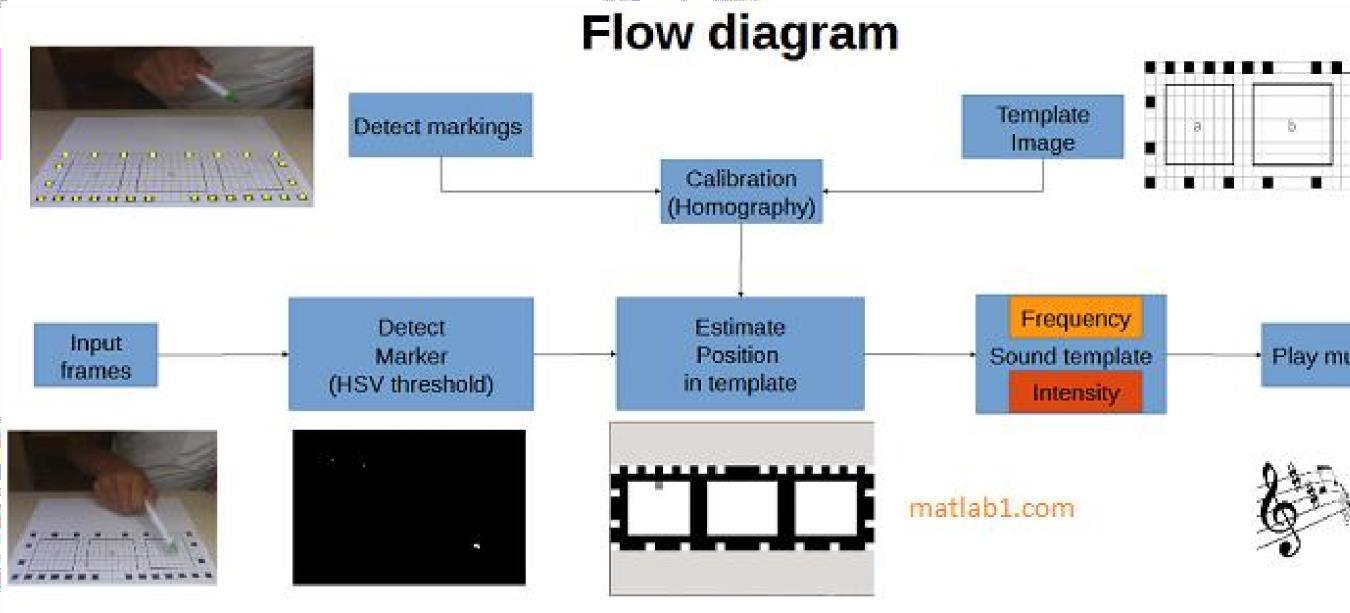
All the calls to other methods will be made from here.

1. Model.py: All the code that interacts with the database is written here. It contains methods like add\_song(),remove\_song(),add\_song\_to\_favourite(),load\_

songs\_from\_f avourite(), remove songs from\_favourite() etc

1. Player.py: This file will manage the actual playing / pausing / rewind of songs. Also as mentioned, the View and Model do not directly communicate with each other. Sotheir communication is done via Controller which is the file Player.py. It contains methods like play\_song(), stop\_song(), pause\_song() as well as bridge methods like add\_song\_to\_favourite(),load\_songs\_from\_favourite() etc to pass on the user's request

to the Model



#### **Diagram(**Flow chart :- use case, class diagram, sequence diagram, Dfd explanation)

* + 1. **Database desiging**

#### HOW TO CREATE A NEW USER IN ORACLE

1. Login to System user with the password you have set. We are assuming the password is abc:

connect system/abc;

1. Now type the following command

Create user mouzikka identified by music;

1. Now type the next command as shown below to give the user privileges Grant resource , connect to mouzikka;

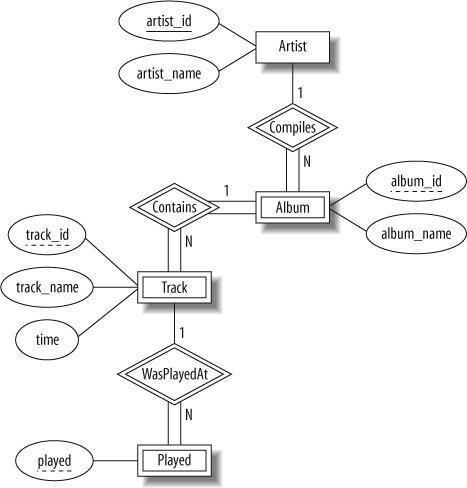
Now login to the user mouzikka and create the table Myfavourites with the structure shown

below:

My favouraites:

|  |  |
| --- | --- |
| **Column Name** | **Data Type** |
| **SONG\_ID** | **Number(4)** |
| **SONG\_NAME** | **Varchar2(100)** |
| **SONG\_PATH** | **Varchar2(200)** |

1. **E-R Diagram**



### Chapter:- 4 Implementation and Results

* + External Interface Requirements oUser Interfaces
    1. Back - end: python
    2. ii. Backend software: SQL+

#### Software Interfaces

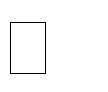
best free open-source music player software for Windows and also work this this project on linux platform(ubuntu etc.)

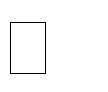
This music player provides all the basic but, essential tools to play audio files including Play/Pause Buttons, Next and Previous songs buttons to switch songs, Loop (to play one song or a list of songs continuously), and **Random** (to play songs in a random manner).

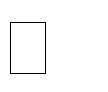
* + Other Nonfunctional Requirements
  + Performance Requirements The steps involved to perform the implementation of oracle database are as listed below.

1. **E-R diagram**

The E-R Diagram constitutes a technique for representing the logical structure of a database in a pictorial manner. This analysis is then used to organize data as a relation, normalizing relation and finally obtaining a relation database.

 **ENTITIES:** Which specify distinct real-world items in an application.

 **PROPERTIES/ATTRIBUTES:** Which specify properties of an entity and relationships.

 **RELATIONSHIPS:** Which connect entities and represent meaningful dependencies between them.

### Normalization

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored.

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulated the first, second and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and complete understanding of its implications.

* + Safety Requirements

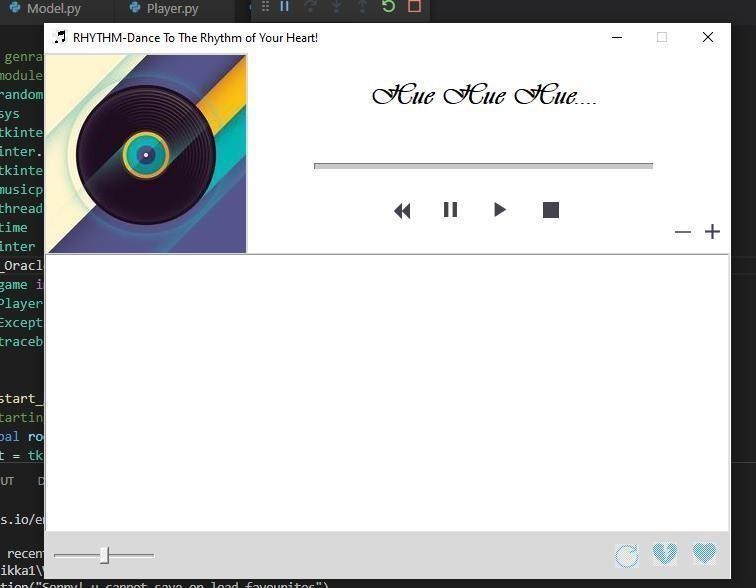
If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

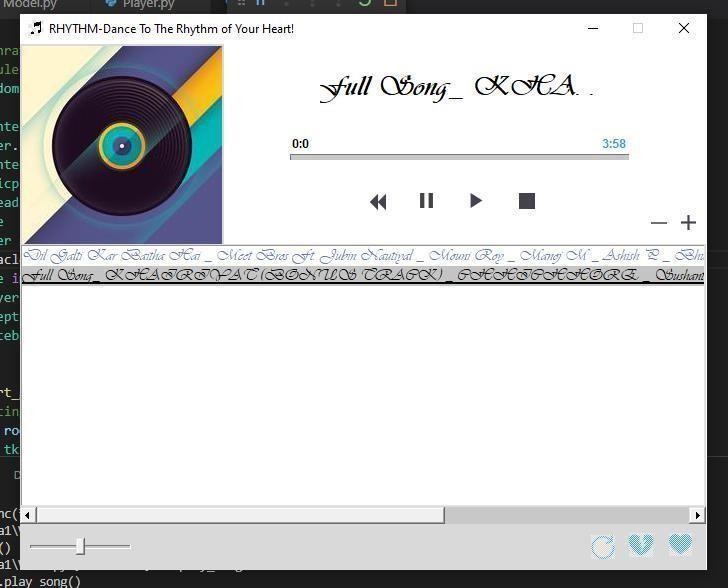
* + Security Requirements

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

* Software Quality Attributes

1. **AVAILABILITY**
2. **CORRECTNESS**
3. **MAINTAINABLITY**
4. **USABILITY**





**Final code is divided in the parts**

* Model.py
* View.py
* Player.py
* Model.py

All the code that interacts with the database is written here. It contains methods like add\_song(),remove\_song(),add\_song\_to\_favourite(),

lo songs\_from\_f avourite(), remove\_song\_from\_favourite() etc

# from sqlite3 import DatabaseError, connect from cx\_Oracle import DatabaseError,connect from traceback import\*

from cx\_Oracle import \* class Model:

def init (self): self.song\_dict={} self.db\_status=True self.conn=None self.cur=None

try:

self.conn=connect("mouzikka/music@127.0.0.1/xe") print("connected to successfully to the DB") self.cur=self.conn.cursor()

except DatabaseError: self.db\_status=False

print("DB error:",format\_exc())

def get\_db\_status(self): return self.db\_status

def close\_db\_connection(self): if self.cur is not None:

self.cur.close()

print("cursor closed succesfully") if self.conn is not None:

self.conn.close()

print("disconnected successfully from the DB")

def add\_song(self,song\_name,song\_path): self.song\_dict[song\_name]=song\_path print("song added:",self.song\_dict[song\_name])

def get\_song\_path(self,song\_name): return self.song\_dict[song\_name]

def remove\_song(self,song\_name):

self.song\_dict.pop(song\_name) print("after deletion:",self.song\_dict)

def search\_song\_in\_favourites(self,song\_name): self.cur.excute("select song\_name from myfavourite where

song\_name=:1",(song\_name))

song\_tuple=self.cur.fetchone() if song\_tuple is None:

return False else:

return True

def add\_song\_to\_favourites(self,song\_name,song\_path): is\_song\_present=self.search\_song\_in\_favorites(song\_name) if is\_song\_present==True:

return "song already present in favourites" self.cur.excute("select max(song\_id) from myfavourites") last\_song\_id=self.cur.fetchone()[0]

next\_song\_id=1

if last\_song\_id is not None: next\_song\_id=last\_song\_id+1

self.cur.execute("insert into myfavourites values(:1,:2,:3)",(next\_song\_id,song\_name,song\_path)) self.conn.commit()

return "song sucessfully added to your favourites"

def load\_song\_from\_favourites(self):

self.cur.execute("select song\_name,song\_path from myfavourites") for song\_name,song\_path in self.cur:

self.song\_dict[song\_name]=song\_path song\_present=True

if song\_present:

return "list populated from favourites" else: return "no songs present in your favourite"

def remove\_song\_from\_favourites(self,song\_name): self.cur.execute("delete from myfavourites where

song\_name=:1",(song\_name,))

count=self.cur.rowcount if count==0:

return "song not present in your favourites" else: self.song\_dict.pop(song\_name) self.conn.commit() return "song deleted from your favourites"

def get\_song\_count(self):

return len(self.song\_dict)

#### View.py

All the front end code is written here. It contains the root window and UI elements our application will require and the execution will begin from here. All the calls to other methods will be made from here

import random

import sys

system information import tkinter as tk

# for randomly changing color of song name

# sys module provides exit method and other

from tkinter.constants import COMMAND, FALSE

import tkinter.ttk as ttk import musicplayer\_support

import threading # it is used for multithreading

import time # it will handle time

from tkinter import filedialog,messagebox # for providing message to the user

from cx\_Oracle import DatabaseError from pygame import mixer

import Player

from MyException import\* # for handling our exceptions

import traceback

def vp\_start\_gui():

# Starting point when module is the main routine. global root

root = tk.Tk() top=View(root)

musicplayer\_support.init(root, top) root.resizable(False, False) root.mainloop()

class View:

def init (self, top=None):

#This class configures and populates the toplevel window. # top is the toplevel containing window.

\_bgcolor = '#d9d9d9' # X11 color: 'gray85'

\_fgcolor = '#000000' # X11 color: 'black'

\_compcolor = '#d9d9d9' # X11 color: 'gray85'

\_ana1color = '#d9d9d9' # X11 color: 'gray85'

\_ana2color = '#ececec' # Closest X11 color: 'gray92'

font11 = "-family {Avenir Next Cyr Medium} -size 23 -weight " \ "normal -slant roman -underline 0 -overstrike 0"

font12 = "-family {Avenir Next Cyr} -size 9 -weight bold " \

"-slant roman -underline 0 -overstrike 0"

font13 = "-family {Vivaldi} -size 22 -weight " \

"bold -slant roman -underline 0 -overstrike 0" self.style = ttk.Style() if sys.platform == "win32":

self.style.theme\_use('winnative')

self.style.configure('.',background=\_bgcolor) self.style.configure('.',foreground=\_fgcolor) self.style.configure('.',font="TkDefaultFont") self.style.map('.',background=

[('selected', \_compcolor), ('active',\_ana2color)])

top.geometry("687x526+558+155") top.title("New Toplevel") top.configure(background="#fff") self.top=top self.songName = tk.Label(top)

self.songName.place(relx=0.437, rely=0.038, height=44, width=281) self.songName.configure(background="#fff") self.songName.configure(disabledforeground="#a3a3a3") self.songName.configure(font=font13) self.songName.configure(foreground="#000000") self.songName.configure(text='''Hue Hue Hue ''')

self.songProgress = ttk.Progressbar(top) self.songProgress.place(relx=0.393, rely=0.209, relwidth=0.495

, relheight=0.0, height=7)

width=29)

self.songTotalDuration = ttk.Label(top) self.songTotalDuration.place(relx=0.844, rely=0.171, height=19,

self.songTotalDuration.configure(background="#fff") self.songTotalDuration.configure(foreground="#3399ff") self.songTotalDuration.configure(font=font12) self.songTotalDuration.configure(relief='flat')

self.songTimePassed = ttk.Label(top) self.songTimePassed.place(relx=0.393, rely=0.171, height=19, width=29) self.songTimePassed.configure(background="#ffffff") self.songTimePassed.configure(foreground="#000000") self.songTimePassed.configure(font=font12) self.songTimePassed.configure(relief='flat')

self.pauseButton = tk.Button(top) self.pauseButton.place(relx=0.568, rely=0.266, height=34, width=34) self.pauseButton.configure(activebackground="#ececec")

self.pauseButton.configure(activeforeground="#000000") self.pauseButton.configure(background="#fff") self.pauseButton.configure(borderwidth="0") self.pauseButton.configure(disabledforeground="#a3a3a3") self.pauseButton.configure(foreground="#000000") self.pauseButton.configure(highlightbackground="#d9d9d9") self.pauseButton.configure(highlightcolor="black") self.\_img1 = tk.PhotoImage(file="./icons/pause.png") self.pauseButton.configure(image=self.\_img1) self.pauseButton.configure(pady="0") self.pauseButton.configure(text='''Button''')

self.playButton = tk.Button(top) self.playButton.place(relx=0.64, rely=0.266, height=34, width=34) self.playButton.configure(activebackground="#ececec") self.playButton.configure(activeforeground="#000000") self.playButton.configure(background="#fff") self.playButton.configure(borderwidth="0") self.playButton.configure(disabledforeground="#a3a3a3") self.playButton.configure(foreground="#000000") self.playButton.configure(highlightbackground="#d9d9d9") self.playButton.configure(highlightcolor="black")

self.\_img2 = tk.PhotoImage(file="./icons/play.png") self.playButton.configure(image=self.\_img2) self.playButton.configure(pady="0") self.playButton.configure(text='''Button''')

self.stopButton = tk.Button(top) self.stopButton.place(relx=0.713, rely=0.266, height=34, width=34) self.stopButton.configure(activebackground="#ececec") self.stopButton.configure(activeforeground="#000000") self.stopButton.configure(background="#fff") self.stopButton.configure(borderwidth="0") self.stopButton.configure(disabledforeground="#a3a3a3") self.stopButton.configure(foreground="#000000") self.stopButton.configure(highlightbackground="#d9d9d9") self.stopButton.configure(highlightcolor="black")

self.\_img3 = tk.PhotoImage(file="./icons/stop.png") self.stopButton.configure(image=self.\_img3) self.stopButton.configure(pady="0") self.stopButton.configure(text='''Button''')

self.vinylRecordImage = tk.Label(top) self.vinylRecordImage.place(relx=0.0, rely=0.0, height=204, width=204) self.vinylRecordImage.configure(background="#d9d9d9") self.vinylRecordImage.configure(disabledforeground="#a3a3a3") self.vinylRecordImage.configure(foreground="#000000")

self.\_img4 = tk.PhotoImage(file="./icons/vinylrecord.png")

self.vinylRecordImage.configure(image=self.\_img4) self.vinylRecordImage.configure(text='''Label''')

self.playList = ScrolledListBox(top) self.playList.place(relx=0.0, rely=0.38, relheight=0.532,

relwidth=0.999)

self.playList.configure(background="white") self.playList.configure(disabledforeground="#a3a3a3") self.playList.configure(font="TkFixedFont") self.playList.configure(foreground="black") self.playList.configure(highlightbackground="#d9d9d9") self.playList.configure(highlightcolor="#d9d9d9") self.playList.configure(selectbackground="#c4c4c4") self.playList.configure(selectforeground="black") self.playList.configure(width=10)

self.previousButton = tk.Button(top) self.previousButton.place(relx=0.509, rely=0.285, height=16, width=16) self.previousButton.configure(background="#fff") self.previousButton.configure(borderwidth="0") self.previousButton.configure(disabledforeground="#a3a3a3") self.previousButton.configure(foreground="#000000")

self.\_img5 = tk.PhotoImage(file="./icons/previous.png") self.previousButton.configure(image=self.\_img5) self.previousButton.configure(text='''Label''')

self.bottomBar = ttk.Label(top)

self.bottomBar.place(relx=0.0, rely=0.913, height=49, width=686) self.bottomBar.configure(background="#d9d9d9") self.bottomBar.configure(foreground="#000000") self.bottomBar.configure(font="TkDefaultFont") self.bottomBar.configure(relief='flat') self.bottomBar.configure(width=686) self.bottomBar.configure(state='disabled')

self.vol\_scale = ttk.Scale(top) self.vol\_scale.place(relx=0.015, rely=0.932, relwidth=0.146,

relheight=0.0

, height=26, bordermode='ignore') self.vol\_scale.configure(takefocus="")

self.addSongsToPlayListButton = tk.Button(top) self.addSongsToPlayListButton.place(relx=0.961, rely=0.323, height=17

, width=17) self.addSongsToPlayListButton.configure(activebackground="#ececec") self.addSongsToPlayListButton.configure(activeforeground="#d9d9d9") self.addSongsToPlayListButton.configure(background="#fff")

self.addSongsToPlayListButton.configure(borderwidth="0") self.addSongsToPlayListButton.configure(disabledforeground="#a3a3a3") self.addSongsToPlayListButton.configure(foreground="#000000") self.addSongsToPlayListButton.configure(highlightbackground="#d9d9d9") self.addSongsToPlayListButton.configure(highlightcolor="black") self.\_img6 = tk.PhotoImage(file="./icons/add.png") self.addSongsToPlayListButton.configure(image=self.\_img6) self.addSongsToPlayListButton.configure(pady="0") self.addSongsToPlayListButton.configure(text='''Button''')

")

")

a3")

9d9")

self.deleteSongsFromPlaylistButton = tk.Button(top) self.deleteSongsFromPlaylistButton.place(relx=0.917, rely=0.323

, height=18, width=18) self.deleteSongsFromPlaylistButton.configure(activebackground="#ececec

self.deleteSongsFromPlaylistButton.configure(activeforeground="#000000

self.deleteSongsFromPlaylistButton.configure(background="#fff") self.deleteSongsFromPlaylistButton.configure(borderwidth="0") self.deleteSongsFromPlaylistButton.configure(disabledforeground="#a3a3

self.deleteSongsFromPlaylistButton.configure(foreground="#000000") self.deleteSongsFromPlaylistButton.configure(highlightbackground="#d9d

self.deleteSongsFromPlaylistButton.configure(highlightcolor="black") self.\_img7 = tk.PhotoImage(file="./icons/delete.png") self.deleteSongsFromPlaylistButton.configure(image=self.\_img7) self.deleteSongsFromPlaylistButton.configure(pady="0") self.deleteSongsFromPlaylistButton.configure(text='''Button''')

self.addFavourite = tk.Button(top) self.addFavourite.place(relx=0.932, rely=0.913, height=42, width=42) self.addFavourite.configure(activebackground="#ececec") self.addFavourite.configure(activeforeground="#000000") self.addFavourite.configure(background="#d9d9d9") self.addFavourite.configure(borderwidth="0") self.addFavourite.configure(disabledforeground="#a3a3a3") self.addFavourite.configure(foreground="#000000") self.addFavourite.configure(highlightbackground="#d9d9d9") self.addFavourite.configure(highlightcolor="black")

self.\_img8 = tk.PhotoImage(file="./icons/like.png") self.addFavourite.configure(image=self.\_img8) self.addFavourite.configure(pady="0") self.addFavourite.configure(text='''Button''') self.addFavourite.configure(width=42)

self.removeFavourite = tk.Button(top)

self.removeFavourite.place(relx=0.873, rely=0.913, height=42, width=42) self.removeFavourite.configure(activebackground="#ececec") self.removeFavourite.configure(activeforeground="#000000") self.removeFavourite.configure(background="#d9d9d9") self.removeFavourite.configure(borderwidth="0") self.removeFavourite.configure(disabledforeground="#a3a3a3") self.removeFavourite.configure(foreground="#000000") self.removeFavourite.configure(highlightbackground="#d9d9d9") self.removeFavourite.configure(highlightcolor="black") self.\_img9 = tk.PhotoImage(file="./icons/broken-heart.png") self.removeFavourite.configure(image=self.\_img9) self.removeFavourite.configure(pady="0") self.removeFavourite.configure(text='''Button''') self.removeFavourite.configure(width=48)

self.loadFavourite = tk.Button(top) self.loadFavourite.place(relx=0.83, rely=0.932, height=26, width=26) self.loadFavourite.configure(activebackground="#ececec") self.loadFavourite.configure(activeforeground="#000000") self.loadFavourite.configure(background="#d9d9d9") self.loadFavourite.configure(borderwidth="0") self.loadFavourite.configure(disabledforeground="#a3a3a3") self.loadFavourite.configure(foreground="#000000") self.loadFavourite.configure(highlightbackground="#d9d9d9") self.loadFavourite.configure(highlightcolor="black")

self.\_img10 = tk.PhotoImage(file="./icons/refresh.png") self.loadFavourite.configure(image=self.\_img10) self.loadFavourite.configure(pady="0") self.loadFavourite.configure(text='''Button''') self.setup\_player()

DB")

tes)

defsetup\_player(self): try:

self.my\_player=Player.Player()

if self.my\_player.get\_db\_status(): messagebox.showinfo("Success!","Connected successfully to the

self.addFavourite.config(command=self.add\_song\_to\_favourites) self.loadFavourite.config(command=self.load\_songs\_from\_favouri

self.removeFavourite.config(command=self.remove\_song\_from\_favo else:

urites)

raise Exception("Sorry! u cannot save or load favourites")

except Exception as ex: messagebox.showerror("Error",ex)

print(traceback.format\_exc( )) self.addFavourite.config(state="disabled") self.loadFavourite.config(state="disabled") self.removeFavourite.config(state="disabled")

self.vol\_scale.config(from\_=0,to=100,command=self.change\_volume) self.vol\_scale.set(50) self.addSongsToPlayListButton.config(command=self.add\_song) self.deleteSongsFromPlaylistButton.config(command=self.remove\_song) self.playButton.config(command=self.play\_song) self.stopButton.config(command=self.stop\_song) self.pauseButton.config(command=self.pause\_song) self.playList.config(font="Vivaldi 12") self.playList.bind("<Double-1>",self.list\_double\_click) self.top.title("RHYTHM-Dance To The Rhythm of Your Heart! ") img=tk.PhotoImage(file="./icons/music1.png") self.top.iconphoto(self.top,img)

self.top.protocol("WM\_DELETE\_WINDOW",self.closewindow) # WM\_DELETE\_WINDOW method is Operating system method handled by window manager when we click on close symbol , to handle close window option we will call our method closewindow

self.isPaused=False self.isPlaying=False self.previousButton.config(command=self.load\_previous\_song)

self.my\_thread=None self.isThreadRunning=False self.stopThread=False

def change\_volume(self,val): volume\_level=float(val)/100 self.my\_player.set\_volume(volume\_level) print("Volume:",val)

def add\_song(self): song\_name=self.my\_player.add\_song() if song\_name is None:

return self.playList.insert(tk.END,song\_name) rcolor=lambda : random.randint(0,255) red=hex(rcolor()) green=hex(rcolor())

blue=hex(rcolor()) red=red[2:]

green=green[2:] blue=blue[2:] if len(red)==1:

red="0"+red if len(green)==1:

green="0"+green if len(blue)==1:

blue="0"+blue

mycolor="#"+red+green+blue self.playList.config(fg=mycolor)

def remove\_song(self): self.song\_index\_tuple=self.playList.curselection() try:

if len(self.song\_index\_tuple)==0:

raise NoSongSelectedError("Please select a song to remove") song\_name=self.playList.get(self.song\_index\_tuple[0]) self.playList.delete(self.song\_index\_tuple[0]) self.my\_player.remove\_song(song\_name)

except (NoSongSelectedError) as ex1: messagebox.showerror("Error!",ex1)

def show\_song\_details(self): self.song\_length=int(self.my\_player.get\_song\_length(self.song\_name)) min,sec=divmod(self.song\_length,60) # divmod function is used to do

the division and mod of a number .

self.songTotalDuration.config(text= str(min)+":"+str(sec)) self.songTimePassed.config(text="0:0") ext\_index=self.song\_name.rfind(".") song\_name\_str=self.song\_name[0:ext\_index] # song name without having

extension ".mp3"

if(len(song\_name\_str)>14): song\_name\_str=song\_name\_str[0:14]+". . ."

self.songName.configure(text=song\_name\_str)

def play\_song(self):

if self.isThreadRunning==True: self.my\_player.stop\_song() self.isThreadRunning=False self.isPlaying=False time.sleep(1)

self.sel\_song\_index\_tuple=self.playList.curselection()

try:

if len(self.sel\_song\_index\_tuple)==0:

raise NoSongSelectedError("Please select a song to play") self.song\_name=self.playList.get(self.sel\_song\_index\_tuple[0]) print("playing : ",self.song\_name) self.show\_song\_details() self.my\_player.play\_song()

self.change\_volume(self.vol\_scale.get()) # This step is

avoidable since it is changing the volume to 50 , after new song played therefore it is advisable to avoid it .

self.songProgress.config(length=self.song\_length) self.songProgress.config(maximum=self.song\_length) self.setup\_thread()

except (NoSongSelectedError) as ex1: messagebox.showerror("Error!",ex1)

def stop\_song(self):

if self.isThreadRunning==True: self.my\_player.stop\_song()

print("Stop playing : ",self.song\_name) self.stopThread=True self.isThreadRunning=False self.isPlaying=False

time.sleep(1)

def setup\_thread(self): self.my\_thread=threading.Thread(target=self.show\_timer,args=(self.song

\_length,),name="Hue Hue Hue ")

self.isPlaying=True self.isThreadRunning=True self.my\_thread.start()

def pause\_song(self): if self.isPlaying:

if self.isPaused:

self.my\_player.unpause\_song() # print("playing song") self.isPaused=False

else:

self.my\_player.pause\_song() # print("song paused") self.isPaused=True

def list\_double\_click(self,e):

if self.isThreadRunning==True: self.stopThread=True self.play\_song()

def closewindow(self):

result=messagebox.askyesno("App Closing!!!","Do you want to quit ?") if result: self.stopThread==True self.my\_player.close\_player() messagebox.showinfo("Have a good day!","Thank you for using

\"MOUZIKKA\" ") # Closing App message self.top.destroy()

defload\_previous\_song(self): try:

if self.isThreadRunning==True: self.stopThread=True

if hasattr(self,"sel\_song\_index\_tuple")==False:

raise NoSongSelectedError("Please select a song") self.prev\_song\_index=self.sel\_song\_index\_tuple[0]-1 if self.prev\_song\_index==-1:

print("size:",self.my\_player.get\_song\_count()) self.prev\_song\_index=self.my\_player.get\_song\_count()-1

self.playList.select\_clear(0,tk.END) self.playList.selection\_set(self.prev\_song\_index) self.play\_song()

except (NoSongSelectedError) as ex1: messagebox.showerror("Error!",ex1)

defload\_next\_song(self): try:

if hasattr(self,"sel\_song\_index\_tuple")==False:

raise NoSongSelectedError("Please select a song") self.next\_song\_index=self.sel\_song\_index\_tuple[0]+1

if self.next\_song\_index==self.my\_player.get\_song\_count(): self.next\_song\_index=0 self.playList.select\_clear(0,tk.END) self.playList.selection\_set(self.next\_song\_index) self.play\_song()

except (NoSongSelectedError) as ex1: messagebox.showerror("Error!",ex1)

def add\_song\_to\_favourites(self): fav\_song\_index\_tuple=self.playList.curselection()

try:

to favourites")

if len(fav\_song\_index\_tuple)==0:

raise NoSongSelectedError("Please select a song before adding

song\_name=self.playList.get(fav\_song\_index\_tuple[0]) result=self.my\_player.add\_song\_to\_favourites(song\_name) messagebox.showinfo("success!",result)

except (NoSongSelectedError) as ex1: messagebox.showerror("Error!",ex1)

except (DatabaseError) as ex2:

messagebox.showerror("Database error!","Song cannot be added!") print(traceback.format\_exc())

defload\_songs\_from\_favourites(self): try : load\_result=self.my\_player.load\_songs\_from\_favourites() result=load\_result[0]

if result.find("No songs present")!=-1: messagebox.showinfo("Favourites Empty!!!","No Song present in

your favourites")

return self.playList.delete(0,tk.END)

song\_dict=load\_result[1]

for song\_name in song\_dict: self.playList.insert(tk.END,song\_name) rcolor=lambda : random.randint(0,255)

red=hex(rcolor()) green=hex(rcolor()) blue=hex(rcolor()) red=red[2:] green=green[2:] blue=blue[2:] if len(red)==1:

red="0"+red

if len(green)==1: green="0"+green if len(blue)==1:

blue="0"+blue

mycolor="#"+red+green+blue self.playList.config(fg=mycolor) messagebox.showinfo("Favourites Loaded!!!","Songs loaded

successfully from favourites")

except (DatabaseError) as ex2:

messagebox.showerror("Database error!","Songs cannot be loaded!") print(traceback.format\_exc())

def remove\_song\_from\_favourites(self):

remove\_fav\_song\_index\_tuple=self.playList.curselection() try: if len(remove\_fav\_song\_index\_tuple)==0:

raise NoSongSelectedError("Please select a song before removing from

favourites")

song\_name=self.playList.get(remove\_fav\_song\_index\_tuple[0]) result=self.my\_player.remove\_song\_from\_favourites(song\_name) messagebox.showinfo("Removing",result)

if result.find("Song not present")==-1: self.song\_index\_tuple=self.playList.curselection() song\_name=self.playList.get(self.song\_index\_tuple[0]) self.playList.delete(self.song\_index\_tuple[0]) self.my\_player.remove\_song(song\_name)

except (NoSongSelectedError) as ex1: messagebox.showerror("Error!",ex1)

except (DatabaseError) as ex2:

messagebox.showerror("Database error!","Song cannot be removed from favourites !") print(traceback.format\_exc())

defshow\_timer(self,total\_sec): curr\_sec=1 self.songProgress.stop()

print(threading.current\_thread().getName(),"Activated!!!") print("No. of active threads:",threading.active\_count()) print("show\_timer thread name:",threading.current\_thread().getName()) while curr\_sec<=total\_sec:

min,sec=divmod(curr\_sec,60) self.songTimePassed.config(text=str(min)+":"+str(sec)) time.sleep(1)

curr\_sec+=1 self.songProgress.step() if self.stopThread==True:

break

if self.isPaused==True: print("Song Paused") while self.isPaused==True:

time.sleep(1) print("song resumed")

print("Previous Thread terminated!") if self.stopThread==False:

self.load\_next\_song() else: self.stopThread=False

# The following code is added to facilitate the Scrolled widgets you specified. class AutoScroll(object):

'''Configure the scrollbars for a widget.'''

def init (self, master):

# Rozen. Added the try-except clauses so that this class

# could be used for scrolled entry widget for which vertical # scrolling is not supported. 5/7/14.

try:

vsb = ttk.Scrollbar(master, orient='vertical', command=self.yview) except: pass

hsb = ttk.Scrollbar(master, orient='horizontal', command=self.xview)

#self.configure(yscrollcommand=\_autoscroll(vsb), # xscrollcommand=\_autoscroll(hsb))

try:

self.configure(yscrollcommand=self.\_autoscroll(vsb)) except: pass self.configure(xscrollcommand=self.\_autoscroll(hsb))

self.grid(column=0, row=0, sticky='nsew') try: vsb.grid(column=1, row=0, sticky='ns') except:

pass

hsb.grid(column=0, row=1, sticky='ew')

master.grid\_columnconfigure(0, weight=1) master.grid\_rowconfigure(0, weight=1)

# Copy geometry methods of master (taken from ScrolledText.py) methods = tk.Pack. dict .keys() | tk.Grid. dict .keys() \

| tk.Place. dict .keys()

for meth in methods:

if meth[0] != '\_' and meth not in ('config', 'configure'): setattr(self, meth, getattr(master, meth))

@staticmethod

def \_autoscroll(sbar):

'''Hide and show scrollbar as needed.''' def wrapped(first, last):

first, last = float(first), float(last)

if first <= 0 and last >= 1: sbar.grid\_remove() else:

sbar.grid() sbar.set(first, last) return wrapped

def str (self):

return str(self.master)

def \_create\_container(func):

'''Creates a ttk Frame with a given master, and use this new frame to place the scrollbars and the widget.'''

def wrapped(cls, master, \*\*kw): container = ttk.Frame(master)

container.bind('<Enter>', lambda e: \_bound\_to\_mousewheel(e, container)) container.bind('<Leave>', lambda e: \_unbound\_to\_mousewheel(e, container)) return func(cls, container, \*\*kw) return wrapped

class ScrolledListBox(AutoScroll, tk.Listbox):

'''A standard Tkinter Text widget with scrollbars that will automatically show/hide as needed.''' @\_create\_container

def init (self, master, \*\*kw): tk.Listbox. init (self, master, \*\*kw) AutoScroll. init (self, master)

import platform

def \_bound\_to\_mousewheel(event, widget): child = widget.winfo\_children()[0]

if platform.system() == 'Windows' or platform.system() == 'Darwin': child.bind\_all('<MouseWheel>', lambda e: \_on\_mousewheel(e, child)) child.bind\_all('<Shift-MouseWheel>', lambda e: \_on\_shiftmouse(e,

child))

else:

child.bind\_all('<Button-4>', lambda e: \_on\_mousewheel(e, child)) child.bind\_all('<Button-5>', lambda e: \_on\_mousewheel(e, child)) child.bind\_all('<Shift-Button-4>', lambda e:

\_on\_shiftmouse(e, child)) child.bind\_all('<Shift-Button-5>', lambda e: \_on\_shiftmouse(e, child))

def \_unbound\_to\_mousewheel(event, widget):

if platform.system() == 'Windows' or platform.system() == 'Darwin': widget.unbind\_all('<MouseWheel>') widget.unbind\_all('<Shift-MouseWheel>') else:

widget.unbind\_all('<Button-4>') widget.unbind\_all('<Button-5>') widget.unbind\_all('<Shift-Button-4>') widget.unbind\_all('<Shift-Button-5>')

def \_on\_mousewheel(event, widget):

if platform.system() == 'Windows':

widget.yview\_scroll(-1\*int(event.delta/120),'units') elif platform.system() == 'Darwin':

widget.yview\_scroll(-1\*int(event.delta),'units') else:

if event.num == 4: widget.yview\_scroll(-1, 'units')

elif event.num == 5: widget.yview\_scroll(1, 'units')

def \_on\_shiftmouse(event, widget):

if platform.system() == 'Windows':

widget.xview\_scroll(-1\*int(event.delta/120), 'units') elif platform.system() == 'Darwin':

widget.xview\_scroll(-1\*int(event.delta), 'units') else:

if event.num == 4: widget.xview\_scroll(-1, 'units')

elif event.num == 5: widget.xview\_scroll(1, 'units')

if name == ' main ': vp\_start\_gui()

#### Player..py

This file will manage the actual playing / pausing / rewind of songs. Also as mentioned , the View and Model do not directly communicate with each other. So their communication is done via Controller which is the file Player.py. It contains methods like play\_song(), stop\_song(), pause\_song() as well as bridge methods like add\_song\_to\_favourite(),load\_songs\_from\_favourite() etc to pass on the user's request to the Model

import Model

from pygame import mixer

from tkinter import filedialog import os

from mutagen.mp3 import MP3

class Player:

def init (self):

mixer.init() self.my\_model=Model.Model()

def get\_db\_status(self):

return self.my\_model.get\_db\_status()

def close\_player(self): mixer.music.stop() self.my\_model.close\_db\_connection()

def set\_volume(self,volume\_level): mixer.music.set\_volume(volume\_level)

def add\_song(self): song\_path=filedialog.askopenfilename(title="select your song...",filetype=[("mp3 files","\*mp3")]) if

song\_path=="":

return song\_name=os.path.basename(song\_path) self.my\_model.add\_song(song\_name,song\_path) return song\_name

def remove\_song(self,song\_name): self.my\_model.remove\_song(song\_name)

def get\_song\_length(self,song\_name): self.song\_path=self.my\_model.get\_song\_path(song\_name) self.audio\_tag=MP3(self.song\_path) song\_length=self.audio\_tag.info.length return song\_length

defplay\_song(self): mixer.quit()

mixer.init(frequency=self.audio\_tag.info.sample\_rate) mixer.music.load(self.song\_path) mixer.music.play()

def stop\_song(self): mixer.music.stop() def pause\_song(self): mixer.music.pause()

def unpause\_song(self): mixer.music.unpause()

def add\_song\_to\_favourites(self,song\_name): song\_path=self.my\_model.get\_song\_path(song\_name)

result=self.my\_model.add\_song\_to\_favourites(song\_name,song\_path) return result

def load\_song\_from\_favourites(self): result=self.my\_model.load\_songs\_from\_favourites() return result,self.my\_model.song\_dict

def remove\_songs\_from\_favourites(self,song\_name): result=self.my\_model.remove\_song\_from\_favourites(song\_name) return result

### Chapter :-6 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 five options — 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.

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 name, singer’s name, duration of the song, size of the file, etc.

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 and resume 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 concluded.